

---

# **AMICI Documentation**

***Release 0.20.0***

**The AMICI developers**

**Nov 21, 2023**



# ABOUT

<b>1</b>	<b>About AMICI</b>	<b>3</b>
1.1	Features . . . . .	3
1.2	Interfaces & workflow . . . . .	4
<b>2</b>	<b>Availability</b>	<b>5</b>
2.1	Source code . . . . .	5
2.2	Python package . . . . .	5
2.3	Installation instructions . . . . .	5
<b>3</b>	<b>License conditions</b>	<b>7</b>
3.1	AMICI . . . . .	7
3.2	AMICI Logo . . . . .	7
3.3	Dependencies . . . . .	8
<b>4</b>	<b>How to cite AMICI</b>	<b>9</b>
<b>5</b>	<b>References</b>	<b>11</b>
<b>6</b>	<b>Background</b>	<b>13</b>
6.1	Publications on various features of AMICI . . . . .	13
6.2	Third-Party numerical algorithms used by AMICI . . . . .	14
<b>7</b>	<b>Changelog</b>	<b>15</b>
7.1	v0.X Series . . . . .	15
7.2	What's Changed . . . . .	24
<b>8</b>	<b>Glossary</b>	<b>55</b>
<b>9</b>	<b>Contributing</b>	<b>57</b>
9.1	Contributing to AMICI . . . . .	57
9.2	Code of Conduct . . . . .	57
<b>10</b>	<b>Python interface</b>	<b>59</b>
10.1	Installing the AMICI Python package . . . . .	59
10.2	Examples . . . . .	65
10.3	Using AMICI's Python interface . . . . .	257
10.4	FAQ . . . . .	261
10.5	AMICI Python API . . . . .	262
<b>11</b>	<b>C++ interface</b>	<b>405</b>
11.1	Building the C++ library . . . . .	405

11.2	Using AMICI's C++ interface . . . . .	406
11.3	AMICI C++ API . . . . .	408
<b>12</b>	<b>Matlab interface</b>	<b>685</b>
12.1	Installing the AMICI MATLAB toolbox . . . . .	685
12.2	Using AMICI's MATLAB interface . . . . .	685
12.3	FAQ . . . . .	692
12.4	AMICI Matlab API . . . . .	693
12.5	AMICI developer's guide . . . . .	726
12.6	Handling of Discontinuities . . . . .	734
<b>13</b>	<b>Indices and tables</b>	<b>737</b>
	<b>Python Module Index</b>	<b>739</b>
	<b>Index</b>	<b>741</b>

Version: 0.20.0

Source code: <https://github.com/AMICI-dev/amici>



## ABOUT AMICI

AMICI provides a multi-language (Python, C++, Matlab) interface to the *SUNDIALS* solvers *CVODES* (for *ODEs*) and *IDAS* (for *DAEs*). AMICI allows the user to read differential equation models specified as *SBML* or *PySB* and automatically compiles such models into Python modules, C++ libraries or *.mex* simulation files (Matlab).

In contrast to the (no longer maintained) *sundialsTB* Matlab interface, all necessary functions are transformed into native C++ code, which allows for a significantly faster simulation.

Beyond forward integration, the compiled simulation file also allows for forward sensitivity analysis, steady state sensitivity analysis and adjoint sensitivity analysis for likelihood-based output functions.

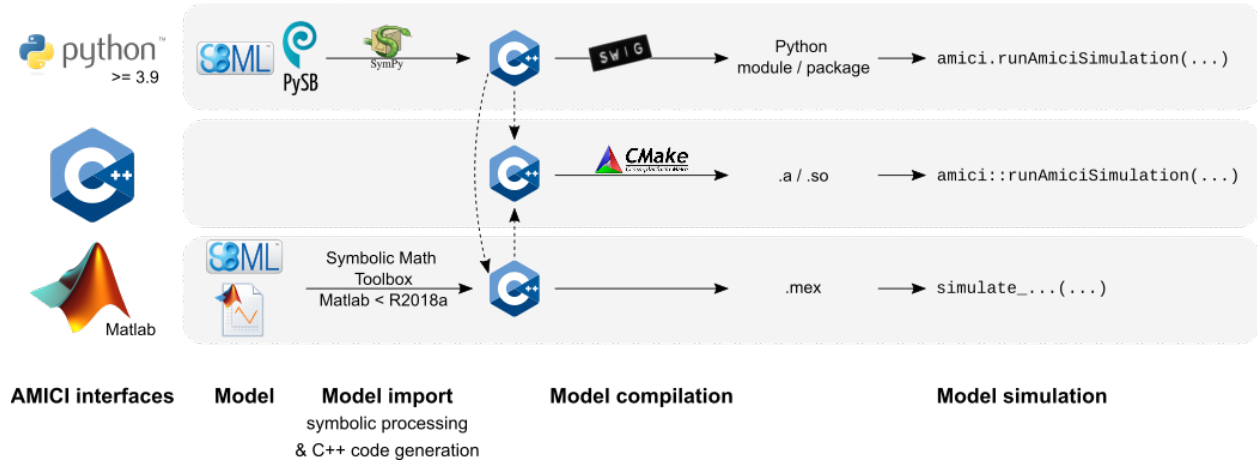
The interface was designed to provide routines for efficient gradient computation in parameter estimation of biochemical reaction models but it is also applicable to a wider range of differential equation constrained optimization problems.

## 1.1 Features

- *SBML* import
- *PySB* import
- Generation of C++ code for model simulation and sensitivity computation
- Access to and high customizability of *CVODES* and *IDAS* solver
- Python, C++, Matlab interface
- Sensitivity analysis
  - forward
  - steady state
  - adjoint
  - first- and second-order (second-order Matlab-only)
- Pre-equilibration and pre-simulation conditions
- Support for discrete events and logical operations

## 1.2 Interfaces & workflow

The AMICI workflow starts with importing a model from either *SBML* (Matlab, Python), *PySB* (Python), or a Matlab definition of the model (Matlab-only). From this input, all equations for model simulation are derived symbolically and C++ code is generated. This code is then compiled into a C++ library, a Python module, or a Matlab *.mex* file and is then used for model simulation.



The functionality of the Python, Matlab and C++ interfaces slightly differ, as shown in the following table:

Feature \ Interface	Python	C++	Matlab
<i>SBML</i> import	yes ( <i>details</i> )	no	yes (<=R2017b)
<i>PySB</i> import	yes	no	no
<i>DAE</i> import	no	no	yes
Forward sensitivities	yes	yes	yes
Adjoint sensitivities	yes	yes	yes
Steadystate sensitivities	yes	yes	yes
Second-order sensitivities	no	no	yes
Events	yes	yes	yes
<i>preequilibration</i>	yes	yes	yes
<i>presimulation</i>	yes	yes	no



## AVAILABILITY

### 2.1 Source code

The AMICI source code is available as

- [tar archive](#)
- [zip archive](#)
- Git repository on [GitHub](#)

If AMICI was downloaded as an archive, it needs to be unpacked. If AMICI was obtained via cloning the Git repository, no further unpacking is necessary.

#### 2.1.1 Obtaining AMICI via the Git version control system

In order to always stay up-to-date with the latest AMICI versions, simply pull it from our Git repository and recompile it when a new version is available. For more information about Git, check out their [website](#).

The Git repository can currently be found at <https://github.com/AMICI-dev/AMICI> and clone is done via:

```
git clone https://github.com/AMICI-dev/AMICI.git AMICI
```

### 2.2 Python package

A Python package is available on [PyPI](#).

### 2.3 Installation instructions

Installation instructions are available for

- *Python*
- *C++*
- *Matlab*



## LICENSE CONDITIONS

### 3.1 AMICI

AMICI is released under the 3-Clause BSD License (BSD-3-Clause) with the following terms:

Copyright (c) 2015-2020, Fabian Fröhlich, Jan Hasenauer, Daniel Weindl and Paul Stapor All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

### 3.2 AMICI Logo

The AMICI logo is released under the Creative Commons CC0 1.0 Universal (CC0 1.0) license with the terms given in [documentation/gfx/LICENSE.md](#).

### 3.3 Dependencies

- Parts of the *SUNDIALS* solver suite are redistributed under the BSD 3-Clause License (BSD-3-Clause) with terms given in `ThirdParty/sundials/LICENSE`
- Parts of *SuiteSparse* are redistributed under the various licenses with the terms given in `ThirdParty/SuiteSparse/LICENSE.txt`
- *gsl-lite* is redistributed under the MIT License (MIT) with the terms given in `ThirdParty/gsl/gsl/gsl-lite.hpp`
- *xml2struct* and *struct2xml* are redistributed under the BSD 2-Clause License (BSD-2-Clause) with terms given in `matlab/auxiliary/xml2struct/license.txt` and `matlab/auxiliary/struct2xml/license.txt`
- *CalcMD5* is redistributed under the BSD 2-Clause License (BSD-2-Clause) with terms given in `matlab/auxiliary/CalcMD5/license.txt`

## HOW TO CITE AMICI

### Citable DOI for the latest AMICI release:

There is a list of [publications using AMICI](#). If you used AMICI in your work, we are happy to include your project, please let us know via a Github issue.

When using AMICI in your project, please cite

- Fröhlich, F., Weindl, D., Schälte, Y., Pathirana, D., Paszkowski, Ł., Lines, G.T., Stapor, P. and Hasenauer, J., 2021. AMICI: High-Performance Sensitivity Analysis for Large Ordinary Differential Equation Models. *Bioinformatics*, btab227, DOI:10.1093/bioinformatics/btab227.

```
@article{frohlich2020amici,  
  title={AMICI: High-Performance Sensitivity Analysis for Large Ordinary  
↪ Differential Equation Models},  
  author={Fröhlich, Fabian and Weindl, Daniel and Schälte, Yannik and  
↪ Pathirana, Dilan and Paszkowski, Łukasz and Lines, Glenn Terje and Stapor,  
↪ Paul and Hasenauer, Jan},  
  journal = {Bioinformatics},  
  year = {2021},  
  month = {04},  
  issn = {1367-4803},  
  doi = {10.1093/bioinformatics/btab227},  
  note = {btab227},  
  eprint = {https://academic.oup.com/bioinformatics/advance-article-pdf/doi/10.1093/  
↪ bioinformatics/btab227/36866220/btab227.pdf},  
}
```

When presenting work that employs AMICI, feel free to use one of the icons in [documentation/gfx/](#), which are available under a [CC0](#) license:





## REFERENCES

List of publications using AMICI. Total number is 82.

If you applied AMICI in your work and your publication is missing, please let us know via a new GitHub issue.





## BACKGROUND

*This section is to be extended.*

### 6.1 Publications on various features of AMICI

Some mathematical background for AMICI is provided in the following publications:

- Fröhlich, F., Kaltenbacher, B., Theis, F. J., & Hasenauer, J. (2017). **Scalable Parameter Estimation for Genome-Scale Biochemical Reaction Networks.** PLOS Computational Biology, 13(1), e1005331. doi:[10.1371/journal.pcbi.1005331](https://doi.org/10.1371/journal.pcbi.1005331).
- Fröhlich, F., Theis, F. J., Rädler, J. O., & Hasenauer, J. (2017). **Parameter estimation for dynamical systems with discrete events and logical operations.** Bioinformatics, 33(7), 1049-1056. doi:[10.1093/bioinformatics/btw764](https://doi.org/10.1093/bioinformatics/btw764).
- Terje Lines, Glenn, Łukasz Paszkowski, Leonard Schmiester, Daniel Weindl, Paul Stapor, and Jan Hasenauer. 2019. **Efficient Computation of Steady States in Large-Scale ODE Models of Biochemical Reaction Networks.** IFAC-PapersOnLine 52 (26): 32–37. DOI: [10.1016/j.ifacol.2019.12.232](https://doi.org/10.1016/j.ifacol.2019.12.232).
- Stapor, Paul, Fabian Fröhlich, and Jan Hasenauer. 2018. **Optimization and Profile Calculation of ODE Models Using Second Order Adjoint Sensitivity Analysis.** Bioinformatics 34 (13): i151–i159. DOI: [10.1093/bioinformatics/bty230](https://doi.org/10.1093/bioinformatics/bty230).
- Lakrisenko, Polina, Paul Stapor, Stephan Grein, Łukasz Paszkowski, Dilan Pathirana, Fabian Fröhlich, Glenn Terje Lines, Daniel Weindl, and Jan Hasenauer. 2022. **Efficient Computation of Adjoint Sensitivities at Steady-State in ODE Models of Biochemical Reaction Networks.** bioRxiv 2022.08.08.503176. DOI: [10.1101/2022.08.08.503176](https://doi.org/10.1101/2022.08.08.503176).
- L. Contento, P. Stapor, D. Weindl, and J. Hasenauer, “A more expressive spline representation for SBML models improves code generation performance in AMICI,” bioRxiv, 2023, DOI: [10.1101/2023.06.29.547120](https://doi.org/10.1101/2023.06.29.547120).

---

**Note:** Implementation details of the latest AMICI versions may differ from the ones given in the references manuscripts.

---

## 6.2 Third-Party numerical algorithms used by AMICI

AMICI uses the following packages from SUNDIALS:

- CVODES:

The sensitivity-enabled ODE solver in SUNDIALS. Radu Serban and Alan C. Hindmarsh. *ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2005. [PDF](#)

- IDAS

AMICI uses the following packages from SuiteSparse:

- Algorithm 907: **KLU** A Direct Sparse Solver for Circuit Simulation Problems. Timothy A. Davis, Ekanathan Palamadai Natarajan, *ACM Transactions on Mathematical Software*, Vol 37, Issue 6, 2010, pp 36:1-36:17. [PDF](#)
- Algorithm 837: **AMD**, an approximate minimum degree ordering algorithm, Patrick R. Amestoy, Timothy A. Davis, Iain S. Duff, *ACM Transactions on Mathematical Software*, Vol 30, Issue 3, 2004, pp 381-388. [PDF](#)
- Algorithm 836: **COLAMD**, a column approximate minimum degree ordering algorithm, Timothy A. Davis, John R. Gilbert, Stefan I. Larimore, Esmond G. Ng *ACM Transactions on Mathematical Software*, Vol 30, Issue 3, 2004, pp 377-380. [PDF](#)

Others:

- SuperLU\_MT

“A general purpose library for the direct solution of large, sparse, nonsymmetric systems of linear equations” ([https://crd-legacy.lbl.gov/~xiaoye/SuperLU/#superlu\\_mt](https://crd-legacy.lbl.gov/~xiaoye/SuperLU/#superlu_mt)). SuperLU\_MT is optional and is so far only available from the C++ interface.

## CHANGELOG

### 7.1 v0.X Series

#### 7.1.1 v0.20.0 (2023-11-23)

##### Fixes

- Fixed CMake `cmake_minimum_required` deprecation warning by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2183>
- Fixed misleading preequilibration failure messages by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2181>
- Removed `setuptools<64` restriction by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2180>
- Fixed `ExpData` equality operator for Python by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2194>
- Enabled `deepcopy` for `ExpData(View)` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2196>
- Allowed subsetting simulation conditions in `simulate_petab` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2199>
- Set CMake `CMP0144` to prevent warning by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2209>

##### Features

- Possibility to evaluate and plot symbolic expressions based on simulation results by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2152>
- Easier access to timepoints via `ExpDataView` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2193>
- Nicer `__repr__` for `ReturnDataView` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2192>

##### Documentation

- Added installation instructions for Arch Linux by @stephanmg in <https://github.com/AMICI-dev/AMICI/pull/2173>
- Updated reference list by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2172>
- Installation guide: optional requirements by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2207>

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.19.0...v0.20.0>

### 7.1.2 v0.19.0 (2023-08-26)

#### Features

- SBML import now supports `rateOf` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2120>
- Added `Model.{get,set}SteadyStateComputationMode` (analogous to `SteadyStateSensitivityMode`) which allows to choose how steady state is computed. by @plakrisenko in <https://github.com/AMICI-dev/AMICI/pull/2074>

**Note:** The default `SteadyStateSensitivityMode` changed from `newtonOnly` to `integrateIfNewtonFails`.

- SBML import: Allow hardcoding of numerical values by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2134>
- Added `antimony2amici` for more convenient antimony import (simplifies working with raw ODEs, see documentation) by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2142>
- Added `AMICI_TRY_ENABLE_HDF5` environment variable to control whether to search for HDF5 or not by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2148>

#### Fixes

- Fixed SBML import for events with trigger functions depending on parameters that are initial assignment targets by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2145>
- Fixed SBML import for event-assigned parameters with non-float initial assignments by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2156>
- Fixed `unistd.h` dependency of `hdf5.cpp` that led to compilation failures on Windows by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2154>
- Set CMake policies for cmake 3.27 by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2162>
- Fixed a `lib/` vs `lib64/` issue, leading to `SUNDIALSConfig.cmake-not-found` issues on some systems by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2165>
- CMake: fixed scope of `-DHAS_BOOST_CHRONO` which may have lead to a mix of `boost::chrono::thread_clock` and `std::clock` being used in programs using amici, and potentially segmentation faults by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2163>

#### Performance:

- Combined code for sparse model functions and their index files for slightly faster compilation of small models by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2159>
- Removed complex / complex long KLU functions for slightly faster amici package installation by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2160>

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.18.1...v0.19.0>

### 7.1.3 v0.18.1 (2023-06-26)

Fixes:

- Fixed pysb pattern matching during PETab import by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/2118>
- Fixed `sp.Matrix` errors with `numpy==1.25` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2124>
- Readme: added info containers by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2125>
- Fixed deprecation warnings by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2122>  
<https://github.com/AMICI-dev/AMICI/pull/2131>
- Fixed logging typo in SBML import by @dilpath in <https://github.com/AMICI-dev/AMICI/pull/2126>
- Added minimum version for pandas by @dilpath in <https://github.com/AMICI-dev/AMICI/pull/2129>

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.18.0...v0.18.1>

### 7.1.4 v0.18.0 (2023-05-26)

Features:

- More efficient handling of splines in SBML models by @paulstapor, @lcontento, @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1515>
- Partial support of current PETab2.0 draft, including support for PySB models by @dweindl, @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1800>

Fixes:

- **Fixed incorrect forward sensitivities for models with events with state-dependent trigger functions** by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2084>
- Model import: Don't create `spl.h` and `sspl.h` for models without splines by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2088>
- SBML import - faster processing of SpeciesReference IDs by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2094>
- Update swig ignores by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/2098>
- CMake: Fixed choosing SWIG via SWIG env variable by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2100>
- CMake: Try FindBLAS if no other information was provided by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2104>
- Fixed cblas error for models without solver states in combination with forward sensitivities by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2108>
- Fixed compilation error for models with events and `xdot=0` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2111>
- Fixed import error for models with events and 0 states by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2112>

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.17.1...v0.18.0>

### 7.1.5 v0.17.1 (2023-05-10)

This release fixes two bugs:

- One bug introduced in v0.17.0, that causes an `ImportError` on macOS (<https://github.com/AMICI-dev/AMICI/issues/2075>).
- An `AttributeError` in `petab_import_pysb` with `petab>=0.2.0` <https://github.com/AMICI-dev/AMICI/pull/2079>

### 7.1.6 v0.17.0 (2023-05-09)

AMICI v0.17.0 requires Python $\geq$ 3.9 and a C++17 compatible compiler

Features

- DAE support in SBML by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/2017>
- SBML import: flatten SBML-comp models by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2063>
- Added `sllh` computation back to `petab_objective.simulate_petab` by @dilpath in <https://github.com/AMICI-dev/AMICI/pull/1548>
- CMake-based Python extension builds by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1992>

Fixes

- Fixed CPU time tracking with multi-threading (partially) by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2023>
- Fixed HDF5 ambiguous overload by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2031>
- Fixed varying cmake libdir `lib(64)/` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2033>
- Fixed Equilibration cpu time computation by @plakrisenko in <https://github.com/AMICI-dev/AMICI/pull/2035>
- CMake: add header files to library sources for generated models by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2047>
- CMake: Handle header-dependency of swig files by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2046>
- Don't try to detect conservation laws for models with `Species-AssignmentRules` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2056>
- Smith benchmark and SBML initialization fix by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/2034>
- SBML import: Fixed check for required packages by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2064>
- Nan observables by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/2065>
- Fixed check for discontinuities for conservation law computation by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2068>
- Specify visualization dependencies by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2070>
- Fixed sympy symbol name clashes during PETab import by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2069>
- Fixed `ReturnData::{preeq_wrms,posteq_wrms}` with FSA and `check_sensi_steadystate_conv_=True` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2071>

Extended / updated documentation, for example:

- Jax example notebook by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1996>
- Updated Windows/MSVC installation instructions by @Podde1 in <https://github.com/AMICI-dev/AMICI/pull/2053>

#### New Contributors

- @Podde1 made their first contribution in <https://github.com/AMICI-dev/AMICI/pull/2053>

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.16.1...v0.17.0>

### 7.1.7 v0.16.1 (2023-02-24)

#### Fixes:

- Additional package names for finding blas via pkg-config by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1959>
- Changed default interpolation type from hermite to polynomial by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1960>
- PySB import: Change default simplify to work with multiprocessing by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1961>
- Add `--no-validate` to `amici_import_petab` @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1963>
- Fix `get_model` for direct import of swig interface by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1969>
- Fix `PytestReturnNotNoneWarning` in `test_conserved_quantities_demartino.py` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1968>
- Fix MSVC builds / remove `-W*` flags by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1972>
- Add option to use IDs when plotting trajectories by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1974>
- Fix `assignmentRules2observables` - skip non-assignment-rule targets by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1973>
- Use `std::clock` for measuring solver time by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1982> (*Note that this uses cpu-time consumed by all threads*)
- Fix `narrowing-conversion-warning` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1983>
- PETab import: allow specifying default values for output parameters by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1987>
- Print stacktraces only with debug logging level by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1985>
- Change default `ReturnData::status` to `AMICI_NOT_RUN` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1984>
- Reduce time-tracking overhead by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1988>
- Fix `equilibraton` status discrepancy by @plakrisenko in <https://github.com/AMICI-dev/AMICI/pull/1991>
- Pass `model_name` to `_create_model_output_dir_name` by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1994>
- CMake: Build with OpenMP support if available by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2000>
- Fix SuiteSparse Makefiles for compiler-paths by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2003>

- CMake: Build with HDF5 support if available by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1999>
- CMake: Fix reading version file on Windows by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2001>
- CMake: raise minimum required version to 3.15 by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2002>
- Fix/extend runtime logging by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/2005>
- Fix error logging in steadystate solver by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/2008>
- Don't pass -py3 to swig after 4.1.0 by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/2010>
- SWIG \_\_repr\_\_s for different templated vector classes by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/2009>
- Matlab: If mex fails, print mex arguments by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2013>
- Simplify OpenBLAS installation on Windows by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2016>
- Remove model name prefix in generated model files by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2015>
- ...

#### Documentation:

- Restructure sphinx doc by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1978>
- Instructions for AMICI with singularity by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1964>
- Illustrate options for potentially speeding up model import/simulation by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1965>
- ...

#### Dependencies:

- Updated SuiteSparse to v7.0.1 by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/2018>

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.16.0...v0.16.1>

## 7.1.8 v0.16.0 (2023-01-25)

### Features

- Python 3.11 compatibility by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1876>
- AMICI now runs on binder (<https://mybinder.org/v2/gh/AMICI-dev/AMICI/develop?labpath=binder%2Foverview.ipynb>) by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1935>, <https://github.com/AMICI-dev/AMICI/pull/1937>, <https://github.com/AMICI-dev/AMICI/pull/1939>
- More informative Solver.\_\_repr\_\_ and ExpData.\_\_repr\_\_ by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1928> and @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1948>
- simulate\_petab returns the generated/used ExpDatas by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1933>
- Model module is now accessible from model instance by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1932>
- Added plot\_jacobian by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1930>
- Now logs all nested execution times as debug by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1947>



- Always check for finite initial states, not only with `Model.setAlwaysCheckFinite(True)` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1955>

#### Fixes

- `ReturnDataView.status` now returns `int` instead of `float` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1929>
- Updated simulation status codes by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1931>
- Skip irrelevant frames in stacktraces by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1934>
- Fixed compiler warning (matlab) by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1954>

#### Documentation:

- Added a notebook demonstrating common simulation failures and show how to analyze / fix them by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1946>
- various minor fixes / updates

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.15.0...v0.16.0>

### 7.1.9 v0.15.0 (2023-01-11)

#### Features

- Improved logging by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1907>

For Python: Don't print messages to stdout, but collect them in `ReturnData` and forward them to python logging, making it easier to filter specific messages or to disable output completely. Messages are also available via `ReturnData.messages`.

**breaking change for C++ interface:** Messages aren't printed to stdout by default, but are collected in `ReturnData`. The user has to decide what to do with them.

- MultiArch docker build by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1903>
- Added cmake target for cmake-format by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1909>
- Updated clang-format style, fixed clang-format target by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1908>
- Subsetting `ReturnData` fields by ID via `ReturnDataView.by_id` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1911> <https://github.com/AMICI-dev/AMICI/pull/1916>

#### Fixes

- PETab import: fixed handling of fixed parameters for rule targets by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1915>
- Fixed compiler warnings for matlab interface by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1919>
- Fixed pandas DeprecationWarning for `Series.iteritems()` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1921>
- Fixed circular import in `amici.petab_import_pysb` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1922>
- Fixed 'operator ==' swig warning by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1923>
- Prevent swig4.0.1 segfault by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1924>

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.14.0...v0.15.0>

### 7.1.10 v0.14.0 (2022-11-23)

#### Features:

- Added optional functionality to apply C99 math optimization to generated C++ code by @dweindl and @lcontento in <https://github.com/AMICI-dev/AMICI/pull/1377>, <https://github.com/AMICI-dev/AMICI/pull/1878>
- Added option to treat fixed parameters as constants in PETab import by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1877>
- Added equality operator for ExpData by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1881>
- Updated base image for Dockerfile to Ubuntu 22.04/Python 3.10 by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1896>

#### Fixes:

- Fixed deprecation warnings by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1873>, <https://github.com/AMICI-dev/AMICI/pull/1893>
- Fixes/updates to GitHub actions by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1885>, <https://github.com/AMICI-dev/AMICI/pull/1893>, <https://github.com/AMICI-dev/AMICI/pull/1889>, <https://github.com/AMICI-dev/AMICI/pull/1891>
- Added hdf5 search directories for arm64 architecture (M1/M2 macs) by @Doresic in <https://github.com/AMICI-dev/AMICI/pull/1894>
- Fixed missing return in generated non-void functions by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1892>
- Fixed import failure for pre-compiled models by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1897>

#### Documentation:

- Update reference list by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1874>, <https://github.com/AMICI-dev/AMICI/pull/1884>

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.13.0...v0.14.0>

### 7.1.11 v0.13.0 (2022-10-04)

- Fixed extraction of common subexpressions by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1865>
- Added function to convert `ReturnData::status` flags to string by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1864>

And further contributions by @dweindl, @FFroehlich

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.12.0...v0.13.0>

### 7.1.12 v0.12.0 (2022-08-26)

Features:

- Support for event observables via the Python interface by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1845>
- Treat non-estimated parameters as constants during SBML-PETab import by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1810>
- Updated SUNDIALS to v5.8.0 by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1836>
- Option to extract common subexpressions by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1852>, <https://github.com/AMICI-dev/AMICI/pull/1856> **not available in this release, use v0.13.0**
- Parallelize matrix simplification by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1778>
- Validate PETab problems before attempting import by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1842>
- Improved type annotations for the swig interface by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1860>

Fixes:

- Fixed an issue with potentially infinite loops during conservation law processing by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1833>
- Fixed potential deadlocks during parallel simplification by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1844>
- Fix resetting `ReturnData::numstepsB` when re-using Solver by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1841>

And further contributions by @dilpath, @dweindl, @FFroehlich

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.11.32...v0.12.0>

### 7.1.13 v0.11.32 (2022-07-15)

Fixes:

- Fixed `ImportErrors` during package installation with recent `setuptools` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1830>

### 7.1.14 v0.11.31 (2022-07-12)

Fixes:

- Fixed `ParameterMapping.__getitem__` to either return a `ParameterMappingForCondition` or a new `ParameterMapping`, but not a list by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1826>

### 7.1.15 v0.11.30 (2022-07-07)

#### Features:

- Allow overriding model name during PySB import by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1801>
- Added **repr** for parameter mapping classes by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1799>
- More informative warning messages for NaNs/Infs by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1798>
- Moved `sim_steps` increment by @plakrisenko in <https://github.com/AMICI-dev/AMICI/pull/1806>
- Re-arranged application of parameters from `ExpData` to avoid initial sensitivities error by @dilpath in <https://github.com/AMICI-dev/AMICI/pull/1805>
- Checking for unused parameters in `simulate_petab` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1816>
- Add `create_parameter_mapping` kwarg forwarding by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1820>

#### Other

- Remove `constant_species_to_parameters` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1809>

#### Fixes

- Fixed handling of SBML models given as `pathlib.Path` in ``petab_import.import_model_sbml` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1808>
- Fixed missing CPU time reset by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1814>
- Raise in `simulate_petab` with `scaled_parameters=True` `problem_parameters=None` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1819>

...

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.11.29...v0.11.30>

### 7.1.16 v0.11.29 (2022-05-06)

## 7.2 What's Changed

#### Features:

- Performance: Limit newton step convergence check by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1780>
- More informative NaN/Inf warnings by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1640>
- SBML import can now handle initial events by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1789>

#### Fixes:

- Avoid error if no measurements in PETab problem; fixed type handling in PETab parameter mapping by @dilpath in <https://github.com/AMICI-dev/AMICI/pull/1783>
- Fixed substitution of expressions in `root` and `stau` by @dilpath in <https://github.com/AMICI-dev/AMICI/pull/1784>

- Workaround for PETab problems with state-dependent noise models by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1791>

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.11.28...v0.11.29>

### 7.2.1 v0.11.28 (2022-04-08)

New features:

- Added `Solver.setSteadyStateToleranceFactor` and `Solver.setSteadyStateSensiToleranceFactor` to specify a steady state tolerance factor by @dilpath in <https://github.com/AMICI-dev/AMICI/pull/1758>

**NOTE: This also relaxed the default steady state (sensitivity) tolerances by a factor of 100.**

- Added support for `pathlib.Path` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1769>
- Allow specifying initial timepoint with `ExpData` by @dilpath in <https://github.com/AMICI-dev/AMICI/pull/1776>

Performance:

- Speedup for models with conservation laws by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1765>
- Improved efficiency of newton step convergence check by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1775>

Fixes:

- Fixed deprecation warning for `pandas.DataFrame.append` in `rdatas_to_measurement_df` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1770>
- Fixed Rule-target handling in PETab import by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1753>

Removed functionality:

- Removed long deprecated `sbml2amici` arguments `modelName` and `constantParameters` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1774>

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.11.27...v0.11.28>

### 7.2.2 v0.11.27 (2022-04-04)

New features:

- Checking condition number when computing sensitivities via Newton by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1730>
- Removed SPBCG solver by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1729>
- Added Newton step convergence checks to steadystate solver by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1737>
- Removed legacy options/members `amioption.newton_preeq` and ``Solver::r...` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1744>
- Added `ReturnData::cpu_time_total` to track total time spent in `runAmiciSimulation` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1743>
- SBML import: Alternative algorithm for identifying conservation laws by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1748>
- Use `amici.AmiciVersionError` to indicate version mismatch by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1764>

**Performance:**

- Optional parallel computation of derivatives during model import by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1740>
- Sparsify jacobian by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1766>
- Speedup conservation law computation by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1754>
- Exploit stoichiometric matrix in pysb import by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1761>
- Speedup edata construction from petab problems by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1746>

**Fixes:**

- Fixed `get_model_settings` that would to setting incorrect initial states and initial state sensitivities for models with parameter-dependent initial states by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1751>
- Use correct tolerances for convergence check in Newton solver by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1728>
- Harmonized convergence checks by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1731>
- Made sundials' `KLU_INDEXTYPE` match actual klu index type by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1733>
- Fixed `Model::setStateIsNonNegative` logic that would raise exceptions in cases where it shouldn't by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1736>
- Fixed undefined reference to `dladdr` by @kristianmeyerr in <https://github.com/AMICI-dev/AMICI/pull/1738>
- Fixed HDF5 OSX intermediate group creation errors by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1741>
- Fixed recent cmake-based build issues due to changed sundials library directory by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1756>
- Updated Windows installation instructions by @paulflang in <https://github.com/AMICI-dev/AMICI/pull/1763>

... and other contributions by @FFroehlich, @dweindl

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.11.26...v0.11.27>

### 7.2.3 v0.11.26 (2022-03-14)

**New features:**

- Import of BioNetGenLanguage (BNGL) models by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1709>
- Added support for observable-dependent sigmas by @dweindl, @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1692>
- Added support for pysb local functions by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1666>
- Added experimental support for conservation laws for non-constant species to SBML import: conservation laws for non-constant species by @stephanmg, @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1669> Enable this feature by setting environment variable `AMICI_EXPERIMENTAL_SBML_NONCONST_CLS` to any value
  - Allow using states eliminated by conservation laws to be used in root functions by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1677>

- Added support for parameter-dependent conservation laws by @dweindl, @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1678>
- Added optional caching for symbolic simplifications in ODE export by @dilpath in <https://github.com/AMICI-dev/AMICI/pull/1672>
- Added CLI option `--no-sensitivities` to `amici_import_petab` by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1688>

Fixes:

- SBML import: Raise in case of nested observables by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1690>
- Sympy 1.10 compatibility by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1694>

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.11.25...v0.11.26>

## 7.2.4 v0.11.25 (2022-02-09)

- Fixed a bug where `Model::setStateIsNonNegative(Model::getStateIsNonNegative())` would raise an exception in case conservation laws were enabled - by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1648>
- Fixed a bug where `Model::setStateIsNonNegative` would be ignored in certain model expressions - by @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1650>
- Fixed a bug where special function parsing inside `min()` and `max()` would not be parsed correctly - by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1655>
- Fixed a numpy dependency issues for Mac+ARM systems - by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1657>
- Fixed convergence check in Newton method - by @plakrisenko in <https://github.com/AMICI-dev/AMICI/pull/1663>
- Add `AMICI_CXX_OPTIONS` to pass libamici-specific compiler options during CMake-based builds - by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1664>
- Fixed various warnings and updated documentation - by @dweindl

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.11.24...v0.11.25>

## 7.2.5 v0.11.24 (2022-02-01)

Features:

- Introduced environment variable `AMICI_DLL_DIRS` to control DLL directories on Windows (useful for setting BLAS library directory, as required by Python $\geq$ 3.8) by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1637>
- Dropped Python3.7 support by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1635>
- Include header files in CMake targets for better IDE integration by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1639>

Fixes:

- Fixed an issue in PEtab import where all-integer parameters would previously result in a `TypeError` by @stephanmg in <https://github.com/AMICI-dev/AMICI/pull/1634>

- Fixed tempdir deletion issues for test suite on Windows by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1636>
- Added functions to provide state IDs/names for x\_solver by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1638>
- Fixed docs on RTD by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1643>

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.11.23...v0.11.24>

## 7.2.6 v0.11.23 (2022-01-11)

Features:

- Added overload for Model::setParameterScale with vector by @dilpath in <https://github.com/AMICI-dev/AMICI/pull/1614>
- Removed assert\_fun argument from gradient checking, improve output by @dweindl, @FFroehlich in <https://github.com/AMICI-dev/AMICI/pull/1609>
- Added get\_expressions\_as\_dataframe by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1621>
- Added id field to ExpData and ReturnData by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1622>
- Included condition id in dataframes by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1623>

Fixes:

- C++: Fixed SUNMatrixWrapper ctor for size 0 matrices by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1608>
- Python: Handle TemporaryDirectory cleanup failures on Windows by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1617>
- Python: pysb.Model.initial\_conditions throws a DeprecationWarning by @PaulJonasJost in <https://github.com/AMICI-dev/AMICI/pull/1620>
- Fixed wrong array size in warnings by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1624>

NOTE: AMICI 0.11.23 requires numpy<1.22.0

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.11.22...v0.11.23>

## 7.2.7 v0.11.22 (2021-12-02)

- *\*Require sympy>=1.9,pysb>=1.13.1* by @FFroehlich, @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1599>
- Fixed sympy deprecation warning by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1600>
- Updated Windows installation instructions for Python>=3.8 by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1597>
- Fixed plot labels by @dweindl in <https://github.com/AMICI-dev/AMICI/pull/1598>

**Full Changelog:** <https://github.com/AMICI-dev/AMICI/compare/v0.11.21...v0.11.22>



### 7.2.8 v0.11.21 (2021-11-21)

Fixes:

- Fixed a bug in recursion depth computation for model expressions. This may have resulted in incorrect sensitivities for models with expressions nested more than 2 levels. (#1595)
- Fixed improper handling of Piecewise functions in PySB import which may have produced incorrect simulation results. (#1594)
- Fixed changed googletest reference which broke the CMake-based build if tests were enabled (#1592)

New:

- It's now possible to build AMICI using Ninja (#1593)

### 7.2.9 v0.11.20 (2021-11-12)

New:

- Changed parameter mappings such that unassigned values have non-nan default values. This fixes erroneous evaluation of `llh` as NaN in some situations (#1574)
- Added support for Python 3.10 (#1555)

Fixes:

- Fixed a bug when simulation start time was not transferred when copying a solver instance (#1573)
- Fixed a bug which led to incorrect sensitivities for models with multiple assignment rules or rate rules (#1584)

Other:

- Update CI and documentation settings (#1569, #1527, #1572, #1575, #1579, #1580, #1589, #1581)
- Extend set of validated benchmark models that is checked during CI (#1571, #1577)
- Fixed string formatting in derivative checks (#1585)
- Added helper methods to save and restore model instance-only settings (#1576)

### 7.2.10 v0.11.19 (2021-10-13)

New:

- Added support for observable transformations (lin/log/log10) (#1567). Thereby supporting additional noise distributions in combination with least squares solvers.

Fixes:

- Fixed a bug when Newton sensitivity computation was activated despite specifying `newton_steps == 0`. The error occurs when simulation converges to a steadystate but simulation sensitivities are not converged according to convergence criteria. In that case simulation returned failure, but the newton rootfinding “finds” a steadystate even before the iteration check, leading to the erroneous computation of sensitivities via Newton/IFT. For singular jacobians this means the overall simulation still fails, but a different, more informative error message is displayed. (#1541)
- Fixed a bug where argument “outdir” in `ODEExporter.init` would not be used (#1543)

Other:

- Improve checking support for SBML extensions (#1546)

- SBML import: Use more descriptive IDs for flux expressions (#1551)
- Optimized SUNMatrixWrapper functions (#1538)
- C++: Changed test suite from CppUTest to gtest (#1532)
- Add CITATION.cff (#1559)
- Updated documentation (#1563, #1554, #1536)
- Removed distutils dependency (#1557)
- Require sympy<1.9

### 7.2.11 v0.11.18 (2021-07-12)

New:

- Allow specifying maximum integration time via `amici::Solver::setMaxTime()` (#1530)
- Py: Add `failfast` and `num_threads` argument to `simulate_petab` (#1528, #1524)
- Enable typehints / static type checking for AMICI-generated model modules (#1514) (`amici.ModelModule`, available with Python>=3.8)

Fixes:

- Python: Fix unused argument `generate_sensitivity_code` in `pysb2amici` (#1526)
- Python: Fix C(++) stdout redirection which could have led to deadlocks in exotic situations (#1529)
- Py: Fixed deprecation warning (#1525)
- Py: Proper typehints for SWIG interface (#1523), enabling better static type checking and IDE integration (available with Python>=3.9)
- C++: Fixed clang compiler warning (#1521)
- C++: Fix inherited variadic ctor in exception class (#1520)
- PETab: More informative output for unhandled species overrides (#1519)
- Return `SbmlImporter` from PETab model import (#1517)

### 7.2.12 v0.11.17 (2021-05-30)

Fixes:

- Fix “maybe-uninitialized” compiler warning (#1495)
- Fix substitution of expressions in ``drootdt_total` (#1512)
- C++: Fix serialization and `==` operator (#1493)
- C++: Avoid `w` in `root` and `stau` headers (refactor) (#1503)

Documentation:

- Updated OpenBLAS Windows installation instructions (#1496)
- Updated how-to-cite to Bioinformatics paper (#1499)
- Updated list of papers using AMICI (#1509)

Other:

- Remove `sllh` computation from `petab_objective.simulate_petab` (#1498)
- Add `main.py` to Python package to provide info on AMICI installation via `python -m amici` (#1500)

### 7.2.13 v0.11.16 (2021-04-13)

Fixes:

- Fixed serialization bug (#1490)

New:

- Construction of condition specific plist for parameter mappings (#1487, #1488)
- **Add support for error residuals** (#1489)

### 7.2.14 v0.11.15 (2021-03-31)

Fixes:

- Fixed initial state sensitivities in adjoint preequilibration (#1468)
- Fixed various model import / parameter mapping issues (#1469, #1473, #1475)

New:

- New AMICI releases will automatically trigger releases at <https://biosimulators.org/simulators/amici/latest>
- Transparent logo

### 7.2.15 v0.11.14 (2021-03-16)

New features:

- **Python import now supports SBML Events** (#1443)
- Implement support for compilation without sensitivities (#1454)

Fixes:

- Issue #1446: Check whether constant parameters are valid targets (#1450)
- Issue #1422: Fix Steadystate solver failing if preequilibration starts in steadystate (#1461)
- Issue #1401: Ensure diagnostics variables in `ReturnData` are always of expected length (#1438, #1447)
- Fix FIM approximation for parameter dependent sigma (#1441)
- Fix invalid SBML in `PEtab/PySB` import (#1433)
- Fix: No context for `inspect.getouterframes` (#1432)

Documentation:

- Added this CHANGELOG
- Added feature request issue template (#1437)
- Updated reference list (#1430)
- Overhauled experimental conditions notebook (#1460)

CI:

- Test Matlab interface on GHA (#1451)
- Include componentTags in SBML test suite output (#1462)
- Split SBML semantic test suite into multiple jobs (#1452)
- Fix Crauste ref val, fixes #1458 (#1459)

Misc:

- Various cleanup (#1465, #1448, #1455)
- Micro-optimize SUNMatrixWrapper::transpose (#1439)
- Remove constant triggers from roots in Heaviside (#1440)

## 7.2.16 v0.11.13 (2021-02-20)

Breaking changes:

- AMICI requires Python $\geq$ 3.7
- Updated package installation (PEP517/518): Creating source distributions requires <https://github.com/pypa/build> (#1384) (but now handles all package building dependencies properly)

Features:

- More flexible state reinitialization (#1417)

Updated dependencies:

- Upgrade to sundials 5.7.0 (#1392)

Fixes:

- Python: account for heaviside functions in expressions (#1382)
- Python: allow loading of existing models in import\_petab\_problem (#1383)
- Python: Don't override user-provided compiler/linker flags (#1389)
- Python: PETab import reinitialization fixes (#1417)
- Python: Fix PETab observables for pysb models (#1390)
- Python: Substitute expressions in event condition expressions (#1404)
- Python: Unspecified initial states in PETab conditions table default to SBML initial value (#1397)
- C++: Fix timepoint out of bounds access (#1402)
- C++: Fix exported CMake config (#1388)
- Fixed Dockerfile: add python3-venv (#1398, #1408)

Other:

- Slim exported swig interface (#1425)
- Updated documentation
  - Getting started tutorial (#1423)
  - List supported SBML test tags (#1428)
  - Add AMICI C++/Python/Matlab feature comparison (#1409)
  - ...

- Various minor CI improvements
- ...

### 7.2.17 v0.11.12 (2021-01-26)

Features:

- Add expression IDs and names to generated models (#1374)

Fixes:

- Raise minimum sympy version to 1.7.1 (Closes #1367)
- Fix species assignment rules in reactions (#1372)
- Fix id vector for DAEs (#1376)

Docs:

- Update how-to-cite (#1378)

### 7.2.18 v0.11.11 (2020-12-15)

#### Python

- Restore support for species references (#1358)
- Add support for noise models in pysb (#1360)
- Proper handling of discontinuities in the ODE rhs (#1352)
- Fix directly calling AMICI from snakemake (#1348 )
- Extend mathml function support, particularly for numerical arguments (#1357)
- Restore support for sympy 1.6 (#1356)

#### C++

- Fix some compiler related warnings (#1349, #1362 )
- Fix a rare segfault for adjoint sensitivities (#1351)

#### CI

- Move windows tests to GHA (#1354)
- Pin breathe to 4.24.1

## Docker

- Update ubuntu to 20.04

## 7.2.19 v0.11.10 (2020-11-30)

Bugfix release that restores compatibility with sympy 1.7

## 7.2.20 v0.11.9 (2020-11-29)

### Python

- General improvements to SBML import (#1312, #1316, #1315, #1322 , #1324 #1333, #1329)
- Small bugfixes and improvements (#1321 )
- Improve derivative computation for instances of `power` (#1320 )

### C++

- Fix FIM and residual computation for models with parameter dependent sigma. (#1338)
- Disable chi2/residual/FIM computation for non-gaussian objective functions. (#1338)
- Bugfix for integration failure during adjoint solve (#1327)

### Doc

- Update references (#1331, #1336)

### CI

- Update OpenBLAS for windows (#1334)

## 7.2.21 v0.11.8 (2020-10-21)

### Python

- Fix pysb-petab support (#1288)
- Fix ExpData constructor overloading (#1299)
- Fix support for positivity enforcement (#1306)
- **Refactor SBML import, adds support for parameter rate rules and initial assignments** (#1284, #1296, #1304)
- Improve model generation for models with many parameters (#1300)
- Add support for PETab based synthetic data generation (#1283)

## C++

- Make HDF5 an optional dependency (#1285)

## Doc

- General Improvements to Documentation (#1289, #1291, #1292, #1293, #1294, #1286, #1277, #1281)

## CI

- Add python 3.9 support test (#1282)
- Allow manual triggering of GitHub actions (#1287)
- Remove appveyor config (#1295)
- Update GHA env and path management (#1302)

## 7.2.22 v0.11.7 (2020-09-22)

### Python

- Improve and extend available objective functions (#1235)
- Fix processing of compartment definitions (#1223)
- Fix replacement of reserved symbols (#1265)
- Use Hierarchical Derivatives for Expressions (#1224, #1246)
- Fix duplicate running of swig (#1216)
- Overload python interface functions for amici.{Model,Solver,ExpData} and amici.{Model,Solver,ExpData}Ptr (#1271)

## C++

- Fix and extend use of sparse matrix operations (#1230, #1240, #1244, #1247, #1271)
- **Fix application of maximal number of steps.** MaxNumStep parameter now limit total number of steps, not number of steps between output times. (#1267)

## Doc

- Move all Documentation to RTD (#1229, #1241)
- General Improvements to Documentation (#1225, #1227, #1219, #1228, #1232, #1233, #1234, #1237, #1238, #1239, #1243, #1253, #1255, #1262)

## CI

- Better check for doc building (#1226)
- Add more gradient checks (#1213)
- Update GHA to Ubuntu 20.04 (#1268)

## 7.2.23 v0.11.6 (2020-08-20)

### Python

- Bugfix for piecewise functions (#1199)
- Refactor swigging - generate one single wrapper (#1213)

### C++

- Fix warnings: account for zero indexing in nan/inf error (#1112)

### Doc

- Update Windows build instructions (#1200, #1202)
- Update README: Projects using AMICI (#1209)
- Add CODE\_OF\_CONDUCT.md (#1210)
- Update documentation for Python interface (#1208)

## CI

- Create sdist on GHA using swig4.0.1 (#1204) (Fixing broken pypi package)
- Fix links after repository move
- Speed-up swig build: disable all languages except python (#1211)
- Fix doc generation on readthedocs (#1196)

## 7.2.24 v0.11.5 (2020-08-07)

### General

- Move repo to new organization (#1193)
- Update Bibliography



### Python

- Fix bug for energyPySB models (#1191)

### CI

- Fix release deployment (#1189)

## 7.2.25 v0.11.4 (2020-08-06)

### Python

- Skip unnecessary expressions in pysb models (#1185)
- MSVC compiler support (this time for real... #1152)

### CI

- Implement MSVC tests (#1152)
- Rename and group GitHub actions (#1186)
- Fix release deployment (#1186)

## 7.2.26 v0.11.3 (2020-08-06)

### Python

- Fix simplification for pysb models (#1168)
- Pass verbosity flags to pysb network generation (#1173)
- Enable experimental pysb-petab support (#1175)
- Add installation instructions for Fedora (#1177)
- Implement support for SBML rate-references (#1180)

### C++

- Refactoring (#1162, #1163)

### CI

- Move majority of tests to Github Actions (#1166, #1160)
- Improve reporting of skipped tests in SBML testsuite (#1183)

## 7.2.27 v0.11.2 (2020-07-17)

### Python

- Speed up model import, compilation (#1123, #1112)
- Improve/Add steady-state solver documentation (#1102)
- Improve extension import (#1141)
- Bugfixes SBML import (#1135, #1134, #1145, #1154)
- Fixed issue that prevented simplification (#1158)

### C++

- Bugfixes (#1121, #1125, #1131, #1132, #1136)
- Enable openMP by default (#1118)
- Improve memory footprint for simulations with replicates (#1153)
- Improve steady-state solver and add option to to adjoint-steadystate hybrid (#1143, #1099, #1129, #1146)

### CI

- Store build artifacts from github actions (#1138)

## 7.2.28 v0.11.1 (2020-06-05)

### Python

- Upgrade to sympy 1.6.0, which is now required minimum version (#1098, #1103)
- Speed up model import
  - Speed-up computation of `sx0`, reduce file size (#1109)
  - Replace terribly slow `sympy.MutableDenseMatrix.is_zero_matrix` by custom implementation (#1104)
- speedup dataframe creation in `get*AsDataFrame` (#1088)
- Allow caching edatas for `simulate_petab` (#1106)
- Fix wrong deprecation warning (Fixes #1093)
- Fix segmentation faults in `NewtonSolver` under certain conditions (#1089, #1090, #1097)
- fix wrong power function call in `unscale_parameter` (#1094)
- Fix MathML conversion (#1086)
- Fix deepcopy of SymPy objects (#1091)

## Matlab

- handle empty `rdata->{pre|post}eq_numlinsteps` (Closes #1113), which previously made the matlab interface unusable
- Fix generation of `compileMexFile.m` for matlab compilation of python code (#1115)

## C++

- Reduce memory requirements and speedup compilation of large models (#1105)
- Place generated code into own namespace (#937) (#1112)
- Fix several msvc compiler warnings (#1116) (Note that MSVC support is still experimental) **breaking change for users of C++ interface**
- Fix swig warning: ensure base class `ContextManager` is known before use (Fixes #1092) (#1101)

## CI

- Don't install/run valgrind on travis CI (done with github actions... (#1111)

## 7.2.29 v0.11.0 (2020-05-10)

Python:

- **Implement support for variable compartments (#1036)**
- Better handling of constant species (#1047)
- **Better handling of C++ enums, this makes `amici.SensitivityMethod_forward` available as `amici.SensitivityMethod.forward` (#1042)**
- Improve installation routines (#1055, #1056, #1058, #1076)
- Add option to reduce memory usage (#1044)
- **Fix handling of symbolic expressions in nested rules (#1081, 1069)**

Library:

- Update Sundials to 5.2.0 (#1039)
- Update SuiteSparse to 5.4.0 (#1040)
- Refactor use of `ReturnData`, now completely created post-hoc (#1002)
- **Fix propagation of reinitialization in `ExpData` constructor (#1041)**
- **Fix issue where `InternalSensitivityParameter` was sometimes not set (#1075)**
- **Fix or disable certain combinations of equilibration, presimulation and adjoint sensitivity analysis**

CI:

- Move from Codacy to Sonarcloud (#1065)
- Run SBML Testsuite when appropriate (#1058)

### 7.2.30 v0.10.21 (2020-04-04)

Library:

- Fix: Handle paths with blanks in build scripts
- Feature: Add function to write amici::Solver settings to HDF5 (#1023)
- Fix: typehints (#1018, #1022)
- Refactor: Move creation of parameter mapping for objective<->simulation to classes (#1020)

CI:

- Refactor: Cleanup and reorganize tests (#1026)
- Fix: benchmark problem test should fail on missing files (Closes #1015)

### 7.2.31 v0.10.20 (2020-03-18)

- Fixed (re)initialization of sensitivities if ExpData::fixedParametersPreequilibration is set (#994)
- Fixed sensitivities for parameters in sigma expressions for Python/SBML in case provided expression was not just a single parameter ID
- Enable parallel compilation of model files from Python (#997) based on AMICI\_PARALLEL\_COMPILE environment variable
- Fixed computation of log-likelihood for log10-normal distributed noise
- Added `reinitializeFixedParameterInitialStates` to ExpData (#1000) (**breaking change**: overrides settings in `amici::Model`)
- Python model import now verifies that chosen model name is a valid identifier (Closes #928)
- Made `w` available in ReturnData (Closes #990) (#992)
- Fixed setting of log level when passing boolean values to verbose (#991)
- Documentation now on ReadTheDocs <https://amici.readthedocs.io/en/>
- Use proper state/observable names in plotting functions (#979)
- PETab support:
  - Adapt to most recent PETab (0.1.5)
  - Extended support for import of PETab models
  - Added support for computing cost function based on PETab problem
  - Implemented handling of species in condition table
  - `petab_import.import_model` now provides reproducible parameter list (Closes #976)
  - Fix python import error in `import_petab_problem`: Add absolute paths to python path, invalidate caches and reload (#970)
  - Added example notebook
- CI: PETab test suite integrated in CI workflow
- Added AMICI dockerfile and image deployment to dockerhub (#948)
- Removed mention of ‘mex’ in warning/error ids (#968)
- More informative errors on SWIG interface import failures (#959)

### 7.2.32 v0.10.19 (2020-02-13)

Python:

- Fix logo display on pypi
- Fix deadlocks in multithreaded python environments when using openMP parallelization

Matlab:

- Fix compilation errors due to switch to C++14

### 7.2.33 v0.10.18 (2020-02-11)

General:

- AMICI now comes with a logo
- implement getName function for models
- Updated documentation / examples

Python:

- Enable MSVC compilation of Python extensions (#847)
- Always recompile clibs and extensions (Closes #700)
- Extended PETab support (Running
- enable multithreading in swig (#938)
- Fixes pysb (#902) (#907)

C++

- Build optimized AMICI and sundials by default (Closes #934)

Matlab:

- Fix(matlab) Compile CalcMD5 on demand (Fixes #914)
- Don't pass empty include strings to mex
- Fix Matlab compilation error if AMICI or model path contains blanks

CI:

- Running additional test models

... and various minor fixes/updates

### 7.2.34 v0.10.17 (2020-01-15)

- **added python 3.8 support, dropped python 3.6 support** (#898)
- Added logging functionality (#900)
- Fixes PySB import (#879, #902)
- Fixes symbolic processing (#899)
- Improved build scripts (#894,
- Improved petab support (#886, #888, #891)

- CI related fixes (#865, #896)

### 7.2.35 v0.10.16 (2019-12-11)

- **Sparsify dwdp to reduce computation time for adjoints (#858)**
- Fix(matlab) update example name example\_dae\_events->example\_calvetti (Closes #866)
- Fix nullptr deferencing for simulations with events when no measurements are provided (Fixes #866)
- Fix accessing empty vector during adjoint state event update (Closes #866)
- Fix pysb\_import (fixes #878)

### 7.2.36 v0.10.15 (2019-12-03)

Bugfix release due to incorrect sensitivities w.r.t. sigmas introduced in 0.10.14.

No other changes.

### 7.2.37 v0.10.14 (2019-12-02)

**NOTE: For Python-imported SBML-models this release may compute incorrect sensitivities w.r.t. sigma. Bug introduced in 0.10.14, fixed in 0.10.15.**

Python:

- Don't require use of ModelPtr.get to call ExpData(Model)
- Fix import in generated model Python package
- Setup AMICI standalone scripts as setuptools entrypoints
- Simplify symbolic sensitivity expressions during Python SBML import Fixes Infs in the Jacobian when using Hill-functions with states of 0.0.
- Extended Newton solver #848 The changes that allow performing Newton tests from the paper: G. T. Lines, L. Paszkowski, L. Schmiester, D. Weindl, P. Stapor, and J. Hasenauer. Efficient computation of steady states in large-scale ODE models of biochemical reaction networks. accepted for Proceedings of the 8th IFAC Conference on Foundations of Systems Biology in Engineering (FOSBE), Valencia, Spain, October 2019.
- Use SWIG>=4.0 on travis to include PyDoc in sdist / pypi package (#841)
- **Fix choice of likelihood formula; failed if observable names were not equal to observable IDs**
- Fix(sbml-import) Compartment IDs in right-hand side of Rules are not replaced and lead to undefined identifiers in c++ files
- Fix invalid logging level
- Speed up sympy simplification (#871)

C++:

- Performance: Avoid unnecessary repeated function calls for SUNMatrixWrapper dimensions
- Add AmiciApplication class as context for handling so far global settings. This allows for example setting custom logging functions for concurrent AMICI runs, e.g. in multi-thread applications (Closes #576).

Misc:

- Setup performance test on github actions (#853)

- Update documentation and FAQ for CBLAS requirement and others
- Update reference list

### 7.2.38 v0.10.13 (2019-10-09)

- BREAKING CHANGE: Renaming {get|set}tNewtonPreequilibration to {get|set}Preequilibration (Closes #720)
- Make wurlitzer non-optional requirement for AMICI python package (Fixes missing AMICI errors when running from jupyter notebooks)
- Compute initial state for Model::getInitialStates if not already set (Fixes #818)
- Make swig generate pydoc comments from doxygen comments #830 (Closes #745) to provide Python docstrings for C++ wrapper functions
- feature(cmake) Add option to disable compiler optimizations for wrapfunctions.cpp (Fixes #828) (#829)
- Change SBML test suite to pytest to allow for parallel test execution... (#824)
- Fix(cmake): -E option is not available in all sed versions. Neither is the equivalent -r. Use --regexp-extended instead (Closes #826)
- Refactor(python) Move PETab import code from command line script... (#825)
- Fix(core) Fix regular expressions for intel compiler (Closes #754) (#822)
- Update workflow figure to include PySB (Closes #799)
- Fix compiler warnings

### 7.2.39 v0.10.12 (2019-09-28)

- Fix handling of species specified in PETab condition table (#813)
- Fix some Visual C++ issues, update cppcheck handling, cleanup (VisualC++ still not fully supported)
- Minor fixups (#801)
- Create SBML test suite result files for upload to <http://sbml.org/Facilities/Database/> (#798)

### 7.2.40 v0.10.11 (2019-08-31)

- Fixed setting initial conditions for preequilibration (#784)
- Fixed species->parameter conversion during PETab import (#782)
- Set correct Matlab include directories in CMake (#793)
- Extended and updated documentation (#785, #787)
- Fix various SBML import issues
- Run SBML test suite using github actions instead of travisCI (#789)

### 7.2.41 v0.10.10 (2019-08-07)

- Simplify/fix AMICI installation
  - If available use environment modules to detect dependencies
  - Add SWIG installation script
- Update list of publication
- Update documentation
  - Update doc for SWIG build and custom SWIG location.
  - Add AMICI interface overview / workflow figure and show in README
  - Document environment variables for model/core compilation (Closes #737)
- Added handling of abs function, since there seem to be problems with case sensitivity (#713) Closes #770

Details: \* cmake: Use package\_ROOT environment variables \* fix(cmake) Fix finding version.txt \* cmake: Auto-detect loaded MKL environment module \* cmake: Use new FindPython3 modules where possible \* fix(python) Restore python3.6 compatibility \* Inside venv, use pip instead of pip3 which should point to the correct version \* fix(python) Workaround for missing ensurepip during venv creation [ci skip] \* feature(python) Use MKL from environment modules to provide cblas \* fix(python) Fix define\_macros not being passed to setuptools for Extension \* fix(python) Fix define\_macros not being passed to setuptools for clibs \* Do not always add 'cblas' library since users may want to override that by a cblas-compatible library with a different name (closes #736) \* Update HDF5 path hints; use shared library if static is not available. \* Check for HDF5\_BASE from environment module \* Fix system-dependent sundials library directory (Fixes #749) (#750) \* Handle OSTYPE==linux in scripts/buildBNGL.sh (Fixes #751) \* Add SWIG download and build script \* Improve finding swig executable and allow user override via SWIG environment variable \* Provide installation hints if no SWIG found (Closes #724) \* Allow overriding cmake executable with environment variables in build scripts (Closes #738)

### 7.2.42 v0.10.9 (2019-07-24)

Fixup for missing version bump in v0.10.8 release. No code changes compared to v0.10.8.

### 7.2.43 v0.10.8 (2019-07-24)

Changes in this release:

All:

- Updated / extended documentation
- Fix reuse of Solver instances (#541)

C++:

- Check for correct AMICI version for model in CMake
- Add reporting of computation times (#699)

Python:

- Fix manifest file (#698)
- Fix initial amounts/concentrations in SBML import

... and various other minor fixes/improvements



### 7.2.44 v0.10.7 (2019-05-01)

Python

- fix unset noise distribution when automatically generating observables in case None are passed (#691)

### 7.2.45 v0.10.6 (2019-04-19)

C++

- Add SuperLUMT support (#681)
- Sparsified dJydy (#686)
- Enabled support of impulse-free events for DAE models (#687) - thanks to Sebastien Sten for providing a testcase for this

Python

- Enabled support for piecewise functions in SBML import (#662)
- Fix numeric type when constructing ExpData from Dataframes (#690)
- Fix dynamic override in PETab

### 7.2.46 v0.10.5 (2019-04-08)

Bugfix release

Doc

- Update documentation of Windows installation

C++

- Fix missing source files in CMakeLists.txt (#658)
- Set CMake policies to prevent warnings (Closes #676) (#677)
- Start using gsl::span instead of raw pointers (#393) (#678)

Python

- PySB parsing fix (#669)
- Fix failure to propagate BLAS\_LIBS contents (#665)
- Require setuptools at setup (#673)
- Updated PETab import to allow for different noise models

### 7.2.47 v0.10.4 (2019-03-21)

Features / improvements:

- Implement ReturnData and ExpData wrappers as more efficient views (#657)
- Add list of references using AMICI (#659)
- Custom llh (normal/laplace, lin/log/log10) (#656)

Bugfixes:

- Speedup and fix travis build

### 7.2.48 v0.10.3 (2019-03-13)

Features / improvements:

- adds the option for early termination on integration failures for `runAmiciSimulations`
- improve runtime of `SUNMatrixWrapper::multiply`
- expose finite difference routines in public API
- enable parallel compilation of clib source files

Bugfixes:

- fixed symbolic processing for unreleased pysb features

### 7.2.49 v0.10.2 (2019-03-07)

Features / improvements:

- extended `ExpData` interface to allow for condition specific parameters, parameter scales, parameter lists, initial conditions and initial condition sensitivities.

Bugfixes:

- fixed output values of `ReturnData::x_ss` and `ReturnData::sx_ss`

### 7.2.50 v0.10.1 (2019-03-04)

- travis-ci.com migration
- fix problem with `has{variable}` functions
- allow to import sbml model from string, not only file

### 7.2.51 v0.10.0 (2019-03-01)

Features / improvements:

- updated sundials to 4.1.0
- updated SuiteSparse to 5.4.0
- added generic implementations for symbolic expressions that were sparse matrix vector products

Bugfixes:

- fixed return value of `rz` when no data is provided.

### 7.2.52 v0.9.5 (2019-02-26)

Features / improvements:

- allow python installations without compilation of c++ extension
- improvements to ExpData <-> pandas.DataFrame interface
- allow generation of matlab models from python
- implement CLI interface for PEstab
- improve computation time for conservation laws from pysb import

Bugfixes:

- Fix sign in undamped Newton step.

Maintenance:

- use newer CI images

### 7.2.53 v0.9.4 (2019-02-11)

Minor fixes only:

- fix(core) Get solver diagnostics for first(last) timepoint (#588) (Closes #586)
- fix(ci) Fix autodeploy (Closes #589)

### 7.2.54 v0.9.3 (2019-02-07)

#### CRITICAL FIXES

- **fix(python) fix symbolic computations for adjoint (#583)**

#### Features

- feature(python) Check for matching AMICI versions when importing model (Closes #556). Set exact AMICI version as model package requirement.
- feature(core) Add option to rethrow AMICI exception (Closes #552)
- feature(python) Redirect C/C++ output in stdout is redirected (e.g. in ipython notebooks) (Closes #456)

#### Minor fixes

- fix(python) Fix doc and rename sys\_pipes to something more meaningful
- fix(ci) Fix premature exit of scripts/runNotebook.sh
- fix(deploy) Update pyenv shims to find twine

### 7.2.55 v0.9.2 (2019-01-30)

Bugfixes:

- fixes a critical bug in the newton solver
- fixes multiple bugs in sbml import for degenerate models, empty stoichiometry assignments and conversion factors
- improved error messages for sbml import
- #560
- #557
- #559

### 7.2.56 v0.9.1 (2019-01-21)

Features / improvements:

- make pure steadystate results available as `rdata['x_ss']` and `rdata['sx_ss']`
- add option to specify integration tolerances for the adjoint problem via `atolB` and `rtolB`

Bugfixes:

- improved conservation law identification to also account for constant species.
- fixed a bug where simulation results were written into results of the second newton solver attempt
- fixed an openMP related warning

Maintenance:

- attempt to fix automatic deploy to pypi via travis

### 7.2.57 v0.9.0 (2019-01-18)

Features / improvements:

- Allow computation and application of conservation laws for pysb importet models. This enables use of Newton-Solver for preequilibration for models where it was previously not possible.
- Use `klu_refactor` in the sparse Newton solver instead of always using `klu_factor` and only perform symbolic factorization once (#421)
- Allow more detailed finiteness checks (#514)

Bugfixes:

- #491

Maintenance:

- Several improvements to travis log sizes and folding
- Use default copy constructor for Model by implementing class wrappers for sundials matrix types (#516)
- Reenable run of SBML testsuite

### 7.2.58 v0.8.2 (2019-01-07)

Features / improvements:

- Speedup symbolic processing for ODE generation in python

Bugfixes:

- Fix(python) Add missing deepcopy (introduced in 6847ba675f2088854db583199b8754aaa6e01576)
- Fix(core) Prevent parameter scaling length mismatch (#488)
- Fix(python) Set distutils dependency to current version to fix `</usr/lib/python3.6/distutils/dist.py:261: UserWarning: Unknown distribution option: 'long_description_content_type'>`
- fix(python) add symlink to version.txt to be included in package

Backwards-compatibility:

- Replace 'newline' by literal to restore <R2016b compatibility (Fixes #493)

Maintenance:

- Remove obsolete swig library build via cmake and related file copying
- Provide issue template for bug reports
- Providing valid SBML document to import is not optional anymore
- Update documentation and tests
- Add python version check and raise required version to 3.6 to prevent cryptic error messages when encountering f-strings

### 7.2.59 v0.8.1 (2018-11-25)

- [all] **critical** Fix long standing bugs in solving steadystate problems (including preequilibration) (#471)
- [all] Fix AmiVector constructor from `std::vector` (#471)
- [python] Reenable Solver and Model copy constructors
- Update documentation

### 7.2.60 v0.8.0 (2018-11-25)

- replaced symengine by sympy for symbolic processing in python *which fixes several critical bugs* that were due to bugs in symengine (#467)

### 7.2.61 v0.7.13 (2018-11-18)

- fixes a critical bug in objective function computation for models compiled using `sbml2amici` and `pysb2amici` that was introduced in v0.7.12
- fixes a critical bug in sensitivity computation when `model.reinitializeFixedParameterInitialStates` was set to true
- readds the python interface to the `ExpData` copy constructor that was inadvertently removed in 0.7.12 and streamlines the respective convenience wrapper to provide access to the full range of constructors.

### 7.2.62 v0.7.12 (2018-11-17)

- fixed a critical bug in `amici.constructEdataFromDataFrame`
- enabled multithreaded simulation of multiple experiments (requires compilation with openMP)
- modularized sbml import and added pysb import

### 7.2.63 v0.7.11 (2018-10-15)

- [python] Added numpy and python wrappers that provide a more user friendly python API
- [python] Enable import of SBML models with non-float assignment rules
- [python] Enable handling of exceptions in python
- [python] Enable native python access to `std::vector` data-structures
- [core] Provide an API for more fine-grained control over sensitivity tolerances and steady-state tolerances
- [core] Provide an API to specify non-negative state variables (this feature is still preliminary)

### 7.2.64 v0.7.10 (2018-08-29)

- Fixed python SBML import `log()` issues (#412)

### 7.2.65 v0.7.9 (2018-08-24)

- fixes MATLAB compilation of models
- adds option to perform steady state sensitivity analysis via FSA
- condition dependent initial conditions are now newly set after preequilibration is done

### 7.2.66 v0.7.8 (2018-08-19)

- bugfixes for the ExpData interface
- created build configuration that enables debugging of c++ extensions on os x
- fixed python sbml import when stoichiometry is empty

### 7.2.67 v0.7.7 (2018-08-17)

Fixes a couple of bugs just introduced in v0.7.6

### 7.2.68 v0.7.6 (2018-08-13)

Important: Use AMICI v0.7.7 due to <https://github.com/ICB-DCM/AMICI/pull/403/commits/3a495d3db2fdbba70c2b0d52a3d465>

Bug fixes:

- Python import: Fix log10 issues in observables (#382)
- Matlab: Fix broken model compilation (#392)
- Fixed simulation for models without observables (#390)
- Fixed potential matlab memory leaks (#392)

Breaking C++ API changes:

- Revised ExpData interface (#388)

### 7.2.69 v0.7.5 (2018-07-30)

Features/enhancements:

- Add computation of residuals, residuals sensitivity, Fisher information matrix (#223)
- More efficient conversion of std::vector to numpy ndarray (#375)
- Allow specifying timepoints in ExpData (#370)

Minor fixes:

- Condition parameters in ExpData now only temporarily override Model parameters (#371)
- Ensure non-negative states for Newton solver (#378)

### 7.2.70 v0.7.4 (2018-07-27)

Features/enhancements:

- Check SBML model validity (#343)
- Allow per-parameter setting of amioptions::pscale from matlab interface (#350)
- Documentation

Major fixes:

- Don't compile main.cpp into python model module which broke modules if amici was compiled without libhdf5 (#363)

Minor fixes:

- Fix compiler warnings (#353)
- Plotting, SBML example mode, ...

### 7.2.71 v0.7.3 (2018-07-13)

Features:

- Added symbol names to python-wrapped models and make available via `Model.getParameterNames()`, `model.getStateNames()`, ...
- Extended Python interface example

Python package available via pypi: <https://pypi.org/project/amici/0.7.3/>

### 7.2.72 v0.7.2 (2018-07-03)

Features:

- Python package: more flexible HDF5 library localization
- Extended CI: python tests, preequilibration tests, run in venv

Major bugfixes:

- Fix python sbml model import / compilation error (undefined function)
- Fix model preequilibration

Minor fixes:

- Various fixes for mingw compilation of python source distribution
- Cmake compatibility with < 3.8 restored

### 7.2.73 v0.7.1 (2018-06-12)

Features:

- Allow specifying sigma-parameters from Python interface

Major bugfixes:

- Fix `dsigma_y/dp` and downstream sensitivity errors

### 7.2.74 v0.7.0 (2018-06-09)

- Major revision of documentation
- Improved Python interface
- More comprehensive Python interface example
- Fixed sensitivity computation in Python-generated models
- Various other bug fixes

WARNING:

- For models with sigma-parameters and `dsigma_y/dp != 0`, `dsigma_y/dp` was computed incorrectly. This propagates to all dependent sensitivities. This applies also to some older releases and has been fixed in v0.7.1.



### 7.2.75 v0.6.0 (2018-05-22)

Implement experimental support for python via swig. Python interface is now usable, but API will still receive some updates in the future.

WARNING:

- There is a bug in sensitivity computation for Python-generated models
- Matlab C++ compilation will fail due to undefined M\_PI -> Please use v0.7.0

### 7.2.76 v0.5.0 (2018-03-15)

Main new features are:

- Reimplemented support for DAE equations
- Added newton solver for steady state calculation and preequilibration
- Better caching for recompilation of models
- Blas support to allow compilation of large models with many observables
- Improved SBML support
- Added c++ interface
- Added support for second order adjoint computation
- Rewrote large parts of the code as proper c++11 code to allow easier code maintenance
- Substantially extended testing in continuous integration to assure code quality

### 7.2.77 v0.4.0 (2017-05-15)

- First citable version of AMICI (via zenodo integration).
- Better support for standalone compilation
- Improved SBML import scripts
- General Code Cleanup

### 7.2.78 v0.3.0 (2016-09-05)

This update comes with many improvements, bug fixes and several new features. Most notably:

1. AMICI should now run on older versions of MATLAB (back to R2014a)
2. AMICI now compiles using older versions of Visual Studio
3. AMICI now also supports second order adjoint sensitivities (full (via the o2flag = 1 and as a vector product via o2flag = 2)
4. AMICI now supports more SBML, SBML v2 and rate rules

### 7.2.79 0.2.1 (2016-05-09)

Bugfix release. This release also includes some changes that should improve the performance on the new R2016a release of MATLAB.

### 7.2.80 v0.2 (2016-03-17)

This update comes with many improvements to the computation time for both compilation and simulation. Moreover several new features were included:

1. Hessian Vector products for second order forward sensitivities
2. Correct treatment of parameter/state dependent discontinuities for both forward and adjoint sensitivities

### 7.2.81 v0.1 (2015-11-05)

This is the initial release of the toolbox

## GLOSSARY

**BNGL** **BioNetGenLanguage** is a language for modular, structure-based modeling of biochemical reaction networks.

**CVODES** **CVODES** is a solver for stiff and non-stiff *ODE* systems with sensitivity analysis capabilities and is used by AMICI. It is part of the *SUNDIALS* solver suite.

**DAE** Differential-Algebraic Equation

**fixed parameters** In AMICI, *fixed parameters* are parameters with respect to which no sensitivities are computed. They usually correspond to experimental conditions. For fixed parameters, different values can be set for *preequilibration*, *presimulation* and simulation.

**IDAS** **IDAS** is a solver *DAE* systems with sensitivity analysis capabilities and is used by AMICI. It is part of the *SUNDIALS* solver suite.

**ODE** Ordinary Differential Equation

**PEtab** **PEtab** is a format for specifying parameter estimation problems. It is based on an *SBML* model and tab-separated value files specifying the observation model and experimental conditions.

**preequilibration** Simulating or solving the dynamical system for the steadystate.

**presimulation** Simulation for a fixed time before the regular simulation. Can be used to specify pretreatments.

**PySB** **PySB** is a tool for specifying rule-based systems biology models as Python code.

**SBML** The **Systems Biology Markup Language** is a commonly used format for specifying systems biology models.

**SUNDIALS** **SUNDIALS**: SUite of Nonlinear and Differential/ALgebraic equation Solvers. Provides the *CVODES* and *IDAS* solvers used by AMICI.

**SWIG** **SWIG** is a tool that creates interfaces for C(++) code to a variety of languages. Much of the AMICI Python interface is generated by SWIG.



## CONTRIBUTING

### 9.1 Contributing to AMICI

We are happy about contributions to AMICI in any form, be it new functionality, documentation, bug reports, or anything else.

If you would to contribute to AMICI, a good start is looking for issues tagged `good first issue` or `help wanted`. For other ideas or questions, just post an issue.

For code contributions, please read our [developer's guide](#) first.

### 9.2 Code of Conduct

#### 9.2.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

#### 9.2.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

### 9.2.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

### 9.2.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

### 9.2.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at [froehlichfab@gmail.com](mailto:froehlichfab@gmail.com). All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

### 9.2.6 Attribution

This Code of Conduct is adapted from the [Contributor Covenant](https://www.contributor-covenant.org/version/1/4/code-of-conduct.html), version 1.4, available at <https://www.contributor-covenant.org/version/1/4/code-of-conduct.html>

For answers to common questions about this code of conduct, see <https://www.contributor-covenant.org/faq>

## PYTHON INTERFACE

### 10.1 Installing the AMICI Python package

#### 10.1.1 Short guide

Installation of the AMICI Python package has the following prerequisites:

- Python $\geq 3.9$
- *SWIG* $\geq 3.0$
- CBLAS compatible BLAS library (e.g., OpenBLAS, CBLAS, Atlas, Accelerate, Intel MKL)
- a C++17 compatible C++ compiler and a C compiler (e.g., g++ $\geq 9.1$ , clang $\geq 12$ , Intel C++ compiler, mingw)

If these requirements are fulfilled and all relevant paths are setup properly, AMICI can be installed using:

```
pip3 install amici
```

If this worked, you can now import the Python module via:

```
import amici
```

If this does not work for you, please follow the full instructions below.

#### 10.1.2 Installation on Linux

##### Ubuntu 22.04

Install the AMICI dependencies via apt (this requires superuser privileges):

```
sudo apt install libatlas-base-dev swig

# optionally for HDF5 support:
sudo apt install libhdf5-serial-dev

# optionally for boost support (thread-specific CPU times, extended math functions,
↳ serialization)
libboost-chrono-dev libboost-math-dev libboost-serialization-dev
```

Install AMICI:

```
pip3 install amici
```

## Fedora 32

Install the AMICI dependencies via apt (this requires superuser privileges):

```
sudo dnf install blas-devel swig
```

Install AMICI:

```
pip3 install amici
```

## Arch Linux

Install the AMICI dependencies via pacman (this requires superuser privileges):

```
sudo pacman -S python swig openblas gcc hdf5 boost-libs
```

Install AMICI:

```
pip3 install amici
```

Alternatively:

1. Check if packages are already installed with the required versions for AMICI installation.

```
sudo pacman -Si python swig openblas gcc hdf5 boost-libs
```

2. Upgrade installed packages if required minimum versions are not satisfied for AMICI installation.

```
sudo pacman -Su python swig openblas gcc hdf5 boost-libs
```

3. Install AMICI:

```
pip3 install amici
```

## 10.1.3 Installation on OSX

Install the AMICI dependencies using homebrew:

```
brew install swig

# optionally for HDF5 support:
brew install hdf5

# optionally for parallel simulations:
brew install libomp
# followed by either `brew link openmp` once,
# or `export OpenMP_ROOT=$(brew --prefix)/opt/libomp`` where `OpenMP_ROOT` will have to
→ be set during every re-installation of AMICI or any new model import
```

(continues on next page)



(continued from previous page)

```
# optionally for boost support (thread-specific CPU times, extended math functions,
↳ serialization)
brew install boost && export BOOST_ROOT=$(brew --prefix)/opt/boost
# followed by either `brew link boost` once,
# or `export BOOST_ROOT=$(brew --prefix)/opt/boost` where `BOOST_ROOT` will have to be
↳ set during every re-installation of AMICI or any new model import
```

Install AMICI:

```
pip3 install amici
```

## 10.1.4 Installation on Windows

Some general remarks:

- Consider using the [Windows Subsystem for Linux \(WSL\)](#) and follow the instructions for installation on linux.
- Install all libraries in a path not containing white spaces, e.g. directly under C:.
- Replace the following paths according to your installation.
- Slashes can be preferable to backslashes for some environment variables.
- See also [#425](<https://github.com/AMICI-dev/amici/issues/425>) for further discussion.

### Using the Microsoft Visual Studio

We assume that Visual Studio (not to be confused with Visual Studio Code) is already installed. Using Visual Studio Installer, the following components need to be included:

- Microsoft Visual C++ (MSVC). This is part of multiple packages, including Desktop Development with C++.
- Windows Universal C Runtime. This is an individual component and installs some DLLs that we need.

### OpenBLAS

There are prebuilt OpenBLAS binaries available, but they did not seem to work well here. Therefore, we recommend building OpenBLAS from scratch. This requires an installation of CMake. CMake can be installed from <https://cmake.org/download/> (system-wide), or via `pip install cmake` (in the current Python environment).

To build OpenBLAS, download the following scripts from the AMICI repository:

- <https://github.com/AMICI-dev/AMICI/blob/master/scripts/installOpenBLAS.ps1>
- <https://github.com/AMICI-dev/AMICI/blob/master/scripts/compileBLAS.cmd>

The first script needs to be called in Powershell, and it needs to call `compileBLAS.cmd`, so you will need to modify line 11:

```
cmd /c "scriptscompileBLAS.cmd $version"
```

Additionally, in `compileBLAS.cmd` make sure that you point to your Visual Studio installation on line 3. Newer installations could be located under `C:\Program Files\Microsoft Visual Studio\...\VC\Auxiliary\Build\vcvars64.bat`.

so that it matches your directory structure. This will download OpenBLAS and compile it, creating

C:\BLAS\OpenBLAS\lib\openblas.lib C:\BLAS\OpenBLAS\bin\openblas.dll

You will also need to define two environment variables:

```
BLAS_LIBS="-LIBPATH:C:/BLAS/OpenBLAS/lib openblas.lib"
BLAS_CFLAGS="-IC:/BLAS/OpenBLAS"
```

One way to do that is to run a PowerShell script with the following commands:

```
[System.Environment]::SetEnvironmentVariable("BLAS_LIBS", "-LIBPATH:C:/BLAS/OpenBLAS/lib\
↪openblas.lib", [System.EnvironmentVariableTarget]::User)
[System.Environment]::SetEnvironmentVariable("BLAS_LIBS", "-LIBPATH:C:/BLAS/OpenBLAS/lib\
↪openblas.lib", [System.EnvironmentVariableTarget]::Process)
[System.Environment]::SetEnvironmentVariable("BLAS_CFLAGS", "-IC:/BLAS/OpenBLAS/include/
↪openblas", [System.EnvironmentVariableTarget]::User)
[System.Environment]::SetEnvironmentVariable("BLAS_CFLAGS", "-IC:/BLAS/OpenBLAS/include/
↪openblas", [System.EnvironmentVariableTarget]::Process)
```

The call ending in Process sets the environment variable in the current process, and it is no longer in effect in the next process. The call ending in User is permanent, and takes effect the next time the user logs on.

Now you need to make sure that all required DLLs are within the scope of the PATH variable. In particular, the following directories need to be included in PATH:

C:\BLAS\OpenBLAS\bin C:\Program Files (x86)\Windows Kits\10\Redist\ucrt\DLLs\x64

The first one is needed for openblas.dll and the second is needed for the Windows Universal C Runtime.

If any DLLs are missing in the PATH variable, Python will return the following error upon `import amici`:

ImportError: DLL load failed: The specified module could not be found.

Almost all of the DLLs are standard Windows DLLs and should be included in either Windows or Visual Studio. But, in case it is necessary to test this, here is a list of some DLLs required by AMICI (when compiled with MSVC):

- openblas.dll
- python37.dll
- MSVCP140.dll
- KERNEL32.dll
- VCRUNTIME140\_1.dll
- VCRUNTIME140.dll
- api-ms-win-crt-convert-l1-1-0.dll
- api-ms-win-crt-heap-l1-1-0.dll
- api-ms-win-crt-stdio-l1-1-0.dll
- api-ms-win-crt-string-l1-1-0.dll
- api-ms-win-crt-runtime-l1-1-0.dll
- api-ms-win-crt-time-l1-1-0.dll
- api-ms-win-crt-math-l1-1-0.dll

MSVCP140.dll, VCRUNTIME140.dll, and VCRUNTIME140\_1.dll are needed by MSVC (see Visual Studio above). KERNEL32.dll is part of Windows and in C:\Windows\System32. The api-ms-win-crt-XXX-l1-1-0.dll are needed by openblas.dll and are part of the Windows Universal C Runtime.

**Note:** Since Python 3.8, the library directory needs to be set either from Python:

```
import os
# directory containing `openblas.dll`
os.add_dll_directory("C:\\BLAS\\OpenBLAS\\bin")
import amici
```

or via the environment variable `AMICI_DLL_DIRS="C:\\BLAS\\OpenBLAS\\bin"`.

---

## 10.1.5 Further topics

### Installation of development versions

To install development versions which have not been released to PyPI yet, you can install AMICI with pip directly from GitHub using:

```
pip3 install -e git+https://github.com/AMICI-dev/amici.git@develop#egg=amici\&
↳ subdirectory=python/sdist
```

Replace `develop` by the branch or commit you want to install.

Note that this will only work on Windows if you have enabled developer mode, because symlinks are not supported by default ([more information](#)).

### Light installation

In case you only want to use the AMICI Python package for generating model code for use with Matlab or C++ and don't want to be bothered with any unnecessary dependencies, you can run

```
pip3 install --install-option --no-libs amici
```

---

**Note:** Following this installation, you will not be able to simulate the imported models in Python.

---

---

**Note:** If you run into an error with above installation command, install all AMICI dependencies listed in [setup.py](#) manually, and try again. (This is because `pip --install-option` is applied to *all* installed packages, including dependencies.)

---

### Custom installation

Installation of the AMICI Python package can be customized using a number of environment variables:

Variable	Purpose	Example
SWIG	Path to the <i>SWIG</i> executable	SWIG=\$HOME/bin/swig4.0
CC	Setting the C(++) compiler	CC=/usr/bin/g++
CFLAGS	Extra compiler flags used in every compiler invocation	
BLAS_CFLAGS	<b>Compiler flags for, e.g. BLAS</b> include directories	
BLAS_LIBS	Flags for linking BLAS	
ENABLE_GCOV_COVERAGE	Set to build AMICI to generate code coverage information	ENABLE_GCOV_COVERAGE=TRUE
ENABLE_AMICI_DEBUGGING	Set to build AMICI with debugging symbols	ENABLE_AMICI_DEBUGGING=TRUE
AMICI_PARALLEL_COMPILE	Set to the number of parallel processes to be used for C(++) compilation (defaults to 1)	AMICI_PARALLEL_COMPILE=4
AMICI_TRY_ENABLE_HDF5	Whether to build AMICI with HDF5-support if possible. Default: ON	AMICI_TRY_ENABLE_HDF5=OFF

## Installation under Anaconda

To use an Anaconda installation of Python (<https://www.anaconda.com/distribution/>, Python>=3.7), proceed as follows:

Since Anaconda provides own versions of some packages which might not work with AMICI (in particular the gcc compiler), create a minimal virtual environment via:

```
conda create --name ENV_NAME pip python
```

Here, replace ENV\_NAME by some name for the environment.

To activate the environment, run:

```
source activate ENV_NAME
```

(and `conda deactivate` later to deactivate it again).

*SWIG* must be installed and available in your PATH, and a CBLAS-compatible BLAS must be available. You can also use conda to install the latter locally, using:

```
conda install -c conda-forge openblas
```

To make AMICI use openblas, set the following environment variable:

```
export BLAS_LIBS=-lopenblas
```

BLAS\_LIBS needs to be set during installation of the AMICI package, as well as during any future model import.

To install AMICI, now run:

```
pip install amici
```

The pip option `--no-cache` may be helpful here to make sure the installation is done completely anew.

Now, you are ready to use AMICI in the virtual environment.

---

**Note: Anaconda on Mac**

If the above installation does not work for you, try installing AMICI via:

```
CFLAGS="-stdlib=libc++" CC=clang CXX=clang pip3 install --verbose amici
```

This will use the clang compiler.

You will have to pass the same options when compiling any model later on. This can be done by inserting the following code before model import:

```
import os
os.environ['CC'] = 'clang'
os.environ['CXX'] = 'clang'
os.environ['CFLAGS'] = '-stdlib=libc++'
```

(For further discussion see <https://github.com/AMICI-dev/AMICI/issues/357>)

---

## Optional Boost support

Boost is an optional C++ dependency only required for special functions (including e.g. gamma derivatives) in the Python interface. Boost can be installed via package managers via

```
apt-get install libboost-math-dev
```

or

```
brew install boost
```

As only headers are required, also a [source code](#) download suffices. The compiler must be able to find the module in the search path.

## 10.2 Examples

Various example notebooks.

### 10.2.1 Getting Started in AMICI

This notebook is a brief tutorial for new users that explains the first steps necessary for model simulation in AMICI, including pointers to documentation and more advanced notebooks.

## Model Compilation

Before simulations can be run, the model must be imported and compiled. In this process, AMICI performs all symbolic manipulations that later enable scalable simulations and efficient sensitivity computation. The first step towards model compilation is the creation of an `SbmlImporter` instance, which requires an SBML Document that specifies the model using the [Systems Biology Markup Language \(SBML\)](#).

For the purpose of this tutorial, we will use `model_steadystate_scaled.xml`, which is contained in the same directory as this notebook.

```
[1]: import amici

sbml_importer = amici.SbmlImporter("model_steadystate_scaled.xml")
```

Next, we will compile the model as python extension using the `amici.SBMLImporter.sbml2amici` method. The first two arguments of this method are the name of the model, which will also be the name of the generated python module, and the model directory, which defines the directory in which the model module will be placed. Compilation will take a couple of seconds.

```
[2]: model_name = "model_steadystate"
model_dir = "model_dir"
sbml_importer.sbml2amici(model_name, model_dir)
```

## Loading the model module

To run simulations, we need to instantiate `amici.Model` and `amici.Solver` instances. As simulations require instances matching the imported model, they have to be imported from the generated model module.

```
[3]: # load the model module
model_module = amici.import_model_module(model_name, model_dir)
# instantiate model
model = model_module.getModel()
# instantiate solver
solver = model.getSolver()
```

The model allows the user to manipulate model related properties of simulations. This includes the values of model parameters that can be set by using `amici.Model.setParameterByName`. Here, we set the model parameter `p1` to a value of `1e-3`.

```
[4]: model.setParameterByName("p1", 1e-3)
```

In contrast, the solver instance allows the specification of simulation related properties. This includes setting options for the SUNDIALS solver such as absolute tolerances via `amici.Solver.setAbsoluteTolerance`. Here we set the absolute integration tolerances to `1e-10`.

```
[5]: solver.setAbsoluteTolerance(1e-10)
```

## Running Model Simulations

Model simulations can be executed using the `amici.runAmiciSimulations` routine. By default the model does not contain any timepoints for which the model is to be simulated. Here we define a simulation timecourse with two timepoints at 0 and 1 and then run the simulation.

```
[6]: # set timepoints
model.setTimepoints([0, 1])
rdata = amici.runAmiciSimulation(model, solver)
```

Simulation results are returned as `ReturnData` instance. The simulated SBML species are stored as `x` attribute, where rows correspond to the different timepoints and columns correspond to different species.

```
[7]: rdata.x
[7]: array([[0.1      , 0.4      , 0.7      ],
          [0.98208413, 0.51167992, 0.10633388]])
```

All results attributes are always ordered according to the model. For species, this means that the columns of `rdata.x` match the ordering of species in the model, which can be accessed as `amici.Model.getStateNames`

```
[8]: model.getStateNames()
[8]: ('x1', 'x2', 'x3')
```

This notebook only explains the basics of AMICI simulations. In general, AMICI simulations are highly customizable and can also be used to simulate sensitivities. The [ExampleSteadystate](#) notebook in this folder gives more detail about the model employed here and goes into the basics of sensitivity analysis. The [ExampleEquilibrationLogic](#) notebook, builds on this by using a modified version of this model to give detailed insights into the methods and options to compute steady states before and after simulations, as well as respective sensitivities. The [ExampleExperimentalConditions](#) notebook, goes into the details of how even more complex experimental setups, such as addition of drugs at predefined timepoints, can be simulated in AMICI. Finally, the [petab](#) notebook explains how standardized definitions of experimental data and conditions in the [PEtab](#) format can be imported in AMICI.

## 10.2.2 SBML import, observation model, sensitivity analysis, data export and visualization

This is an example using the `[model_steadystate_scaled.sbml]` model to demonstrate:

- SBML import
- specifying the observation model
- performing sensitivity analysis
- exporting and visualizing simulation results

```
[1]: # SBML model we want to import
sbml_file = "model_steadystate_scaled_without_observables.xml"
# Name of the model that will also be the name of the python module
model_name = "model_steadystate_scaled"
# Directory to which the generated model code is written
model_output_dir = model_name
```

(continues on next page)

(continued from previous page)

```
import libsbml
import amici
import numpy as np
import matplotlib.pyplot as plt
```

### The example model

Here we use libsbml to show the reactions and species described by the model (this is independent of AMICI).

```
[2]: sbml_reader = libsbml.SBMLReader()
sbml_doc = sbml_reader.readSBML(sbml_file)
sbml_model = sbml_doc.getModel()
dir(sbml_doc)

print("Species: ", [s.getId() for s in sbml_model.getListOfSpecies()])

print("\nReactions:")
for reaction in sbml_model.getListOfReactions():
    reactants = " + ".join(
        [
            "%s %s"
            % (
                int(r.getStoichiometry()) if r.getStoichiometry() > 1 else "",
                r.getSpecies(),
            )
            for r in reaction.getListOfReactants()
        ]
    )
    products = " + ".join(
        [
            "%s %s"
            % (
                int(r.getStoichiometry()) if r.getStoichiometry() > 1 else "",
                r.getSpecies(),
            )
            for r in reaction.getListOfProducts()
        ]
    )
    reversible = "<" if reaction.getReversible() else ""
    print(
        "%3s: %10s %1s->%10s\t\t[%s]"
        % (
            reaction.getId(),
            reactants,
            reversible,
            products,
            libsbml.formulaToL3String(reaction.getKineticLaw().getMath()),
        )
    )
```



```
Species:  ['x1', 'x2', 'x3']
```

```
Reactions:
```

```

r1:      2 x1  ->      x2      [p1 * x1^2]
r2:    x1 + x2  ->      x3      [p2 * x1 * x2]
r3:      x2  ->      2 x1      [p3 * x2]
r4:      x3  ->  x1 + x2      [p4 * x3]
r5:      x3  ->      [k0 * x3]
r6:      ->      x1      [p5]
```

## Importing an SBML model, compiling and generating an AMICI module

Before we can use AMICI to simulate our model, the SBML model needs to be translated to C++ code. This is done by `amici.SbmlImporter`.

```
[3]: # Create an SbmlImporter instance for our SBML model
sbml_importer = amici.SbmlImporter(sbml_file)
```

In this example, we want to specify fixed parameters, observables and a  $\sigma$  parameter. Unfortunately, the latter two are not part of the [SBML standard](#). However, they can be provided to `amici.SbmlImporter.sbml2amici` as demonstrated in the following.

### Constant parameters

Constant parameters, i.e. parameters with respect to which no sensitivities are to be computed (these are often parameters specifying a certain experimental condition) are provided as a list of parameter names.

```
[4]: constant_parameters = ["k0"]
```

### Observables

Specifying observables is beyond the scope of SBML. Here we define them manually.

If you are looking for a more scalable way for defining observables, then checkout [PEtab](#). Another possibility is using SBML's `AssignmentRules` [https://sbml.org/software/libsbml/5.18.0/docs/formatted/python-api/classlibsbml\\_1\\_1\\_assignment\\_rule.html](https://sbml.org/software/libsbml/5.18.0/docs/formatted/python-api/classlibsbml_1_1_assignment_rule.html) to specify model outputs within the SBML file.

```
[5]: # Define observables
observables = {
    "observable_x1": {"name": "", "formula": "x1"},
    "observable_x2": {"name": "", "formula": "x2"},
    "observable_x3": {"name": "", "formula": "x3"},
    "observable_x1_scaled": {"name": "", "formula": "scaling_x1 * x1"},
    "observable_x2_offsetted": {"name": "", "formula": "offset_x2 + x2"},
    "observable_x1withsigma": {"name": "", "formula": "x1"},
}
```

### $\sigma$ parameters

To specify measurement noise as a parameter, we simply provide a dictionary with (preexisting) parameter names as keys and a list of observable names as values to indicate which sigma parameter is to be used for which observable.

```
[6]: sigmas = {"observable_x1withsigma": "observable_x1withsigma_sigma"}
```

### Generating the module

Now we can generate the python module for our model. `amici.SbmlImporter.sbml2amici` will symbolically derive the sensitivity equations, generate C++ code for model simulation, and assemble the python module. Standard logging verbosity levels can be passed to this function to see timestamped progression during code generation.

```
[7]: import logging

sbml_importer.sbml2amici(
    model_name,
    model_output_dir,
    verbose=logging.INFO,
    observables=observables,
    constant_parameters=constant_parameters,
    sigmas=sigmas,
)

2023-11-21 19:18:10.827 - amici.sbml_import - INFO - Finished importing SBML          ↵
↪      (6.00E-02s)
2023-11-21 19:18:10.905 - amici.sbml_import - INFO - Finished processing SBML ↵
↪ observables      (7.17E-02s)
2023-11-21 19:18:10.910 - amici.sbml_import - INFO - Finished processing SBML event ↵
↪ observables      (3.22E-06s)
2023-11-21 19:18:10.969 - amici.de_export - INFO - Finished importing SbmlImporter ↵
↪      (3.09E-02s)
2023-11-21 19:18:11.926 - amici.de_export - INFO - Finished generating cpp code ↵
↪      (9.40E-01s)
2023-11-21 19:18:29.435 - amici.de_export - INFO - Finished compiling cpp code ↵
↪      (1.75E+01s)
```

### Importing the module and loading the model

If everything went well, we can now import the newly generated Python module containing our model:

```
[8]: model_module = amici.import_model_module(model_name, model_output_dir)
```

And get an instance of our model from which we can retrieve information such as parameter names:

```
[9]: model = model_module.getModel()

print("Model name:      ", model.getName())
print("Model parameters:", model.getParameterIds())
print("Model outputs:   ", model.getObservableIds())
print("Model states:    ", model.getStateIds())
```

```

Model name:      model_steadystate_scaled
Model parameters: ('p1', 'p2', 'p3', 'p4', 'p5', 'scaling_x1', 'offset_x2', 'observable_
↳x1withsigma_sigma')
Model outputs:   ('observable_x1', 'observable_x2', 'observable_x3', 'observable_x1_
↳scaled', 'observable_x2_offsetted', 'observable_x1withsigma')
Model states:    ('x1', 'x2', 'x3')

```

## Running simulations and analyzing results

After importing the model, we can run simulations using `amici.runAmiciSimulation`. This requires a `Model` instance and a `Solver` instance. Optionally you can provide measurements inside an `ExpData` instance, as shown later in this notebook.

```

[10]: # Create Model instance
model = model_module.getModel()

# set timepoints for which we want to simulate the model
model.setTimepoints(np.linspace(0, 60, 60))

# Create solver instance
solver = model.getSolver()

# Run simulation using default model parameters and solver options
rdata = amici.runAmiciSimulation(model, solver)

```

```

[11]: print(
      "Simulation was run using model default parameters as specified in the SBML model:"
    )
print(dict(zip(model.getParameterIds(), model.getParameters()))))

Simulation was run using model default parameters as specified in the SBML model:
{'p1': 1.0, 'p2': 0.5, 'p3': 0.4, 'p4': 2.0, 'p5': 0.1, 'scaling_x1': 2.0, 'offset_x2': 3.0, 'observable_x1withsigma_sigma': 0.2}

```

Simulation results are provided as `numpy.ndarrays` in the returned dictionary:

```

[12]: # np.set_printoptions(threshold=8, edgeitems=2)
for key, value in rdata.items():
    print("%12s: " % key, value)

      ts: [ 0.          1.01694915  2.03389831  3.05084746  4.06779661  5.08474576
  6.10169492  7.11864407  8.13559322  9.15254237 10.16949153 11.18644068
12.20338983 13.22033898 14.23728814 15.25423729 16.27118644 17.28813559
18.30508475 19.3220339  20.33898305 21.3559322  22.37288136 23.38983051
24.40677966 25.42372881 26.44067797 27.45762712 28.47457627 29.49152542
30.50847458 31.52542373 32.54237288 33.55932203 34.57627119 35.59322034
36.61016949 37.62711864 38.6440678  39.66101695 40.6779661  41.69491525
42.71186441 43.72881356 44.74576271 45.76271186 46.77966102 47.79661017
48.81355932 49.83050847 50.84745763 51.86440678 52.88135593 53.89830508
54.91525424 55.93220339 56.94915254 57.96610169 58.98305085 60.          ]
      x: [[0.1          0.4          0.7          ]
 [0.57995052 0.73365809 0.0951589 ]
 [0.55996496 0.71470091 0.0694127 ]

```

(continues on next page)

(continued from previous page)

```

[0.5462855 0.68030366 0.06349394]
[0.53561883 0.64937432 0.05923555]
[0.52636487 0.62259567 0.05568686]
[0.51822013 0.59943346 0.05268079]
[0.51103767 0.57935661 0.05012037]
[0.5047003 0.56191592 0.04793052]
[0.49910666 0.54673518 0.0460508 ]
[0.49416809 0.53349812 0.04443205]
[0.48980687 0.52193767 0.04303399]
[0.48595476 0.51182731 0.04182339]
[0.48255176 0.50297412 0.04077267]
[0.47954511 0.49521318 0.03985882]
[0.47688833 0.48840304 0.03906254]
[0.47454049 0.48242198 0.03836756]
[0.47246548 0.47716502 0.0377601 ]
[0.47063147 0.47254128 0.03722844]
[0.46901037 0.46847202 0.03676259]
[0.46757739 0.46488881 0.03635397]
[0.46631065 0.46173207 0.03599523]
[0.46519082 0.45894987 0.03568002]
[0.46420083 0.45649684 0.03540285]
[0.4633256 0.45433332 0.03515899]
[0.4625518 0.45242457 0.03494429]
[0.46186768 0.45074016 0.03475519]
[0.46126282 0.44925337 0.03458856]
[0.46072804 0.44794075 0.03444166]
[0.46025521 0.44678168 0.03431212]
[0.45983714 0.44575804 0.03419784]
[0.45946749 0.44485388 0.03409701]
[0.45914065 0.44405514 0.03400802]
[0.45885167 0.44334947 0.03392946]
[0.45859615 0.44272595 0.03386009]
[0.45837021 0.44217497 0.03379883]
[0.45817043 0.44168805 0.03374473]
[0.45799379 0.44125772 0.03369693]
[0.4578376 0.44087738 0.03365471]
[0.45769949 0.44054121 0.0336174 ]
[0.45757737 0.44024405 0.03358444]
[0.45746939 0.43998137 0.03355531]
[0.45737391 0.43974917 0.03352956]
[0.45728948 0.43954389 0.03350681]
[0.45721483 0.43936242 0.0334867 ]
[0.45714882 0.43920198 0.03346892]
[0.45709045 0.43906014 0.03345321]
[0.45703884 0.43893474 0.03343932]
[0.4569932 0.43882387 0.03342704]
[0.45695285 0.43872584 0.03341618]
[0.45691717 0.43863917 0.03340658]
[0.45688561 0.43856254 0.0333981 ]
[0.45685771 0.43849478 0.0333906 ]
[0.45683304 0.43843488 0.03338397]
[0.45681123 0.4383819 0.0333781 ]

```

(continues on next page)

(continued from previous page)

```

[0.45679194 0.43833507 0.03337292]
[0.45677488 0.43829365 0.03336833]
[0.4567598 0.43825703 0.03336428]
[0.45674646 0.43822466 0.0333607 ]
[0.45673467 0.43819603 0.03335753]]
    x0: [0.1 0.4 0.7]
    x_ss: [nan nan nan]
    sx: None
    sx0: None
    sx_ss: None
    y: [[0.1      0.4      0.7      0.2      3.4      0.1      ]
[0.57995052 0.73365809 0.0951589 1.15990103 3.73365809 0.57995052]
[0.55996496 0.71470091 0.0694127 1.11992992 3.71470091 0.55996496]
[0.5462855 0.68030366 0.06349394 1.092571 3.68030366 0.5462855 ]
[0.53561883 0.64937432 0.05923555 1.07123766 3.64937432 0.53561883]
[0.52636487 0.62259567 0.05568686 1.05272975 3.62259567 0.52636487]
[0.51822013 0.59943346 0.05268079 1.03644027 3.59943346 0.51822013]
[0.51103767 0.57935661 0.05012037 1.02207533 3.57935661 0.51103767]
[0.5047003 0.56191592 0.04793052 1.00940059 3.56191592 0.5047003 ]
[0.49910666 0.54673518 0.0460508 0.99821331 3.54673518 0.49910666]
[0.49416809 0.53349812 0.04443205 0.98833618 3.53349812 0.49416809]
[0.48980687 0.52193767 0.04303399 0.97961374 3.52193767 0.48980687]
[0.48595476 0.51182731 0.04182339 0.97190952 3.51182731 0.48595476]
[0.48255176 0.50297412 0.04077267 0.96510352 3.50297412 0.48255176]
[0.47954511 0.49521318 0.03985882 0.95909022 3.49521318 0.47954511]
[0.47688833 0.48840304 0.03906254 0.95377667 3.48840304 0.47688833]
[0.47454049 0.48242198 0.03836756 0.94908097 3.48242198 0.47454049]
[0.47246548 0.47716502 0.0377601 0.94493095 3.47716502 0.47246548]
[0.47063147 0.47254128 0.03722844 0.94126293 3.47254128 0.47063147]
[0.46901037 0.46847202 0.03676259 0.93802074 3.46847202 0.46901037]
[0.46757739 0.46488881 0.03635397 0.93515478 3.46488881 0.46757739]
[0.46631065 0.46173207 0.03599523 0.9326213 3.46173207 0.46631065]
[0.46519082 0.45894987 0.03568002 0.93038164 3.45894987 0.46519082]
[0.46420083 0.45649684 0.03540285 0.92840166 3.45649684 0.46420083]
[0.4633256 0.45433332 0.03515899 0.92665119 3.45433332 0.4633256 ]
[0.4625518 0.45242457 0.03494429 0.9251036 3.45242457 0.4625518 ]
[0.46186768 0.45074016 0.03475519 0.92373536 3.45074016 0.46186768]
[0.46126282 0.44925337 0.03458856 0.92252564 3.44925337 0.46126282]
[0.46072804 0.44794075 0.03444166 0.92145608 3.44794075 0.46072804]
[0.46025521 0.44678168 0.03431212 0.92051041 3.44678168 0.46025521]
[0.45983714 0.44575804 0.03419784 0.91967427 3.44575804 0.45983714]
[0.45946749 0.44485388 0.03409701 0.91893498 3.44485388 0.45946749]
[0.45914065 0.44405514 0.03400802 0.91828131 3.44405514 0.45914065]
[0.45885167 0.44334947 0.03392946 0.91770333 3.44334947 0.45885167]
[0.45859615 0.44272595 0.03386009 0.91719229 3.44272595 0.45859615]
[0.45837021 0.44217497 0.03379883 0.91674042 3.44217497 0.45837021]
[0.45817043 0.44168805 0.03374473 0.91634087 3.44168805 0.45817043]
[0.45799379 0.44125772 0.03369693 0.91598758 3.44125772 0.45799379]
[0.4578376 0.44087738 0.03365471 0.9156752 3.44087738 0.4578376 ]
[0.45769949 0.44054121 0.0336174 0.91539898 3.44054121 0.45769949]
[0.45757737 0.44024405 0.03358444 0.91515474 3.44024405 0.45757737]
[0.45746939 0.43998137 0.03355531 0.91493878 3.43998137 0.45746939]

```

(continues on next page)

(continues on next page)

(continues on next page)

(continued from previous page)

[illegible]

(continues on next page)



[illegible]

(continues on next page)

(continued from previous page)

```
cpu_time_total: 3.079
```

```
[13]: # In particular for interactive use, ReturnDataView.by_id() and amici.evaluate provides
      ↪ a more convenient way to access slices of the result:
      # Time trajectory of observable observable_x1
      print(f"{rdata.by_id('observable_x1')}=")
      # Time trajectory of state variable x2
      print(f"{rdata.by_id('x2')}=")

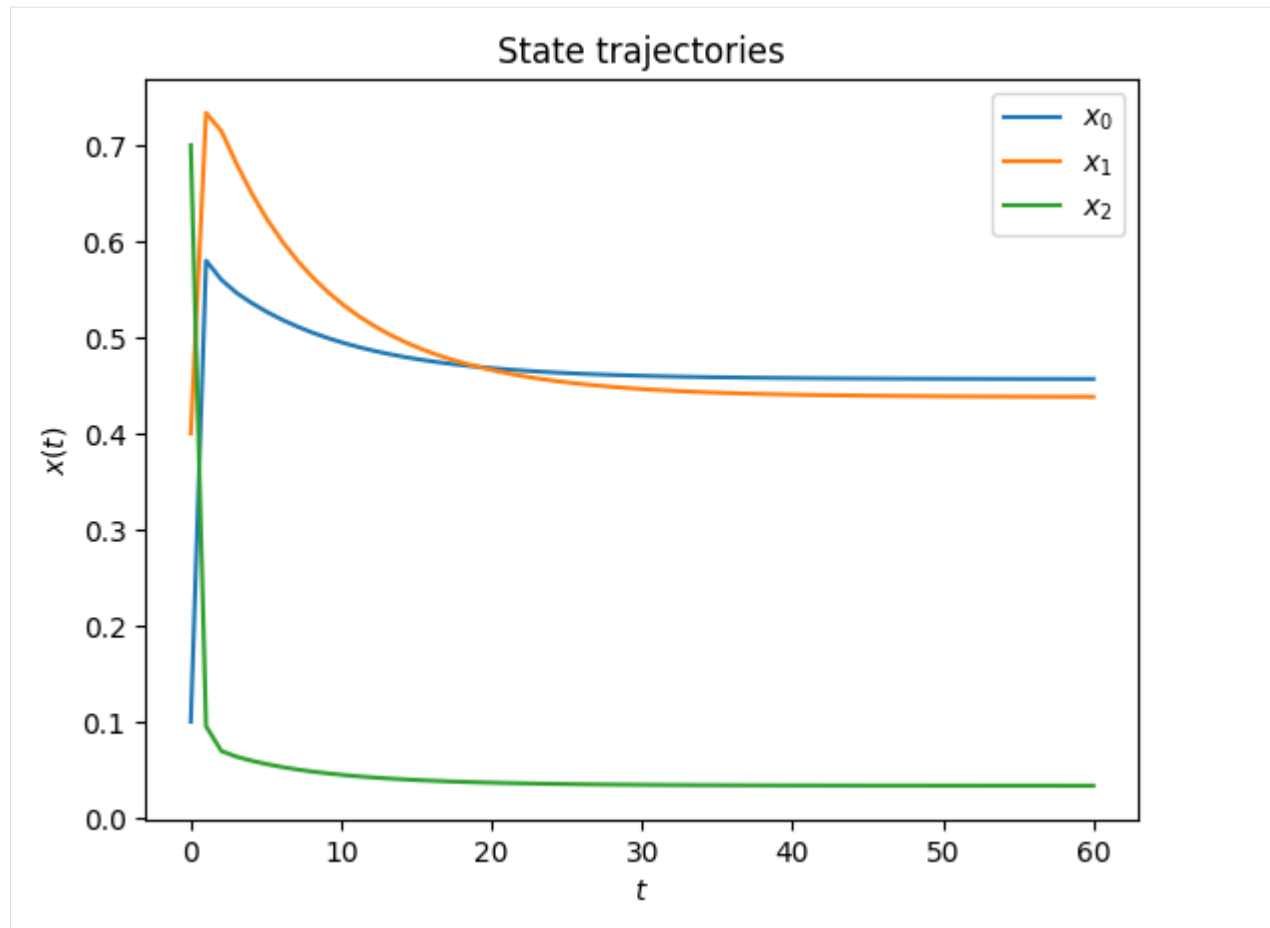
      rdata.by_id('observable_x1')=array([0.1          , 0.57995052, 0.55996496, 0.5462855 , 0.
      ↪ 53561883,
          0.52636487, 0.51822013, 0.51103767, 0.5047003 , 0.49910666,
          0.49416809, 0.48980687, 0.48595476, 0.48255176, 0.47954511,
          0.47688833, 0.47454049, 0.47246548, 0.47063147, 0.46901037,
          0.46757739, 0.46631065, 0.46519082, 0.46420083, 0.4633256 ,
          0.4625518 , 0.46186768, 0.46126282, 0.46072804, 0.46025521,
          0.45983714, 0.45946749, 0.45914065, 0.45885167, 0.45859615,
          0.45837021, 0.45817043, 0.45799379, 0.4578376 , 0.45769949,
          0.45757737, 0.45746939, 0.45737391, 0.45728948, 0.45721483,
          0.45714882, 0.45709045, 0.45703884, 0.4569932 , 0.45695285,
          0.45691717, 0.45688561, 0.45685771, 0.45683304, 0.45681123,
          0.45679194, 0.45677488, 0.4567598 , 0.45674646, 0.45673467])
      rdata.by_id('x2')=array([0.4          , 0.73365809, 0.71470091, 0.68030366, 0.64937432,
          0.62259567, 0.59943346, 0.57935661, 0.56191592, 0.54673518,
          0.53349812, 0.52193767, 0.51182731, 0.50297412, 0.49521318,
          0.48840304, 0.48242198, 0.47716502, 0.47254128, 0.46847202,
          0.46488881, 0.46173207, 0.45894987, 0.45649684, 0.45433332,
          0.45242457, 0.45074016, 0.44925337, 0.44794075, 0.44678168,
          0.44575804, 0.44485388, 0.44405514, 0.44334947, 0.44272595,
          0.44217497, 0.44168805, 0.44125772, 0.44087738, 0.44054121,
          0.44024405, 0.43998137, 0.43974917, 0.43954389, 0.43936242,
          0.43920198, 0.43906014, 0.43893474, 0.43882387, 0.43872584,
          0.43863917, 0.43856254, 0.43849478, 0.43843488, 0.4383819 ,
          0.43833507, 0.43829365, 0.43825703, 0.43822466, 0.43819603])
```

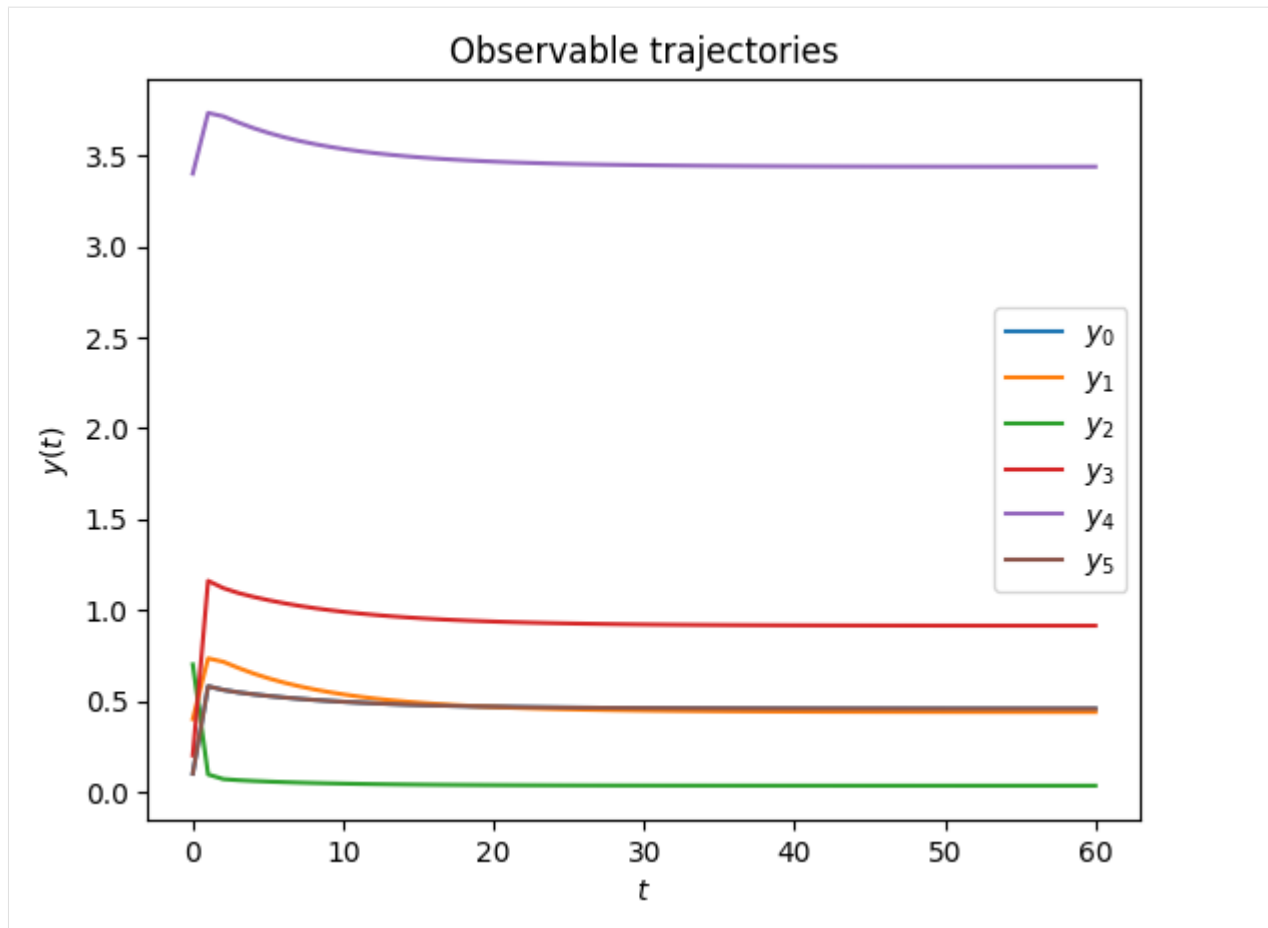
## Plotting trajectories

The simulation results above did not look too appealing. Let's plot the trajectories of the model states and outputs them using `matplotlib.pyplot`:

```
[14]: import amici.plotting

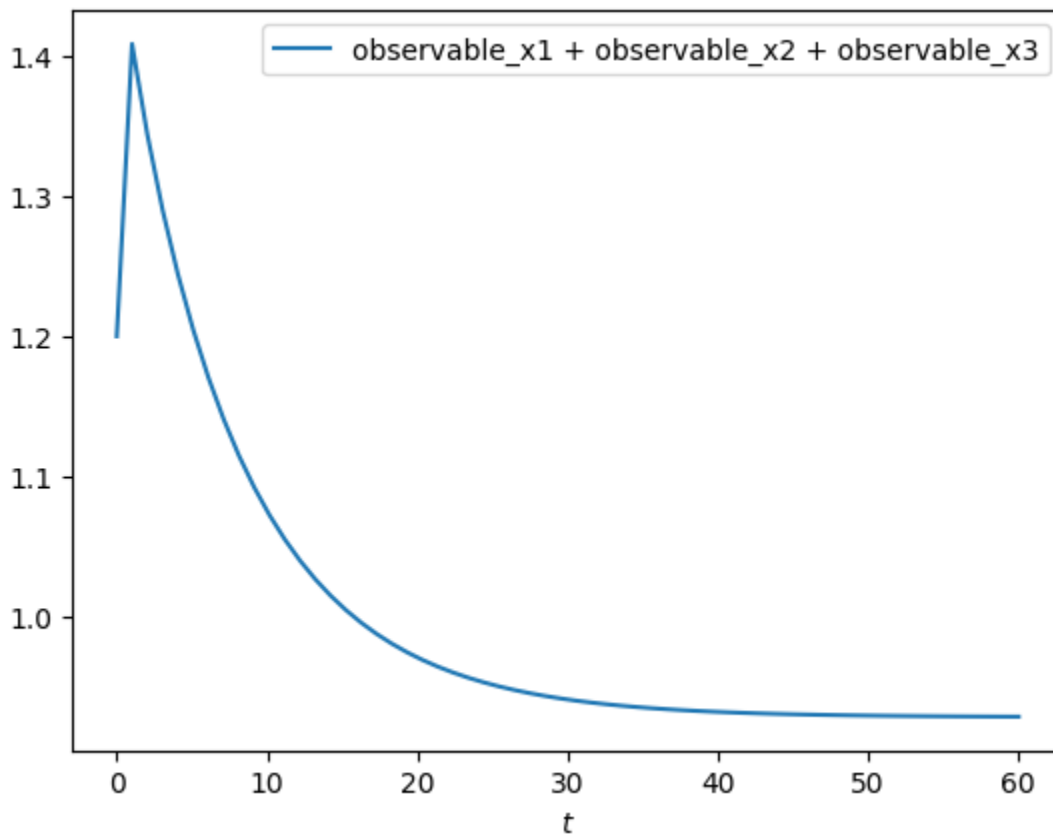
      amici.plotting.plot_state_trajectories(rdata, model=None)
      amici.plotting.plot_observable_trajectories(rdata, model=None)
```





We can also evaluate symbolic expressions of model quantities using `amici.numpy.evaluate`, or directly plot the results using `amici.plotting.plot_expressions`:

```
[15]: amici.plotting.plot_expressions(  
      "observable_x1 + observable_x2 + observable_x3", rdata=rdata  
)
```



### Computing likelihood

Often model parameters need to be inferred from experimental data. This is commonly done by maximizing the likelihood of observing the data given to current model parameters. AMICI will compute this likelihood if experimental data is provided to `amici.runAmiciSimulation` as optional third argument. Measurements along with their standard deviations are provided through an `amici.ExpData` instance.

```
[16]: # Create model instance and set time points for simulation
model = model_module.getModel()
model.setTimepoints(np.linspace(0, 10, 11))

# Create solver instance, keep default options
solver = model.getSolver()

# Run simulation without experimental data
rdata = amici.runAmiciSimulation(model, solver)

# Create ExpData instance from simulation results
edata = amici.ExpData(rdata, 1.0, 0.0)

# Re-run simulation, this time passing "experimental data"
rdata = amici.runAmiciSimulation(model, solver, edata)

print("Log-likelihood %f" % rdata["llh"])
```

```
Log-likelihood -86.282099
```

## Simulation tolerances

Numerical error tolerances are often critical to get accurate results. For the state variables, integration errors can be controlled using `setRelativeTolerance` and `setAbsoluteTolerance`. Similar functions exist for sensitivities, steadystates and quadratures. We initially compute a reference solution using extremely low tolerances and then assess the influence on integration error for different levels of absolute and relative tolerance.

```
[17]: solver.setRelativeTolerance(1e-16)
      solver.setAbsoluteTolerance(1e-16)
      solver.setSensitivityOrder(amici.SensitivityOrder.none)
      rdata_ref = amici.runAmiciSimulation(model, solver, edata)

def get_simulation_error(solver):
    rdata = amici.runAmiciSimulation(model, solver, edata)
    return np.mean(np.abs(rdata["x"] - rdata_ref["x"])), np.mean(
        np.abs(rdata["llh"] - rdata_ref["llh"])
    )

def get_errors(tolfun, tols):
    solver.setRelativeTolerance(1e-16)
    solver.setAbsoluteTolerance(1e-16)
    x_errs = []
    llh_errs = []
    for tol in tols:
        getattr(solver, tolfun)(tol)
        x_err, llh_err = get_simulation_error(solver)
        x_errs.append(x_err)
        llh_errs.append(llh_err)
    return x_errs, llh_errs

atols = np.logspace(-5, -15, 100)
atol_x_errs, atol_llh_errs = get_errors("setAbsoluteTolerance", atols)

rtols = np.logspace(-5, -15, 100)
rtol_x_errs, rtol_llh_errs = get_errors("setRelativeTolerance", rtols)

fig, axes = plt.subplots(1, 2, figsize=(15, 5))

def plot_error(tols, x_errs, llh_errs, tolname, ax):
    ax.plot(tols, x_errs, "r-", label="x")
    ax.plot(tols, llh_errs, "b-", label="llh")
    ax.set_xscale("log")
    ax.set_yscale("log")
    ax.set_xlabel(f"{tolname} tolerance")
    ax.set_ylabel("average numerical error")
```

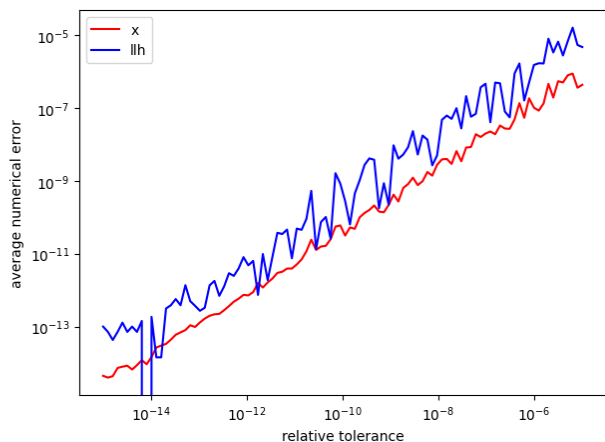
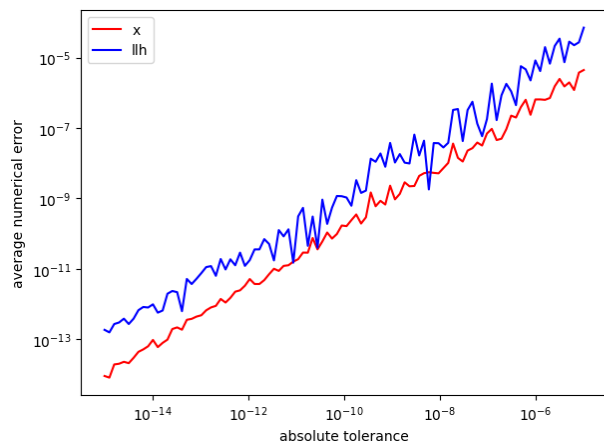
(continues on next page)

(continued from previous page)

```
ax.legend()
```

```
plot_error(atols, atol_x_errs, atol_llh_errs, "absolute", axes[0])
plot_error(rtols, rtol_x_errs, rtol_llh_errs, "relative", axes[1])
```

```
# reset relative tolerance to default value
solver.setRelativeTolerance(1e-8)
solver.setRelativeTolerance(1e-16)
```



## Sensitivity analysis

AMICI can provide first- and second-order sensitivities using the forward- or adjoint-method. The respective options are set on the Model and Solver objects.

### Forward sensitivity analysis

```
[18]: model = model_module.getModel()
model.setTimepoints(np.linspace(0, 10, 11))
model.requireSensitivitiesForAllParameters() # sensitivities w.r.t. all parameters
# model.setParameterList([1, 2])           # sensitivities
# w.r.t. the specified parameters
model.setParameterScale(
    amici.ParameterScaling.none
) # parameters are used as-is (not log-transformed)

solver = model.getSolver()
solver.setSensitivityMethod(
    amici.SensitivityMethod.forward
) # forward sensitivity analysis
solver.setSensitivityOrder(
    amici.SensitivityOrder.first
) # first-order sensitivities

rdata = amici.runAmiciSimulation(model, solver)
```

(continues on next page)

(continued from previous page)

```

# print sensitivity-related results
for key, value in rdata.items():
    if key.startswith("s"):
        print("%12s: " % key, value)

sx: [[[ 0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]]]

[[-2.00747250e-01  1.19873139e-01 -9.44167985e-03]
 [-1.02561396e-01 -1.88820454e-01  1.01855972e-01]
 [ 4.66193077e-01 -2.86365372e-01  2.39662449e-02]
 [ 4.52560294e-02  1.14631370e-01 -3.34067919e-02]
 [ 4.00672911e-01  1.92564093e-01  4.98877759e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]]]

[[-2.23007240e-01  1.53979022e-01 -1.26885280e-02]
 [-1.33426939e-01 -3.15955239e-01  9.49575030e-02]
 [ 5.03470377e-01 -3.52731535e-01  2.81567412e-02]
 [ 3.93630714e-02  1.10770683e-01 -1.05673869e-02]
 [ 5.09580304e-01  4.65255489e-01  9.24843702e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]]]

[[-2.14278104e-01  1.63465064e-01 -1.03268418e-02]
 [-1.60981967e-01 -4.00490452e-01  7.54810648e-02]
 [ 4.87746419e-01 -3.76014315e-01  2.30919334e-02]
 [ 4.28733680e-02  1.15473583e-01 -6.63571687e-03]
 [ 6.05168647e-01  7.07226039e-01  1.23870914e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]]]

[[-2.05888038e-01  1.69308689e-01 -7.93085660e-03]
 [-1.84663809e-01 -4.65451966e-01  5.95026117e-02]
 [ 4.66407064e-01 -3.87612079e-01  1.76410128e-02]
 [ 4.52451104e-02  1.19865712e-01 -4.73313094e-03]
 [ 6.90798449e-01  9.20396633e-01  1.49475827e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]]]

[[-1.98803165e-01  1.73327268e-01 -6.03008179e-03]

```

(continues on next page)



(continued from previous page)

```

[-2.04303740e-01 -5.16111388e-01 4.68785776e-02]
[ 4.47070326e-01 -3.94304029e-01 1.32107437e-02]
[ 4.69732048e-02 1.22961727e-01 -3.35899442e-03]
[ 7.68998995e-01 1.10844286e+00 1.70889328e-01]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00]]

[[-1.92789113e-01 1.75978657e-01 -4.54517629e-03]
[-2.20500138e-01 -5.55540705e-01 3.68776526e-02]
[ 4.30424855e-01 -3.97907706e-01 9.75257113e-03]
[ 4.82793652e-02 1.24952071e-01 -2.30991637e-03]
[ 8.40805131e-01 1.27504628e+00 1.89020151e-01]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00]]

[[-1.87672774e-01 1.77588334e-01 -3.38318222e-03]
[-2.33807210e-01 -5.86081383e-01 2.89236334e-02]
[ 4.16201399e-01 -3.99295277e-01 7.06598588e-03]
[ 4.92546648e-02 1.26089711e-01 -1.50412006e-03]
[ 9.06806543e-01 1.42334018e+00 2.04522708e-01]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00]]

[[-1.83320440e-01 1.78410042e-01 -2.47240692e-03]
[-2.44690164e-01 -6.09568485e-01 2.25774266e-02]
[ 4.04061655e-01 -3.99063012e-01 4.97908386e-03]
[ 4.99612484e-02 1.26581014e-01 -8.85891342e-04]
[ 9.67473970e-01 1.55589415e+00 2.17895305e-01]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00]]

[[-1.79620591e-01 1.78640114e-01 -1.75822439e-03]
[-2.53540123e-01 -6.27448857e-01 1.75019839e-02]
[ 3.93704970e-01 -3.97656641e-01 3.35895484e-03]
[ 5.04492282e-02 1.26586733e-01 -4.13401240e-04]
[ 1.02322336e+00 1.67481439e+00 2.29524046e-01]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00]]

[[-1.76478441e-01 1.78430281e-01 -1.19867662e-03]
[-2.60686971e-01 -6.40868686e-01 1.34365068e-02]
[ 3.84873835e-01 -3.95414931e-01 2.10369522e-03]
[ 5.07601805e-02 1.26231631e-01 -5.46465317e-05]
[ 1.07443160e+00 1.78183962e+00 2.39710937e-01]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00]]]

```

(continues on next page)

(continued from previous page)

```

    sx0: [[0. 0. 0.]
[0. 0. 0.]
[0. 0. 0.]
[0. 0. 0.]
[0. 0. 0.]
[0. 0. 0.]
[0. 0. 0.]]
    sx_ss: [[nan nan nan]
[nan nan nan]
[nan nan nan]
[nan nan nan]
[nan nan nan]
[nan nan nan]
[nan nan nan]]
    sigmay: [[1. 1. 1. 1. 1. 0.2]
[1. 1. 1. 1. 1. 0.2]
[1. 1. 1. 1. 1. 0.2]
[1. 1. 1. 1. 1. 0.2]
[1. 1. 1. 1. 1. 0.2]
[1. 1. 1. 1. 1. 0.2]
[1. 1. 1. 1. 1. 0.2]
[1. 1. 1. 1. 1. 0.2]
[1. 1. 1. 1. 1. 0.2]
[1. 1. 1. 1. 1. 0.2]]
    sy: [[[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
 0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
 0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
 0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
 0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
 0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
 0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e-01
 0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
 0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
 0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
 0.00000000e+00  0.00000000e+00]]

[[-2.00747250e-01  1.19873139e-01 -9.44167985e-03 -4.01494500e-01
 1.19873139e-01 -2.00747250e-01]
 [-1.02561396e-01 -1.88820454e-01  1.01855972e-01 -2.05122791e-01
 -1.88820454e-01 -1.02561396e-01]
 [ 4.66193077e-01 -2.86365372e-01  2.39662449e-02  9.32386154e-01
 -2.86365372e-01  4.66193077e-01]
 [ 4.52560294e-02  1.14631370e-01 -3.34067919e-02  9.05120589e-02
 1.14631370e-01  4.52560294e-02]]

```

(continues on next page)

(continued from previous page)

```

[ 4.00672911e-01  1.92564093e-01  4.98877759e-02  8.01345822e-01
  1.92564093e-01  4.00672911e-01]
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  5.80072436e-01
  0.00000000e+00  0.00000000e+00]
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  1.00000000e+00  0.00000000e+00]
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00]]

[[-2.23007240e-01  1.53979022e-01 -1.26885280e-02 -4.46014480e-01
  1.53979022e-01 -2.23007240e-01]
[-1.33426939e-01 -3.15955239e-01  9.49575030e-02 -2.66853878e-01
 -3.15955239e-01 -1.33426939e-01]
[ 5.03470377e-01 -3.52731535e-01  2.81567412e-02  1.00694075e+00
 -3.52731535e-01  5.03470377e-01]
[ 3.93630714e-02  1.10770683e-01 -1.05673869e-02  7.87261427e-02
  1.10770683e-01  3.93630714e-02]
[ 5.09580304e-01  4.65255489e-01  9.24843702e-02  1.01916061e+00
  4.65255489e-01  5.09580304e-01]
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  5.60534516e-01
  0.00000000e+00  0.00000000e+00]
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  1.00000000e+00  0.00000000e+00]
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00]]

[[-2.14278104e-01  1.63465064e-01 -1.03268418e-02 -4.28556209e-01
  1.63465064e-01 -2.14278104e-01]
[-1.60981967e-01 -4.00490452e-01  7.54810648e-02 -3.21963935e-01
 -4.00490452e-01 -1.60981967e-01]
[ 4.87746419e-01 -3.76014315e-01  2.30919334e-02  9.75492839e-01
 -3.76014315e-01  4.87746419e-01]
[ 4.28733680e-02  1.15473583e-01 -6.63571687e-03  8.57467361e-02
  1.15473583e-01  4.28733680e-02]
[ 6.05168647e-01  7.07226039e-01  1.23870914e-01  1.21033729e+00
  7.07226039e-01  6.05168647e-01]
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  5.46870655e-01
  0.00000000e+00  0.00000000e+00]
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  1.00000000e+00  0.00000000e+00]
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00]]

[[-2.05888038e-01  1.69308689e-01 -7.93085660e-03 -4.11776077e-01
  1.69308689e-01 -2.05888038e-01]
[-1.84663809e-01 -4.65451966e-01  5.95026117e-02 -3.69327617e-01
 -4.65451966e-01 -1.84663809e-01]
[ 4.66407064e-01 -3.87612079e-01  1.76410128e-02  9.32814128e-01
 -3.87612079e-01  4.66407064e-01]
[ 4.52451104e-02  1.19865712e-01 -4.73313094e-03  9.04902208e-02
  1.19865712e-01  4.52451104e-02]
[ 6.90798449e-01  9.20396633e-01  1.49475827e-01  1.38159690e+00
  9.20396633e-01  6.90798449e-01]]

```

(continues on next page)

(continued from previous page)

```

    9.20396633e-01  6.90798449e-01]
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  5.36280366e-01
  0.00000000e+00  0.00000000e+00]
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  1.00000000e+00  0.00000000e+00]
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00]]

[[-1.98803165e-01  1.73327268e-01 -6.03008179e-03 -3.97606330e-01
  1.73327268e-01 -1.98803165e-01]
[-2.04303740e-01 -5.16111388e-01  4.68785776e-02 -4.08607480e-01
 -5.16111388e-01 -2.04303740e-01]
[ 4.47070326e-01 -3.94304029e-01  1.32107437e-02  8.94140651e-01
 -3.94304029e-01  4.47070326e-01]
[ 4.69732048e-02  1.22961727e-01 -3.35899442e-03  9.39464097e-02
  1.22961727e-01  4.69732048e-02]
[ 7.68998995e-01  1.10844286e+00  1.70889328e-01  1.53799799e+00
  1.10844286e+00  7.68998995e-01]
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  5.27091252e-01
  0.00000000e+00  0.00000000e+00]
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  1.00000000e+00  0.00000000e+00]
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00]]

[[-1.92789113e-01  1.75978657e-01 -4.54517629e-03 -3.85578227e-01
  1.75978657e-01 -1.92789113e-01]
[-2.20500138e-01 -5.55540705e-01  3.68776526e-02 -4.41000277e-01
 -5.55540705e-01 -2.20500138e-01]
[ 4.30424855e-01 -3.97907706e-01  9.75257113e-03  8.60849709e-01
 -3.97907706e-01  4.30424855e-01]
[ 4.82793652e-02  1.24952071e-01 -2.30991637e-03  9.65587304e-02
  1.24952071e-01  4.82793652e-02]
[ 8.40805131e-01  1.27504628e+00  1.89020151e-01  1.68161026e+00
  1.27504628e+00  8.40805131e-01]
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  5.18989205e-01
  0.00000000e+00  0.00000000e+00]
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  1.00000000e+00  0.00000000e+00]
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00]]

[[-1.87672774e-01  1.77588334e-01 -3.38318222e-03 -3.75345548e-01
  1.77588334e-01 -1.87672774e-01]
[-2.33807210e-01 -5.86081383e-01  2.89236334e-02 -4.67614420e-01
 -5.86081383e-01 -2.33807210e-01]
[ 4.16201399e-01 -3.99295277e-01  7.06598588e-03  8.32402797e-01
 -3.99295277e-01  4.16201399e-01]
[ 4.92546648e-02  1.26089711e-01 -1.50412006e-03  9.85093296e-02
  1.26089711e-01  4.92546648e-02]
[ 9.06806543e-01  1.42334018e+00  2.04522708e-01  1.81361309e+00
  1.42334018e+00  9.06806543e-01]]

```

(continues on next page)

(continued from previous page)

```

[ 0.00000000e+00 0.00000000e+00 0.00000000e+00 5.11829985e-01
 0.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 1.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00]]

[[-1.83320440e-01 1.78410042e-01 -2.47240692e-03 -3.66640879e-01
 1.78410042e-01 -1.83320440e-01]
[-2.44690164e-01 -6.09568485e-01 2.25774266e-02 -4.89380329e-01
 -6.09568485e-01 -2.44690164e-01]
[ 4.04061655e-01 -3.99063012e-01 4.97908386e-03 8.08123310e-01
 -3.99063012e-01 4.04061655e-01]
[ 4.99612484e-02 1.26581014e-01 -8.85891342e-04 9.99224969e-02
 1.26581014e-01 4.99612484e-02]
[ 9.67473970e-01 1.55589415e+00 2.17895305e-01 1.93494794e+00
 1.55589415e+00 9.67473970e-01]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00 5.05500234e-01
 0.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 1.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00]]

[[-1.79620591e-01 1.78640114e-01 -1.75822439e-03 -3.59241183e-01
 1.78640114e-01 -1.79620591e-01]
[-2.53540123e-01 -6.27448857e-01 1.75019839e-02 -5.07080247e-01
 -6.27448857e-01 -2.53540123e-01]
[ 3.93704970e-01 -3.97656641e-01 3.35895484e-03 7.87409940e-01
 -3.97656641e-01 3.93704970e-01]
[ 5.04492282e-02 1.26586733e-01 -4.13401240e-04 1.00898456e-01
 1.26586733e-01 5.04492282e-02]
[ 1.02322336e+00 1.67481439e+00 2.29524046e-01 2.04644672e+00
 1.67481439e+00 1.02322336e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00 4.99901907e-01
 0.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 1.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00]]

[[-1.76478441e-01 1.78430281e-01 -1.19867662e-03 -3.52956882e-01
 1.78430281e-01 -1.76478441e-01]
[-2.60686971e-01 -6.40868686e-01 1.34365068e-02 -5.21373942e-01
 -6.40868686e-01 -2.60686971e-01]
[ 3.84873835e-01 -3.95414931e-01 2.10369522e-03 7.69747670e-01
 -3.95414931e-01 3.84873835e-01]
[ 5.07601805e-02 1.26231631e-01 -5.46465317e-05 1.01520361e-01
 1.26231631e-01 5.07601805e-02]
[ 1.07443160e+00 1.78183962e+00 2.39710937e-01 2.14886320e+00
 1.78183962e+00 1.07443160e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00 4.94949118e-01

```

(continues on next page)

(continued from previous page)

```

0.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 1.00000000e+00 0.00000000e+00]
[ 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00]]
ssigmay: [[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 1.]]

[[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 1.]]

[[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 1.]]

[[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 1.]]

[[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 1.]]

[[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]

```

(continues on next page)

(continued from previous page)

```
[0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 1.]]
```

```
[[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1.]]
```

```
[[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1.]]
```

```
[[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1.]]
```

```
[[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1.]]
```

```
[[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1.]]]
```

```
sigmaz: None
```

(continues on next page)

(continued from previous page)

[illegible]

(continues on next page)



(continued from previous page)

```
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0.]
```

## Adjoint sensitivity analysis

```
[19]: # Set model options
model = model_module.getModel()
p_orig = np.array(model.getParameters())
p_orig[
    list(model.getParameterIds()).index("observable_x1withsigma_sigma")
] = 0.1 # Change default parameter
model.setParameters(p_orig)
model.setParameterScale(amici.ParameterScaling.none)
model.setTimepoints(np.linspace(0, 10, 21))

solver = model.getSolver()
solver.setMaxSteps(10**4) # Set maximum number of steps for the solver

# simulate time-course to get artificial data
rdata = amici.runAmiciSimulation(model, solver)
edata = amici.ExpData(rdata, 1.0, 0)
edata.fixedParameters = model.getFixedParameters()
# set sigma to 1.0 except for observable 5, so that p[7] is used instead
# (if we have sigma parameterized, the corresponding ExpData entries must NaN, otherwise
# they will override the parameter)
edata.setObservedDataStdDev(
    rdata["t"] * 0 + np.nan,
    list(model.getObservableIds()).index("observable_x1withsigma"),
)

# enable sensitivities
solver.setSensitivityOrder(amici.SensitivityOrder.first) # First-order ...
```

(continues on next page)

(continued from previous page)

```

solver.setSensitivityMethod(
    amici.SensitivityMethod.adjoint
) # ... adjoint sensitivities
model.requireSensitivitiesForAllParameters() # ... w.r.t. all parameters

# compute adjoint sensitivities
rdata = amici.runAmiciSimulation(model, solver, edata)
# print(rdata['sigmay'])
print("Log-likelihood: %f\nGradient: %s" % (rdata["llh"], rdata["sllh"]))

Log-likelihood: -791.969425
Gradient: [-9.55802903e+01 -1.25592040e+02  2.20808520e+02  3.07797050e+01
  4.99814472e+02 -2.31146831e+00  8.26072442e-01  1.32789855e+04]

```

### Finite differences gradient check

Compare AMICI-computed gradient with finite differences

```

[20]: from scipy.optimize import check_grad

def func(x0, symbol="llh", x0full=None, plist=[], verbose=False):
    p = x0[:]
    if len(plist):
        p = x0full[:]
        p[plist] = x0
    verbose and print("f: p=%s" % p)

    old_parameters = model.getParameters()
    solver.setSensitivityOrder(amici.SensitivityOrder.none)
    model.setParameters(p)
    rdata = amici.runAmiciSimulation(model, solver, edata)

    model.setParameters(old_parameters)

    res = np.sum(rdata[symbol])
    verbose and print(res)
    return res

def grad(x0, symbol="llh", x0full=None, plist=[], verbose=False):
    p = x0[:]
    if len(plist):
        model.setParameterList(plist)
        p = x0full[:]
        p[plist] = x0
    else:
        model.requireSensitivitiesForAllParameters()
    verbose and print("g: p=%s" % p)

    old_parameters = model.getParameters()

```

(continues on next page)

(continued from previous page)

```

solver.setSensitivityMethod(amici.SensitivityMethod.forward)
solver.setSensitivityOrder(amici.SensitivityOrder.first)
model.setParameters(p)
rdata = amici.runAmiciSimulation(model, solver, edata)

model.setParameters(old_parameters)

res = rdata["s%s" % symbol]
if not isinstance(res, float):
    if len(res.shape) == 3:
        res = np.sum(res, axis=(0, 2))
    verbose and print(res)
return res

epsilon = 1e-4
err_norm = check_grad(func, grad, p_orig, "llh", epsilon=epsilon)
print("sllh: |error|_2: %f" % err_norm)
# assert err_norm < 1e-6
print()

for ip in range(model.np()):
    plist = [ip]
    p = p_orig.copy()
    err_norm = check_grad(
        func, grad, p[plist], "llh", p, [ip], epsilon=epsilon
    )
    print("sllh: p[%d]: |error|_2: %f" % (ip, err_norm))

print()
for ip in range(model.np()):
    plist = [ip]
    p = p_orig.copy()
    err_norm = check_grad(func, grad, p[plist], "y", p, [ip], epsilon=epsilon)
    print("sy: p[%d]: |error|_2: %f" % (ip, err_norm))

print()
for ip in range(model.np()):
    plist = [ip]
    p = p_orig.copy()
    err_norm = check_grad(func, grad, p[plist], "x", p, [ip], epsilon=epsilon)
    print("sx: p[%d]: |error|_2: %f" % (ip, err_norm))

print()
for ip in range(model.np()):
    plist = [ip]
    p = p_orig.copy()
    err_norm = check_grad(
        func, grad, p[plist], "sigmay", p, [ip], epsilon=epsilon
    )
    print("ssigmay: p[%d]: |error|_2: %f" % (ip, err_norm))

```

```
sllh: |error|_2: 20.102131

sllh: p[0]: |error|_2: 0.000285
sllh: p[1]: |error|_2: 0.007832
sllh: p[2]: |error|_2: 0.037406
sllh: p[3]: |error|_2: 0.000849
sllh: p[4]: |error|_2: 0.132302
sllh: p[5]: |error|_2: 0.000280
sllh: p[6]: |error|_2: 0.001050
sllh: p[7]: |error|_2: 20.101604

sy: p[0]: |error|_2: 0.002974
sy: p[1]: |error|_2: 0.002717
sy: p[2]: |error|_2: 0.001308
sy: p[3]: |error|_2: 0.000939
sy: p[4]: |error|_2: 0.006106
sy: p[5]: |error|_2: 0.000000
sy: p[6]: |error|_2: 0.000000
sy: p[7]: |error|_2: 0.000000

sx: p[0]: |error|_2: 0.001033
sx: p[1]: |error|_2: 0.001076
sx: p[2]: |error|_2: 0.000121
sx: p[3]: |error|_2: 0.000439
sx: p[4]: |error|_2: 0.001569
sx: p[5]: |error|_2: 0.000000
sx: p[6]: |error|_2: 0.000000
sx: p[7]: |error|_2: 0.000000

ssigmay: p[0]: |error|_2: 0.000000
ssigmay: p[1]: |error|_2: 0.000000
ssigmay: p[2]: |error|_2: 0.000000
ssigmay: p[3]: |error|_2: 0.000000
ssigmay: p[4]: |error|_2: 0.000000
ssigmay: p[5]: |error|_2: 0.000000
ssigmay: p[6]: |error|_2: 0.000000
ssigmay: p[7]: |error|_2: 0.000000
```

```
[21]: eps = 1e-4
      op = model.getParameters()

      solver.setSensitivityMethod(
          amici.SensitivityMethod.forward
      ) # forward sensitivity analysis
      solver.setSensitivityOrder(
          amici.SensitivityOrder.first
      ) # first-order sensitivities
      model.requireSensitivitiesForAllParameters()
      solver.setRelativeTolerance(1e-12)
      rdata = amici.runAmiciSimulation(model, solver, edata)
```

(continues on next page)

(continued from previous page)

```

def fd(x0, ip, eps, symbol="llh"):
    p = list(x0[:])
    old_parameters = model.getParameters()
    solver.setSensitivityOrder(amici.SensitivityOrder.none)
    p[ip] += eps
    model.setParameters(p)
    rdata_f = amici.runAmiciSimulation(model, solver, edata)
    p[ip] -= 2 * eps
    model.setParameters(p)
    rdata_b = amici.runAmiciSimulation(model, solver, edata)

    model.setParameters(old_parameters)
    return (rdata_f[symbol] - rdata_b[symbol]) / (2 * eps)

def plot_sensitivities(symbol, eps):
    fig, axes = plt.subplots(4, 2, figsize=(15, 10))
    for ip in range(4):
        fd_approx = fd(model.getParameters(), ip, eps, symbol=symbol)

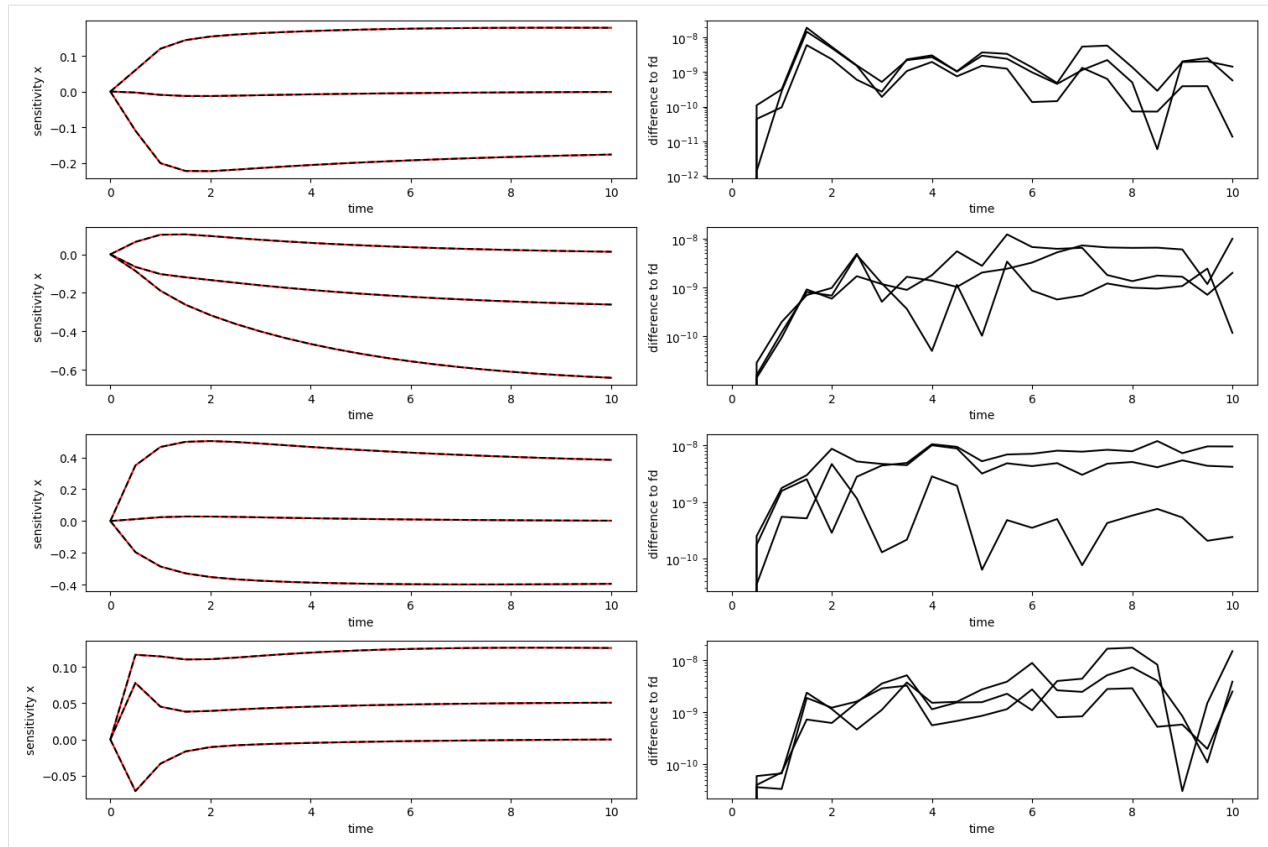
        axes[ip, 0].plot(
            edata.getTimepoints(), rdata[f"s{symbol}"][:, ip, :], "r-"
        )
        axes[ip, 0].plot(edata.getTimepoints(), fd_approx, "k--")
        axes[ip, 0].set_ylabel(f"sensitivity {symbol}")
        axes[ip, 0].set_xlabel("time")

        axes[ip, 1].plot(
            edata.getTimepoints(),
            np.abs(rdata[f"s{symbol}"][:, ip, :] - fd_approx),
            "k-",
        )
        axes[ip, 1].set_ylabel("difference to fd")
        axes[ip, 1].set_xlabel("time")
        axes[ip, 1].set_yscale("log")

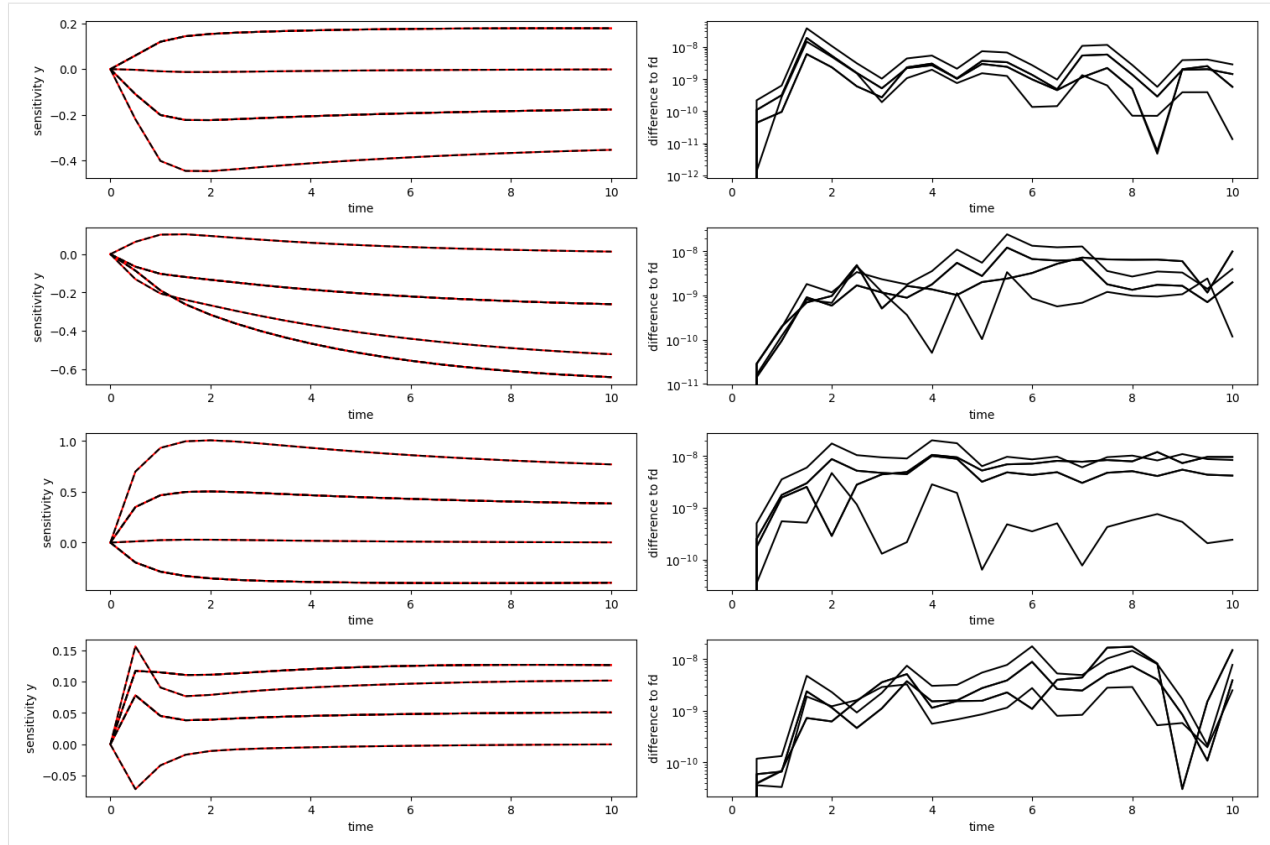
    plt.tight_layout()
    plt.show()

```

```
[22]: plot_sensitivities("x", eps)
```



```
[23]: plot_sensitivities("y", eps)
```



## Export as DataFrame

Experimental data and simulation results can both be exported as pandas DataFrame to allow for an easier inspection of numeric values

```
[24]: # run the simulation
rdata = amici.runAmiciSimulation(model, solver, edata)
```

```
[25]: # look at the ExpData as DataFrame
df = amici.getDataObservablesAsDataFrame(model, [edata])
df
```

```
[25]:
```

	condition_id	time	datatype	t_presim	k0	k0_preeq	k0_presim	\
0		0.0	data	0.0	1.0	NaN	NaN	
1		0.5	data	0.0	1.0	NaN	NaN	
2		1.0	data	0.0	1.0	NaN	NaN	
3		1.5	data	0.0	1.0	NaN	NaN	
4		2.0	data	0.0	1.0	NaN	NaN	
5		2.5	data	0.0	1.0	NaN	NaN	
6		3.0	data	0.0	1.0	NaN	NaN	
7		3.5	data	0.0	1.0	NaN	NaN	
8		4.0	data	0.0	1.0	NaN	NaN	
9		4.5	data	0.0	1.0	NaN	NaN	
10		5.0	data	0.0	1.0	NaN	NaN	
11		5.5	data	0.0	1.0	NaN	NaN	

(continues on next page)

(continued from previous page)

12	6.0	data	0.0	1.0	NaN	NaN
13	6.5	data	0.0	1.0	NaN	NaN
14	7.0	data	0.0	1.0	NaN	NaN
15	7.5	data	0.0	1.0	NaN	NaN
16	8.0	data	0.0	1.0	NaN	NaN
17	8.5	data	0.0	1.0	NaN	NaN
18	9.0	data	0.0	1.0	NaN	NaN
19	9.5	data	0.0	1.0	NaN	NaN
20	10.0	data	0.0	1.0	NaN	NaN

	observable_x1	observable_x2	observable_x3	observable_x1_scaled	\
0	0.187985	-0.080848	0.305322	-1.554071	
1	0.396891	0.363006	-0.700034	-0.164218	
2	0.554254	0.245263	-0.328448	2.271354	
3	-1.160294	-0.408645	-0.704137	-0.456845	
4	3.015808	0.042902	0.513711	-0.480673	
5	1.957870	0.392603	1.835236	1.949022	
6	0.250335	2.475974	-0.010817	-0.570774	
7	0.685815	1.402648	-0.813082	1.534414	
8	1.010868	0.233904	-0.438445	0.383531	
9	0.415664	1.940166	1.571919	1.999110	
10	0.798758	-0.628757	0.288368	1.414603	
11	-0.453344	0.398597	-1.035944	2.123599	
12	-0.334512	1.547925	-1.185508	0.869001	
13	1.809684	0.087959	0.508957	1.063831	
14	1.861750	0.588173	0.661888	-0.347133	
15	1.045739	1.850249	0.561389	-0.084620	
16	0.395499	1.181430	-2.304970	2.106010	
17	1.294998	-0.237728	-0.869179	2.492656	
18	-0.479066	0.878030	-0.354968	0.366001	
19	1.536107	1.134127	0.954165	-0.059381	
20	1.035979	1.466212	0.611306	0.766315	

	observable_x2_offsetted	observable_x1withsigma	observable_x1_std	\
0	4.331282	0.883330	1.0	
1	3.830254	1.468191	1.0	
2	4.537019	0.136057	1.0	
3	2.966372	-0.547702	1.0	
4	5.854082	1.414807	1.0	
5	5.170209	0.150689	1.0	
6	3.666161	1.051665	1.0	
7	4.557583	1.624524	1.0	
8	3.474711	-0.578121	1.0	
9	4.376968	1.331552	1.0	
10	3.081891	1.266366	1.0	
11	3.393821	1.067552	1.0	
12	2.398599	-0.229404	1.0	
13	2.885707	1.071770	1.0	
14	2.704528	1.040706	1.0	
15	4.134466	1.207100	1.0	
16	3.677478	-0.159304	1.0	
17	1.921124	1.170464	1.0	

(continues on next page)



(continued from previous page)

18	3.586544	0.556909	1.0
19	1.736947	0.961276	1.0
20	4.471560	2.161286	1.0
	observable_x2_std	observable_x3_std	observable_x1_scaled_std \
0	1.0	1.0	1.0
1	1.0	1.0	1.0
2	1.0	1.0	1.0
3	1.0	1.0	1.0
4	1.0	1.0	1.0
5	1.0	1.0	1.0
6	1.0	1.0	1.0
7	1.0	1.0	1.0
8	1.0	1.0	1.0
9	1.0	1.0	1.0
10	1.0	1.0	1.0
11	1.0	1.0	1.0
12	1.0	1.0	1.0
13	1.0	1.0	1.0
14	1.0	1.0	1.0
15	1.0	1.0	1.0
16	1.0	1.0	1.0
17	1.0	1.0	1.0
18	1.0	1.0	1.0
19	1.0	1.0	1.0
20	1.0	1.0	1.0
	observable_x2_offsetted_std	observable_x1withsigma_std	
0	1.0	NaN	
1	1.0	NaN	
2	1.0	NaN	
3	1.0	NaN	
4	1.0	NaN	
5	1.0	NaN	
6	1.0	NaN	
7	1.0	NaN	
8	1.0	NaN	
9	1.0	NaN	
10	1.0	NaN	
11	1.0	NaN	
12	1.0	NaN	
13	1.0	NaN	
14	1.0	NaN	
15	1.0	NaN	
16	1.0	NaN	
17	1.0	NaN	
18	1.0	NaN	
19	1.0	NaN	
20	1.0	NaN	

```
[26]: # from the exported dataframe, we can actually reconstruct a copy of the ExpData instance
reconstructed_edata = amici.getEdataFromDataFrame(model, df)
```

```
[27]: # look at the States in rdata as DataFrame
```

```
amici.getResidualsAsDataFrame(model, [edata], [rdata])
```

```
[27]:
```

	condition_id	time	t_presim	k0	k0_preeq	k0_presim	observable_x1	\
0	NaN	0.0	0.0	1.0	NaN	NaN	0.087985	
1	NaN	0.5	0.0	1.0	NaN	NaN	0.142476	
2	NaN	1.0	0.0	1.0	NaN	NaN	0.025819	
3	NaN	1.5	0.0	1.0	NaN	NaN	1.730693	
4	NaN	2.0	0.0	1.0	NaN	NaN	2.455274	
5	NaN	2.5	0.0	1.0	NaN	NaN	1.404814	
6	NaN	3.0	0.0	1.0	NaN	NaN	0.296536	
7	NaN	3.5	0.0	1.0	NaN	NaN	0.144455	
8	NaN	4.0	0.0	1.0	NaN	NaN	0.474587	
9	NaN	4.5	0.0	1.0	NaN	NaN	0.115874	
10	NaN	5.0	0.0	1.0	NaN	NaN	0.271666	
11	NaN	5.5	0.0	1.0	NaN	NaN	0.976259	
12	NaN	6.0	0.0	1.0	NaN	NaN	0.853501	
13	NaN	6.5	0.0	1.0	NaN	NaN	1.294385	
14	NaN	7.0	0.0	1.0	NaN	NaN	1.349920	
15	NaN	7.5	0.0	1.0	NaN	NaN	0.537171	
16	NaN	8.0	0.0	1.0	NaN	NaN	0.110001	
17	NaN	8.5	0.0	1.0	NaN	NaN	0.792383	
18	NaN	9.0	0.0	1.0	NaN	NaN	0.978968	
19	NaN	9.5	0.0	1.0	NaN	NaN	1.038758	
20	NaN	10.0	0.0	1.0	NaN	NaN	0.541030	

	observable_x2	observable_x3	observable_x1_scaled	\
0	0.480848	0.394678	1.754071	
1	0.321673	0.891525	1.242952	
2	0.488024	0.424872	1.111209	
3	1.139297	0.780213	1.597644	
4	0.672934	0.444017	1.601742	
5	0.306148	1.768935	0.842910	
6	1.794011	0.074550	1.664515	
7	0.736539	0.874588	0.451694	
8	0.417398	0.497940	0.689029	
9	1.302651	1.514266	0.936034	
10	1.253438	0.232408	0.360421	
11	0.214135	1.090344	1.077770	
12	0.946322	1.238468	0.168978	
13	0.503270	0.457328	0.033233	
14	0.006618	0.611489	1.370793	
15	1.277720	0.512130	1.101755	
16	0.617327	2.353174	1.095009	
17	0.793962	0.916403	1.487426	
18	0.329149	0.401283	0.633802	
19	0.592119	0.908695	1.054080	
20	0.930631	0.566621	0.223583	

	observable_x2_offsetted	observable_x1withsigma
0	0.931282	7.833304
1	0.145575	9.288235
2	0.803731	4.440159

(continues on next page)

(continued from previous page)

3	0.764280	11.181017
4	2.138246	8.542727
5	1.471458	4.023665
6	0.015802	5.047948
7	0.891474	10.831640
8	0.176591	11.144013
9	0.739453	8.000134
10	0.542791	7.392745
11	0.218912	5.446376
12	1.203004	7.483934
13	0.705522	5.564709
14	0.877027	5.288755
15	0.561937	6.985322
16	0.113374	6.648047
17	1.635110	6.678489
18	0.037662	0.570074
19	1.805061	4.639264
20	0.935979	16.663370

```
[28]: # look at the Observables in rdata as DataFrame
      amici.getSimulationObservablesAsDataFrame(model, [edata], [rdata])
```

```
[28]:
```

	condition_id	time	datatype	t_presim	k0	k0_preeq	k0_presim	\
0		0.0	simulation	0.0	1.0	NaN	NaN	
1		0.5	simulation	0.0	1.0	NaN	NaN	
2		1.0	simulation	0.0	1.0	NaN	NaN	
3		1.5	simulation	0.0	1.0	NaN	NaN	
4		2.0	simulation	0.0	1.0	NaN	NaN	
5		2.5	simulation	0.0	1.0	NaN	NaN	
6		3.0	simulation	0.0	1.0	NaN	NaN	
7		3.5	simulation	0.0	1.0	NaN	NaN	
8		4.0	simulation	0.0	1.0	NaN	NaN	
9		4.5	simulation	0.0	1.0	NaN	NaN	
10		5.0	simulation	0.0	1.0	NaN	NaN	
11		5.5	simulation	0.0	1.0	NaN	NaN	
12		6.0	simulation	0.0	1.0	NaN	NaN	
13		6.5	simulation	0.0	1.0	NaN	NaN	
14		7.0	simulation	0.0	1.0	NaN	NaN	
15		7.5	simulation	0.0	1.0	NaN	NaN	
16		8.0	simulation	0.0	1.0	NaN	NaN	
17		8.5	simulation	0.0	1.0	NaN	NaN	
18		9.0	simulation	0.0	1.0	NaN	NaN	
19		9.5	simulation	0.0	1.0	NaN	NaN	
20		10.0	simulation	0.0	1.0	NaN	NaN	

	observable_x1	observable_x2	observable_x3	observable_x1_scaled	\
0	0.100000	0.400000	0.700000	0.200000	
1	0.539367	0.684679	0.191491	1.078734	
2	0.580072	0.733287	0.096424	1.160145	
3	0.570399	0.730652	0.076076	1.140799	
4	0.560535	0.715836	0.069694	1.121069	
5	0.553056	0.698751	0.066301	1.106112	

(continues on next page)

(continued from previous page)

6	0.546871	0.681963	0.063733	1.093741
7	0.541360	0.666109	0.061506	1.082720
8	0.536280	0.651302	0.059495	1.072561
9	0.531538	0.637515	0.057653	1.063076
10	0.527091	0.624681	0.055960	1.054183
11	0.522914	0.612733	0.054400	1.045829
12	0.518989	0.601603	0.052960	1.037978
13	0.515299	0.591229	0.051629	1.030598
14	0.511830	0.581555	0.050399	1.023660
15	0.508568	0.572529	0.049259	1.017136
16	0.505500	0.564103	0.048203	1.011000
17	0.502615	0.556234	0.047224	1.005231
18	0.499902	0.548881	0.046315	0.999804
19	0.497350	0.542008	0.045471	0.994700
20	0.494949	0.535581	0.044686	0.989898

	observable_x2_offsetted	observable_x1withsigma	observable_x1_std	\
0	3.400000	0.100000	1.0	
1	3.684679	0.539367	1.0	
2	3.733287	0.580072	1.0	
3	3.730652	0.570399	1.0	
4	3.715836	0.560535	1.0	
5	3.698751	0.553056	1.0	
6	3.681963	0.546871	1.0	
7	3.666109	0.541360	1.0	
8	3.651302	0.536280	1.0	
9	3.637515	0.531538	1.0	
10	3.624681	0.527091	1.0	
11	3.612733	0.522914	1.0	
12	3.601603	0.518989	1.0	
13	3.591229	0.515299	1.0	
14	3.581555	0.511830	1.0	
15	3.572529	0.508568	1.0	
16	3.564103	0.505500	1.0	
17	3.556234	0.502615	1.0	
18	3.548881	0.499902	1.0	
19	3.542008	0.497350	1.0	
20	3.535581	0.494949	1.0	

	observable_x2_std	observable_x3_std	observable_x1_scaled_std	\
0	1.0	1.0	1.0	
1	1.0	1.0	1.0	
2	1.0	1.0	1.0	
3	1.0	1.0	1.0	
4	1.0	1.0	1.0	
5	1.0	1.0	1.0	
6	1.0	1.0	1.0	
7	1.0	1.0	1.0	
8	1.0	1.0	1.0	
9	1.0	1.0	1.0	
10	1.0	1.0	1.0	
11	1.0	1.0	1.0	

(continues on next page)

(continued from previous page)

12	1.0	1.0	1.0
13	1.0	1.0	1.0
14	1.0	1.0	1.0
15	1.0	1.0	1.0
16	1.0	1.0	1.0
17	1.0	1.0	1.0
18	1.0	1.0	1.0
19	1.0	1.0	1.0
20	1.0	1.0	1.0
	observable_x2_offsetted_std	observable_x1withsigma_std	
0	1.0	0.1	
1	1.0	0.1	
2	1.0	0.1	
3	1.0	0.1	
4	1.0	0.1	
5	1.0	0.1	
6	1.0	0.1	
7	1.0	0.1	
8	1.0	0.1	
9	1.0	0.1	
10	1.0	0.1	
11	1.0	0.1	
12	1.0	0.1	
13	1.0	0.1	
14	1.0	0.1	
15	1.0	0.1	
16	1.0	0.1	
17	1.0	0.1	
18	1.0	0.1	
19	1.0	0.1	
20	1.0	0.1	

```
[29]: # look at the States in rdata as DataFrame
      amici.getSimulationStatesAsDataFrame(model, [edata], [rdata])
```

```
[29]: condition_id  time  t_presim  k0  k0_preeq  k0_presim      x1      x2  \
0            0.0    0.0    1.0      NaN      NaN  0.100000  0.400000
1            0.5    0.0    1.0      NaN      NaN  0.539367  0.684679
2            1.0    0.0    1.0      NaN      NaN  0.580072  0.733287
3            1.5    0.0    1.0      NaN      NaN  0.570399  0.730652
4            2.0    0.0    1.0      NaN      NaN  0.560535  0.715836
5            2.5    0.0    1.0      NaN      NaN  0.553056  0.698751
6            3.0    0.0    1.0      NaN      NaN  0.546871  0.681963
7            3.5    0.0    1.0      NaN      NaN  0.541360  0.666109
8            4.0    0.0    1.0      NaN      NaN  0.536280  0.651302
9            4.5    0.0    1.0      NaN      NaN  0.531538  0.637515
10           5.0    0.0    1.0      NaN      NaN  0.527091  0.624681
11           5.5    0.0    1.0      NaN      NaN  0.522914  0.612733
12           6.0    0.0    1.0      NaN      NaN  0.518989  0.601603
13           6.5    0.0    1.0      NaN      NaN  0.515299  0.591229
14           7.0    0.0    1.0      NaN      NaN  0.511830  0.581555
```

(continues on next page)

(continued from previous page)

15	7.5	0.0	1.0	NaN	NaN	0.508568	0.572529
16	8.0	0.0	1.0	NaN	NaN	0.505500	0.564103
17	8.5	0.0	1.0	NaN	NaN	0.502615	0.556234
18	9.0	0.0	1.0	NaN	NaN	0.499902	0.548881
19	9.5	0.0	1.0	NaN	NaN	0.497350	0.542008
20	10.0	0.0	1.0	NaN	NaN	0.494949	0.535581

x3							
0	0.700000						
1	0.191491						
2	0.096424						
3	0.076076						
4	0.069694						
5	0.066301						
6	0.063733						
7	0.061506						
8	0.059495						
9	0.057653						
10	0.055960						
11	0.054400						
12	0.052960						
13	0.051629						
14	0.050399						
15	0.049259						
16	0.048203						
17	0.047224						
18	0.046315						
19	0.045471						
20	0.044686						

### 10.2.3 Using PEtab

This notebook illustrates how to use [PEtab](#) with AMICI.

```
[1]: from amici.petab_import import import_petab_problem
    from amici.petab_objective import simulate_petab
    import petab

    import os
```

We use an example model from the [benchmark collection](#):

```
[2]: !git clone --depth 1 https://github.com/Benchmarking-Initiative/Benchmark-Models-PEtab.
    ↪ git tmp/benchmark-models || (cd tmp/benchmark-models && git pull)

Cloning into 'tmp/benchmark-models'...
remote: Enumerating objects: 142, done.
remote: Counting objects: 100% (142/142), done.
remote: Compressing objects: 100% (122/122), done.
remote: Total 142 (delta 41), reused 104 (delta 18), pack-reused 0
```

(continues on next page)

(continued from previous page)

```
Receiving objects: 100% (142/142), 648.29 KiB | 1.23 MiB/s, done.
Resolving deltas: 100% (41/41), done.
```

```
[3]: folder_base = "tmp/benchmark-models/Benchmark-Models/"
     !ls -l $folder_base

total 68
drwxr-xr-x 2 yannik yannik 4096 Mär 17 15:27 Alkan_SciSignal2018
drwxr-xr-x 2 yannik yannik 4096 Mär 17 15:27 Beer_MolBioSystems2014
drwxr-xr-x 2 yannik yannik 4096 Mär 17 15:27 Boehm_JProteomeRes2014
drwxr-xr-x 2 yannik yannik 4096 Mär 17 15:27 Borghans_BiophysChem1997
drwxr-xr-x 2 yannik yannik 4096 Mär 17 15:27 Brannmark_JBC2010
drwxr-xr-x 2 yannik yannik 4096 Mär 17 15:27 Bruno_JExpBio2016
-rwxr-xr-x 1 yannik yannik  654 Mär 17 15:27 checkBenchmarkModels.py
drwxr-xr-x 2 yannik yannik 4096 Mär 17 15:27 Chen_MSB2009
drwxr-xr-x 2 yannik yannik 4096 Mär 17 15:27 Crauste_CellSystems2017
drwxr-xr-x 2 yannik yannik 4096 Mär 17 15:27 Elowitz_Nature2000
drwxr-xr-x 2 yannik yannik 4096 Mär 17 15:27 Fiedler_BMC2016
drwxr-xr-x 2 yannik yannik 4096 Mär 17 15:27 Fujita_SciSignal2010
drwxr-xr-x 2 yannik yannik 4096 Mär 17 15:27 Perelson_Science1996
drwxr-xr-x 2 yannik yannik 4096 Mär 17 15:27 Rahman_MBS2016
drwxr-xr-x 2 yannik yannik 4096 Mär 17 15:27 Sneyd_PNAS2002
drwxr-xr-x 2 yannik yannik 4096 Mär 17 15:27 Weber_BMC2015
drwxr-xr-x 2 yannik yannik 4096 Mär 17 15:27 Zheng_PNAS2012
```

We import a model to PETab from a provided yaml file:

```
[4]: model_name = "Boehm_JProteomeRes2014"
     yaml_file = os.path.join(folder_base, model_name, model_name + ".yaml")
     petab_problem = petab.Problem.from_yaml(yaml_file)
```

Next, we import the model to amici, compile it and obtain a function handle:

```
[5]: amici_model = import_petab_problem(petab_problem)

2020-03-17 15:27:27.586 - amici.petab_import - INFO - Importing model ...
2020-03-17 15:27:27.593 - amici.petab_import - INFO - Model name is 'Boehm_
↳ JProteomeRes2014'. Writing model code to '/home/yannik/amici/python/examples/amici_
↳ models/Boehm_JProteomeRes2014'.
2020-03-17 15:27:27.598 - amici.petab_import - INFO - Species: 8
2020-03-17 15:27:27.599 - amici.petab_import - INFO - Global parameters: 9
2020-03-17 15:27:27.599 - amici.petab_import - INFO - Reactions: 9
2020-03-17 15:27:27.715 - amici.petab_import - INFO - Observables: 3
2020-03-17 15:27:27.715 - amici.petab_import - INFO - Sigmas: 3
2020-03-17 15:27:27.722 - amici.petab_import - DEBUG - Adding output parameters to model:
↳ OrderedDict([('noiseParameter1_pSTAT5A_rel', None), ('noiseParameter1_pSTAT5B_rel',
↳ None), ('noiseParameter1_rSTAT5A_rel', None)])
2020-03-17 15:27:27.725 - amici.petab_import - DEBUG - Condition table: (1, 1)
2020-03-17 15:27:27.726 - amici.petab_import - DEBUG - Fixed parameters are []
2020-03-17 15:27:27.728 - amici.petab_import - INFO - Overall fixed parameters: 0
2020-03-17 15:27:27.729 - amici.petab_import - INFO - Variable parameters: 12
2020-03-17 15:27:27.735 - amici.sbml_import - INFO - Finished processing SBML parameters.
↳ (1.25E-03s)
```

(continues on next page)

(continued from previous page)

```

2020-03-17 15:27:27.749 - amici.sbml_import - INFO - Finished processing SBML species      ↵
↵      (1.26E-02s)
2020-03-17 15:27:27.829 - amici.sbml_import - INFO - Finished processing SBML reactions  ↵
↵      (7.41E-02s)
2020-03-17 15:27:27.833 - amici.sbml_import - INFO - Finished processing SBML ↵
↵ compartments      (4.23E-04s)
2020-03-17 15:27:27.898 - amici.sbml_import - INFO - Finished processing SBML rules    ↵
↵      (6.47E-02s)
2020-03-17 15:27:28.012 - amici.sbml_import - INFO - Finished processing SBML ↵
↵ observables      (6.77E-02s)
2020-03-17 15:27:28.139 - amici.ode_export - INFO - Finished writing J.cpp              ↵
↵      (1.14E-01s)
2020-03-17 15:27:28.160 - amici.ode_export - INFO - Finished writing JB.cpp            ↵
↵      (2.04E-02s)
2020-03-17 15:27:28.167 - amici.ode_export - INFO - Finished writing JDiag.cpp          ↵
↵      (6.41E-03s)
2020-03-17 15:27:28.187 - amici.ode_export - INFO - Finished writing JSparse.cpp        ↵
↵      (1.91E-02s)
2020-03-17 15:27:28.217 - amici.ode_export - INFO - Finished writing JSparseB.cpp      ↵
↵      (2.73E-02s)
2020-03-17 15:27:28.236 - amici.ode_export - INFO - Finished writing Jy.cpp              ↵
↵      (1.65E-02s)
2020-03-17 15:27:28.344 - amici.ode_export - INFO - Finished writing dJydsigmay.cpp      ↵
↵      (1.07E-01s)
2020-03-17 15:27:28.389 - amici.ode_export - INFO - Finished writing dJydy.cpp            ↵
↵      (3.99E-02s)
2020-03-17 15:27:28.466 - amici.ode_export - INFO - Finished writing dwdp.cpp              ↵
↵      (7.61E-02s)
2020-03-17 15:27:28.473 - amici.ode_export - INFO - Finished writing dwdx.cpp              ↵
↵      (5.87E-03s)
2020-03-17 15:27:28.497 - amici.ode_export - INFO - Finished writing dxdotdw.cpp          ↵
↵      (2.32E-02s)
2020-03-17 15:27:28.533 - amici.ode_export - INFO - Finished writing dxdotdp_explicit. ↵
↵ cpp      (3.38E-02s)
2020-03-17 15:27:28.756 - amici.ode_export - INFO - Finished writing dydx.cpp              ↵
↵      (1.98E-01s)
2020-03-17 15:27:28.910 - amici.ode_export - INFO - Finished writing dydp.cpp              ↵
↵      (1.53E-01s)
2020-03-17 15:27:28.926 - amici.ode_export - INFO - Finished writing dsigmaydp.cpp          ↵
↵      (1.40E-02s)
2020-03-17 15:27:28.931 - amici.ode_export - INFO - Finished writing sigmay.cpp            ↵
↵      (2.46E-03s)
2020-03-17 15:27:28.950 - amici.ode_export - INFO - Finished writing w.cpp                ↵
↵      (1.55E-02s)
2020-03-17 15:27:28.967 - amici.ode_export - INFO - Finished writing x0.cpp              ↵
↵      (1.57E-02s)
2020-03-17 15:27:28.975 - amici.ode_export - INFO - Finished writing x0_fixedParameters. ↵
↵ cpp      (4.78E-03s)
2020-03-17 15:27:29.027 - amici.ode_export - INFO - Finished writing sx0.cpp              ↵
↵      (5.01E-02s)
2020-03-17 15:27:29.069 - amici.ode_export - INFO - Finished writing sx0_fixedParameters. ↵
↵ cpp      (3.14E-02s)

```

(continues on next page)



(continued from previous page)

```

2020-03-17 15:27:29.104 - amici.ode_export - INFO - Finished writing xdot.cpp
↳ (3.43E-02s)
2020-03-17 15:27:29.129 - amici.ode_export - INFO - Finished writing y.cpp
↳ (2.16E-02s)
2020-03-17 15:27:29.136 - amici.ode_export - INFO - Finished writing x_rdata.cpp
↳ (4.95E-03s)
2020-03-17 15:27:29.138 - amici.ode_export - INFO - Finished writing total_cl.cpp
↳ (6.59E-04s)
2020-03-17 15:27:29.147 - amici.ode_export - INFO - Finished writing x_solver.cpp
↳ (7.72E-03s)
2020-03-17 15:27:29.166 - amici.ode_export - INFO - Finished generating cpp code
↳ (1.14E+00s)
2020-03-17 15:27:46.200 - amici.ode_export - INFO - Finished compiling cpp code
↳ (1.70E+01s)
2020-03-17 15:27:46.204 - amici.petab_import - INFO - Finished Importing PETab model
↳ (1.86E+01s)
2020-03-17 15:27:46.209 - amici.petab_import - INFO - Successfully loaded model Boehm_
↳ JProteomeRes2014 from /home/yannik/amici/python/examples/amici_models/Boehm_
↳ JProteomeRes2014.

```

running build\_ext

building 'Boehm\_JProteomeRes2014\_Boehm\_JProteomeRes2014' extension

swigging swig/Boehm\_JProteomeRes2014.i to swig/Boehm\_JProteomeRes2014\_wrap.cpp

swig -python -c++ -modern -outdir Boehm\_JProteomeRes2014 -I/home/yannik/amici/python/

↳ sdist/amici/swig -I/home/yannik/amici/python/sdist/amici/include -o swig/Boehm\_

↳ JProteomeRes2014\_wrap.cpp swig/Boehm\_JProteomeRes2014.i

creating build

creating build/temp.linux-x86\_64-3.7

creating build/temp.linux-x86\_64-3.7/swig

gcc -pthread -B /home/yannik/anaconda3/compiler\_compat -Wl,--sysroot=/ -Wsign-compare -

↳ DDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/

↳ examples/amici\_models/Boehm\_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/

↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/

↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/

↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/

↳ include/python3.7m -c swig/Boehm\_JProteomeRes2014\_wrap.cpp -o build/temp.linux-x86\_64-

↳ 3.7/swig/Boehm\_JProteomeRes2014\_wrap.o -std=c++14

cclplus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not

↳ for C++

gcc -pthread -B /home/yannik/anaconda3/compiler\_compat -Wl,--sysroot=/ -Wsign-compare -

↳ DDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/

↳ examples/amici\_models/Boehm\_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/

↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/

↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/

↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/

↳ include/python3.7m -c Boehm\_JProteomeRes2014\_dxdotdw.cpp -o build/temp.linux-x86\_64-3.

↳ 7/Boehm\_JProteomeRes2014\_dxdotdw.o -std=c++14

cclplus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not

↳ for C++

gcc -pthread -B /home/yannik/anaconda3/compiler\_compat -Wl,--sysroot=/ -Wsign-compare -

↳ DDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/

↳ examples/amici\_models/Boehm\_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/

↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/

↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/

↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/

↳ include/python3.7m -c Boehm\_JProteomeRes2014\_total\_cl.cpp -o build/temp.linux-x86\_64-

↳ 7/Boehm\_JProteomeRes2014\_total\_cl.o -std=c++14

(continued from previous page)

```

cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_x_rdata.cpp -o build/temp.linux-x86_64-3.
↳ 7/Boehm_JProteomeRes2014_x_rdata.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dxdotdp_implicit_colptrs.cpp -o build/
↳ temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_dxdotdp_implicit_colptrs.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dsigmaydp.cpp -o build/temp.linux-x86_64-
↳ 3.7/Boehm_JProteomeRes2014_dsigmaydp.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_y.cpp -o build/temp.linux-x86_64-3.7/
↳ Boehm_JProteomeRes2014_y.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dydp.cpp -o build/temp.linux-x86_64-3.7/
↳ Boehm_JProteomeRes2014_dydp.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++

```

(continues on next page)

(continued from previous page)

```

gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_w.cpp -o build/temp.linux-x86_64-3.7/
↳ Boehm_JProteomeRes2014_w.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_JSparseB_rowvals.cpp -o build/temp.linux-
↳ x86_64-3.7/Boehm_JProteomeRes2014_JSparseB_rowvals.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dxdotdw_rowvals.cpp -o build/temp.linux-
↳ x86_64-3.7/Boehm_JProteomeRes2014_dxdotdw_rowvals.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dwdx_rowvals.cpp -o build/temp.linux-x86_
↳ 64-3.7/Boehm_JProteomeRes2014_dwdx_rowvals.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_x0.cpp -o build/temp.linux-x86_64-3.7/
↳ Boehm_JProteomeRes2014_x0.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dwdx.cpp -o build/temp.linux-x86_64-3.7/
↳ Boehm_JProteomeRes2014_dwdx.o -std=c++14

```

(continued from previous page)

```

cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dJdy_colptrs.cpp -o build/temp.linux-x86_
↳ 64-3.7/Boehm_JProteomeRes2014_dJdy_colptrs.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_JSparseB.cpp -o build/temp.linux-x86_64-3.
↳ 7/Boehm_JProteomeRes2014_JSparseB.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_JSparseB_colptrs.cpp -o build/temp.linux-
↳ x86_64-3.7/Boehm_JProteomeRes2014_JSparseB_colptrs.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dxdotdp_explicit_colptrs.cpp -o build/
↳ temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_dxdotdp_explicit_colptrs.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_sx0_fixedParameters.cpp -o build/temp.
↳ linux-x86_64-3.7/Boehm_JProteomeRes2014_sx0_fixedParameters.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++

```

(continues on next page)

(continued from previous page)

```

gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_JSparse_rowvals.cpp -o build/temp.linux-
↳ x86_64-3.7/Boehm_JProteomeRes2014_JSparse_rowvals.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dxdotdp_explicit.cpp -o build/temp.linux-
↳ x86_64-3.7/Boehm_JProteomeRes2014_dxdotdp_explicit.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dJydy.cpp -o build/temp.linux-x86_64-3.7/
↳ Boehm_JProteomeRes2014_dJydy.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dwdp_colptrs.cpp -o build/temp.linux-x86_
↳ 64-3.7/Boehm_JProteomeRes2014_dwdp_colptrs.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_x0_fixedParameters.cpp -o build/temp.
↳ linux-x86_64-3.7/Boehm_JProteomeRes2014_x0_fixedParameters.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dxdotdw_colptrs.cpp -o build/temp.linux-
↳ x86_64-3.7/Boehm_JProteomeRes2014_dxdotdw_colptrs.o -std=c++14

```



(continued from previous page)

```

cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dJydsigmay.cpp -o build/temp.linux-x86_64-
↳ 3.7/Boehm_JProteomeRes2014_dJydsigmay.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dxdotdp_implicit_rowvals.cpp -o build/
↳ temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_dxdotdp_implicit_rowvals.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dwdp.cpp -o build/temp.linux-x86_64-3.7/
↳ Boehm_JProteomeRes2014_dwdp.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_sx0.cpp -o build/temp.linux-x86_64-3.7/
↳ Boehm_JProteomeRes2014_sx0.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_JB.cpp -o build/temp.linux-x86_64-3.7/
↳ Boehm_JProteomeRes2014_JB.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++

```

(continues on next page)

(continued from previous page)

```

gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dwdx_colptrs.cpp -o build/temp.linux-x86_
↳ 64-3.7/Boehm_JProteomeRes2014_dwdx_colptrs.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c wrapfunctions.cpp -o build/temp.linux-x86_64-3.7/wrapfunctions.o
↳ -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_x_solver.cpp -o build/temp.linux-x86_64-3.
↳ 7/Boehm_JProteomeRes2014_x_solver.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_JSparse.cpp -o build/temp.linux-x86_64-3.
↳ 7/Boehm_JProteomeRes2014_JSparse.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_xdot.cpp -o build/temp.linux-x86_64-3.7/
↳ Boehm_JProteomeRes2014_xdot.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dJydy_rowvals.cpp -o build/temp.linux-x86_
↳ 64-3.7/Boehm_JProteomeRes2014_dJydy_rowvals.o -std=c++14

```

(continued from previous page)

```

cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dwdp_rowvals.cpp -o build/temp.linux-x86_
↳ 64-3.7/Boehm_JProteomeRes2014_dwdp_rowvals.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_JSparse_colptrs.cpp -o build/temp.linux-
↳ x86_64-3.7/Boehm_JProteomeRes2014_JSparse_colptrs.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_J.cpp -o build/temp.linux-x86_64-3.7/
↳ Boehm_JProteomeRes2014_J.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dydx.cpp -o build/temp.linux-x86_64-3.7/
↳ Boehm_JProteomeRes2014_dydx.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_JDiag.cpp -o build/temp.linux-x86_64-3.7/
↳ Boehm_JProteomeRes2014_JDiag.o -std=c++14
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++

```

(continues on next page)



(continued from previous page)

```
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_Jy.cpp -o build/temp.linux-x86_64-3.7/
↳ Boehm_JProteomeRes2014_Jy.o -std=c++14
```

```
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
```

```
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_sigmay.cpp -o build/temp.linux-x86_64-3.7/
↳ Boehm_JProteomeRes2014_sigmay.o -std=c++14
```

```
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
```

```
gcc -pthread -B /home/yannik/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -
↳ DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prototypes -fPIC -I/home/yannik/amici/python/
↳ examples/amici_models/Boehm_JProteomeRes2014 -I/home/yannik/amici/python/sdist/amici/
↳ include -I/home/yannik/amici/python/sdist/amici/ThirdParty/gsl -I/home/yannik/amici/
↳ python/sdist/amici/ThirdParty/sundials/include -I/home/yannik/amici/python/sdist/amici/
↳ ThirdParty/SuiteSparse/include -I/usr/include/hdf5/serial -I/home/yannik/anaconda3/
↳ include/python3.7m -c Boehm_JProteomeRes2014_dxdotdp_explicit_rowvals.cpp -o build/
↳ temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_dxdotdp_explicit_rowvals.o -std=c++14
```

```
cc1plus: warning: command line option '-Wstrict-prototypes' is valid for C/ObjC but not
↳ for C++
```

```
g++ -pthread -shared -B /home/yannik/anaconda3/compiler_compat -L/home/yannik/anaconda3/
↳ lib -Wl,-rpath=/home/yannik/anaconda3/lib -Wl,--no-as-needed -Wl,--sysroot=/ build/
↳ temp.linux-x86_64-3.7/swig/Boehm_JProteomeRes2014_wrap.o build/temp.linux-x86_64-3.7/
↳ Boehm_JProteomeRes2014_dxdotdw.o build/temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_
↳ total_cl.o build/temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_x_rdata.o build/temp.
↳ linux-x86_64-3.7/Boehm_JProteomeRes2014_dxdotdp_implicit_colptrs.o build/temp.linux-
↳ x86_64-3.7/Boehm_JProteomeRes2014_dsigmaydp.o build/temp.linux-x86_64-3.7/Boehm_
↳ JProteomeRes2014_y.o build/temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_dydp.o build/
↳ temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_w.o build/temp.linux-x86_64-3.7/Boehm_
↳ JProteomeRes2014_JSparseB_rowvals.o build/temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_
↳ dxdotdw_rowvals.o build/temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_dwdx_rowvals.o
↳ build/temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_x0.o build/temp.linux-x86_64-3.7/
↳ Boehm_JProteomeRes2014_dwdx.o build/temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_dJydy_
↳ colptrs.o build/temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_JSparseB.o build/temp.
↳ linux-x86_64-3.7/Boehm_JProteomeRes2014_JSparseB_colptrs.o build/temp.linux-x86_64-3.7/
↳ Boehm_JProteomeRes2014_dxdotdp_explicit_colptrs.o build/temp.linux-x86_64-3.7/Boehm_
↳ JProteomeRes2014_sx0_fixedParameters.o build/temp.linux-x86_64-3.7/Boehm_
↳ JProteomeRes2014_JSparse_rowvals.o build/temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_
↳ dxdotdp_explicit.o build/temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_dJydy.o build/
↳ temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_dwdp_colptrs.o build/temp.linux-x86_64-3.
↳ 7/Boehm_JProteomeRes2014_x0_fixedParameters.o build/temp.linux-x86_64-3.7/Boehm_
↳ JProteomeRes2014_dxdotdw_colptrs.o build/temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_
↳ dJydsigmay.o build/temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_dxdotdp_implicit_
↳ rowvals.o build/temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_dwdp.o build/temp.linux-
↳ x86_64-3.7/Boehm_JProteomeRes2014_sx0.o build/temp.linux-x86_64-3.7/Boehm_
```

```
10.2. Examples 117
↳ JProteomeRes2014_JB.o build/temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_dwdx_colptrs.
↳ o build/temp.linux-x86_64-3.7/wrapfunctions.o build/temp.linux-x86_64-3.7/Boehm_
↳ JProteomeRes2014_x_solver.o build/temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_JSparse.
↳ o build/temp.linux-x86_64-3.7/Boehm_JProteomeRes2014_xdot.o build/temp.linux-x86_64-3.
```

(continued from previous page)

That's it. Now, we can use the model to perform simulations. For more involved purposes, consider using the objective function provided by `pyPESTO`. For simple simulations, a function `simulate_petab` is available:

```
[6]: simulate_petab(petab_problem, amici_model)
[6]: {'llh': -138.22199570334107,
      'sllh': None,
      'rdatas': [<amici.numpy.ReturnDataView at 0x7f030e1a0fd0>]}
```

This performs a simulation at the nominal parameters. Parameters can also be directly specified, both scaled and unscaled:

```
[7]: parameters = {
      x_id: x_val
      for x_id, x_val in zip(petab_problem.x_ids, petab_problem.x_nominal_scaled)
    }
    simulate_petab(
        petab_problem,
        amici_model,
        problem_parameters=parameters,
        scaled_parameters=True,
    )
[7]: {'llh': -138.22199570334107,
      'sllh': None,
      'rdatas': [<amici.numpy.ReturnDataView at 0x7f030e198590>]}
```

For further information, see the [documentation](#).

## 10.2.4 AMICI Python example “Experimental Conditions”

In this example we will explore some more options for the initialization of experimental conditions, including how to reset initial conditions based on changing values for `fixedParameters` as well as an additional presimulation phase on top of preequilibration. This notebook is expected to run from the `python/example_presimulation` directory.

```
[1]: # SBML model we want to import
      sbml_file = "model_presimulation.xml"
      # Name of the model that will also be the name of the python module
      model_name = "model_presimulation"
      # Directory to which the generated model code is written
      model_output_dir = model_name

      import libsbml
      import amici
      import amici.plotting
      import os
      import sys
      import importlib
      import numpy as np
```

(continues on next page)

(continued from previous page)

```
import pandas as pd
import matplotlib.pyplot as plt
from pprint import pprint
```

## Model Loading

Here we load a simple model of protein phosphorylation that can be inhibited by a drug. This model was created using PySB (see `createModel.py`)

```
[2]: sbml_reader = libsbml.SBMLReader()
sbml_doc = sbml_reader.readSBML(sbml_file)
sbml_model = sbml_doc.getModel()

print("Species:")
pprint([(s.getId(), s.getName()) for s in sbml_model.getListOfSpecies()])

print("\nReactions:")
for reaction in sbml_model.getListOfReactions():
    reactants = " + ".join(
        [
            "%s %s"
            % (
                int(r.getStoichiometry()) if r.getStoichiometry() > 1 else "",
                r.getSpecies(),
            )
            for r in reaction.getListOfReactants()
        ]
    )
    products = " + ".join(
        [
            "%s %s"
            % (
                int(r.getStoichiometry()) if r.getStoichiometry() > 1 else "",
                r.getSpecies(),
            )
            for r in reaction.getListOfProducts()
        ]
    )
    reversible = "<" if reaction.getReversible() else ""
    print(
        "%3s: %10s %1s->%10s\t\t[%s]"
        % (
            reaction.getName(),
            reactants,
            reversible,
            products,
            libsbml.formulaToL3String(reaction.getKineticLaw().getMath()),
        )
    )

print("Parameters:")
```

(continues on next page)

(continued from previous page)

```
pprint([(p.getId(), p.getName()) for p in sbml_model.getListOfParameters()])

Species:
[('__s0', "PROT(kin=None, drug=None, phospho='u')"),
 ('__s1', 'DRUG(bound=None)'),
 ('__s2', 'KIN(bound=None)'),
 ('__s3', "DRUG(bound=1) ._br_PROT(kin=None, drug=1, phospho='u')"),
 ('__s4', "KIN(bound=1) ._br_PROT(kin=1, drug=None, phospho='u')"),
 ('__s5', "PROT(kin=None, drug=None, phospho='p')")]

Reactions:
PROT_DRUG_bind: __s0 + __s1 <-> __s3          [-(koff_prot_drug * __s3) + kon_
↳prot_drug * __s0 * __s1]
PROT_KIN_bind: __s0 + __s2 -> __s4          [kon_prot_kin * __s0 * __s2]
PROT_KIN_phospho: __s4 -> __s2 + __s5      [kphospho_prot_kin * __s4]
PROT_dephospho: __s5 -> __s0              [kdephospho_prot * __s5]

Parameters:
[('initProt', 'initProt'),
 ('initDrug', 'initDrug'),
 ('initKin', 'initKin'),
 ('pPROT_obs', 'pPROT_obs'),
 ('PROT_0', 'PROT_0'),
 ('DRUG_0', 'DRUG_0'),
 ('KIN_0', 'KIN_0'),
 ('kon_prot_drug', 'kon_prot_drug'),
 ('koff_prot_drug', 'koff_prot_drug'),
 ('kon_prot_kin', 'kon_prot_kin'),
 ('kphospho_prot_kin', 'kphospho_prot_kin'),
 ('kdephospho_prot', 'kdephospho_prot'),
 ('__obs0', 'pPROT'),
 ('__obs1', 'tPROT')]
```

```
[3]: # Create an SbmlImporter instance for our SBML model
sbml_importer = amici.SbmlImporter(sbml_file)
```

For this example we want specify the initial drug and kinase concentrations as experimental conditions. Accordingly we specify them as `fixedParameters`. The meaning of `fixedParameters` is defined in the [Glossary](#), which we display here for convenience.

```
[4]: from IPython.display import IFrame

IFrame(
    "https://amici.readthedocs.io/en/latest/glossary.html#term-fixed-parameters",
    width=600,
    height=175,
)
```

```
[4]: <IPython.lib.display.IFrame at 0x10ce15b50>
```

```
[5]: fixedParameters = ["DRUG_0", "KIN_0"]
```

The SBML model specifies a single observable named `pPROT` which describes the fraction of phosphorylated Protein. We load this observable using `amici.assignmentRules2observables`.

```
[6]: # Retrieve model output names and formulae from AssignmentRules and remove the
      ↳ respective rules
observables = amici.assignmentRules2observables(
    sbml_importer.sbml, # the libsbml model object
    filter_function=lambda variable: variable.getName() == "pPROT",
)
print("Observables:")
pprint(observables)

Observables:
{'__obs0': {'formula': '___s5', 'name': 'pPROT'}}
```

Now the model is ready for compilation using `sbml2amici`. Note that we here pass `fixedParameters` as arguments to `constant_parameters`, which ensures that amici is aware that we want to have them as `fixedParameters`:

```
[7]: sbml_importer.sbml2amici(
    model_name,
    model_output_dir,
    verbose=False,
    observables=observables,
    constant_parameters=fixedParameters,
)
# load the generated module
model_module = amici.import_model_module(model_name, model_output_dir)
```

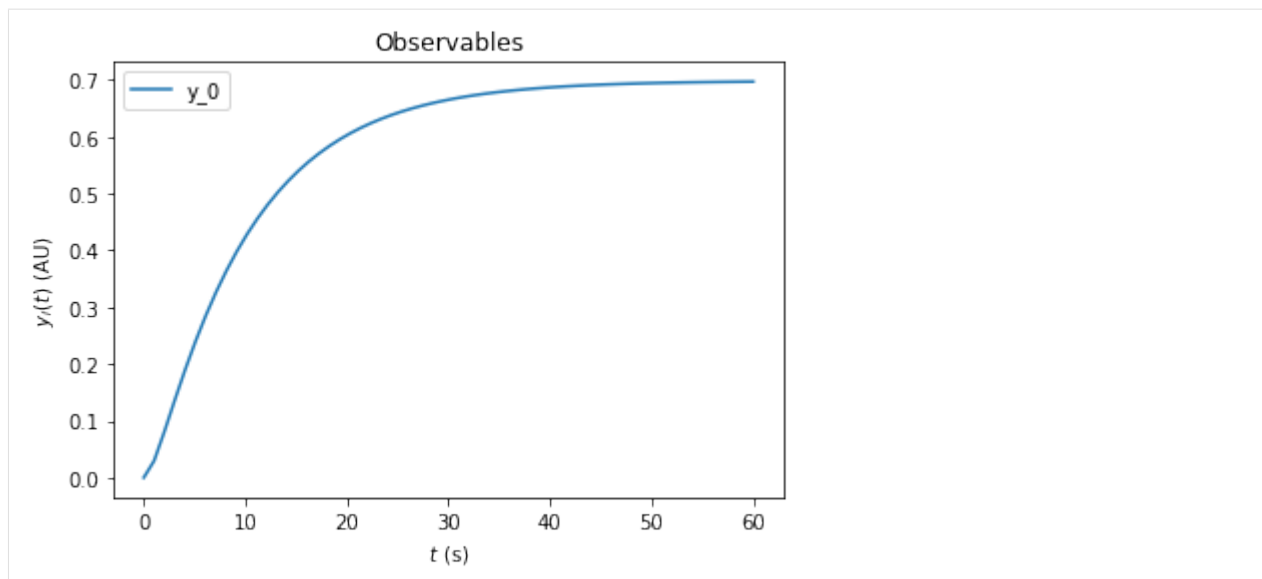
To simulate the model we need to create an instance via the `getModel()` method in the generated model module.

```
[8]: # Create Model instance
model = model_module.getModel()

# Create solver instance
solver = model.getSolver()
```

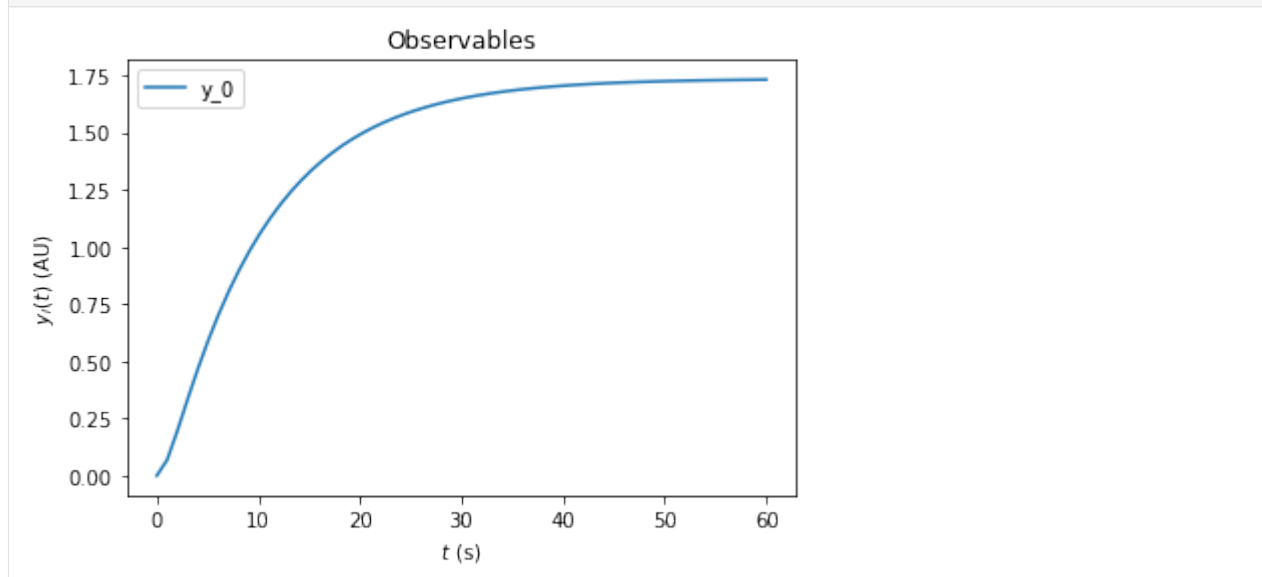
The only thing we need to simulate the model is a timepoint vector, which can be specified using the `setTimepoints` method. If we do not specify any additional options, the default values for `fixedParameters` and `parameters` that were specified in the SBML file will be used.

```
[9]: # Run simulation using default model parameters and solver options
model.setTimepoints(np.linspace(0, 60, 60))
rdata = amici.runAmiciSimulation(model, solver)
amici.plotting.plotObservableTrajectories(rdata)
```



Simulation options can be specified either in the `Model` or in an `ExpData` instance. The `ExpData` instance can also carry experimental data. To initialize an `ExpData` instance from simulation results, amici offers some [convenient constructors](#). In the following we will initialize an `ExpData` object from simulation results, but add noise with standard deviation 0.1 and specify the standard deviation accordingly. Moreover, we will specify custom values for `DRUG_0=0` and `KIN_0=2`. If `fixedParameter` is specified in an `ExpData` instance, `runAmiciSimulation` will use those parameters instead of the ones specified in the `Model` instance.

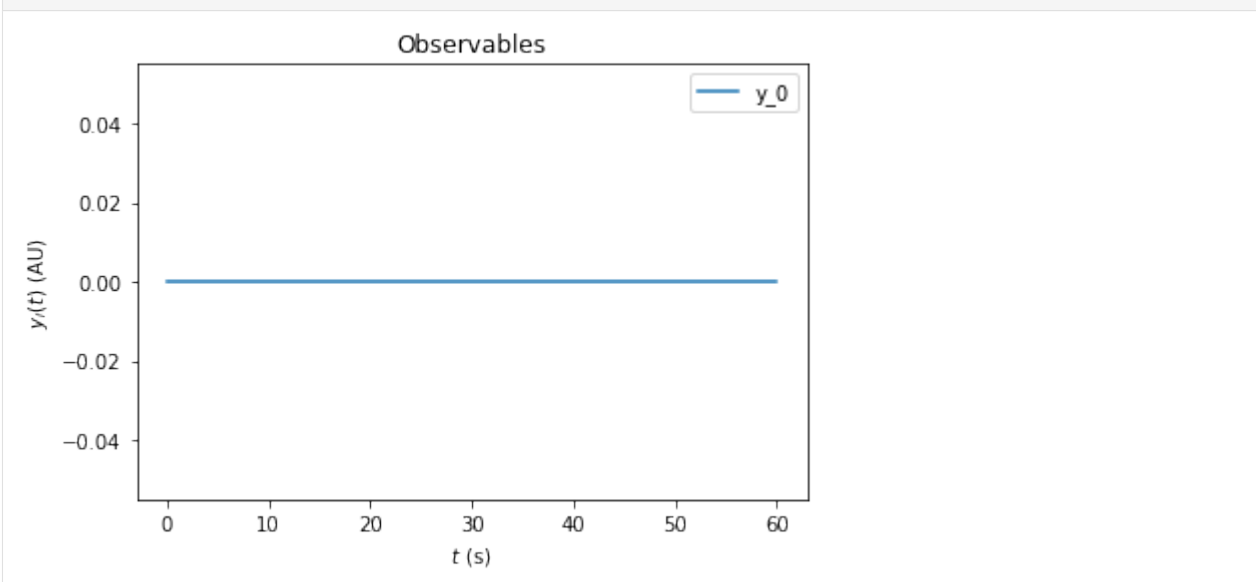
```
[10]: edata = amici.ExpData(rdata, 0.1, 0.0)
      edata.fixedParameters = [0, 2]
      rdata = amici.runAmiciSimulation(model, solver, edata)
      amici.plotting.plotObservableTrajectories(rdata)
```



For many biological systems, it is reasonable to assume that they start in a steady state. In this example we want to specify an experiment where a pretreatment with a drug is performed *before* the kinase is added. We assume that the pretreatment is sufficiently long such that the system reaches steadystate before the kinase is added. To implement this in amici, we can specify `fixedParametersPreequilibration` in the `ExpData` object. This automatically adds a preequilibration phase where the model is run to steadystate, before regular simulation starts. Here we set `DRUG_0=3`

and `KIN_0=0` for the preequilibration. This means that there is no kinase available in the preequilibration phase.

```
[11]: edata.fixedParametersPreequilibration = [3, 0]
      rdata = amici.runAmiciSimulation(model, solver, edata)
      amici.plotting.plotObservableTrajectories(rdata)
```

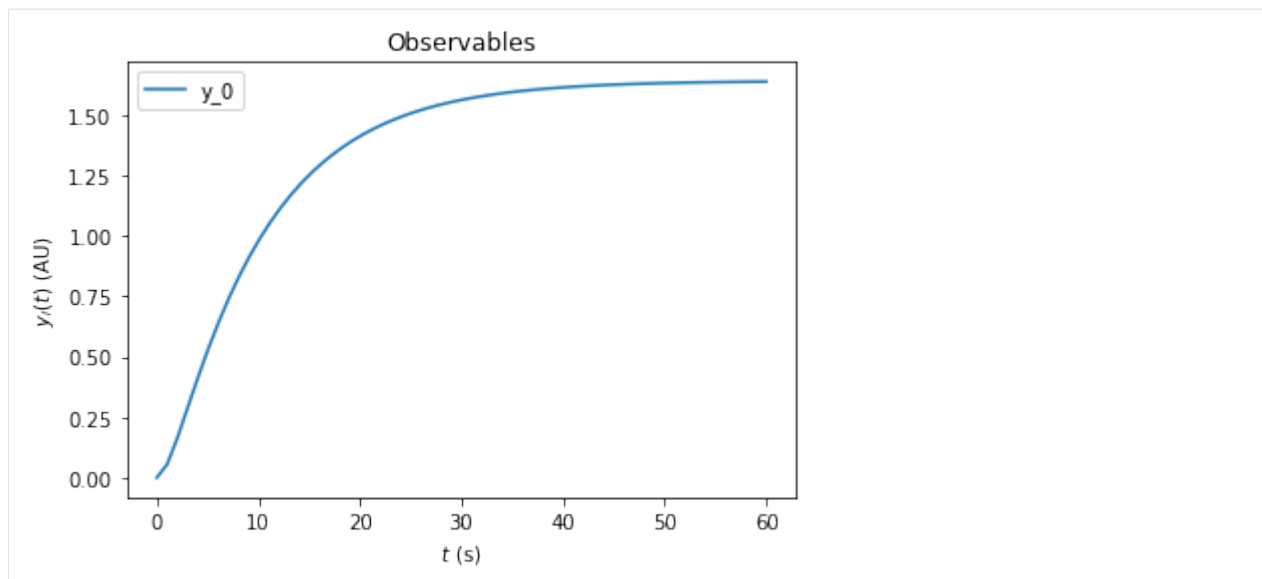


The resulting trajectory is definitely not what one may expect. The problem is that the `DRUG_0` and `KIN_0` set initial conditions for species in the model. By default these initial conditions are only applied at the very beginning of the simulation, i.e., before the preequilibration. Accordingly, the `fixedParameters` that we specified do not have any effect. To fix this, we need to set the `reinitializeFixedParameterInitialStates` attribute to `True`, to specify that AMICI reinitializes all states that have `fixedParameter`-dependent initial states.

```
[12]: edata.reinitializeFixedParameterInitialStates = True
```

With this option activated, the kinase concentration will be reinitialized after the preequilibration and we will see the expected change in fractional phosphorylation:

```
[13]: rdata = amici.runAmiciSimulation(model, solver, edata)
      amici.plotting.plotObservableTrajectories(rdata)
```

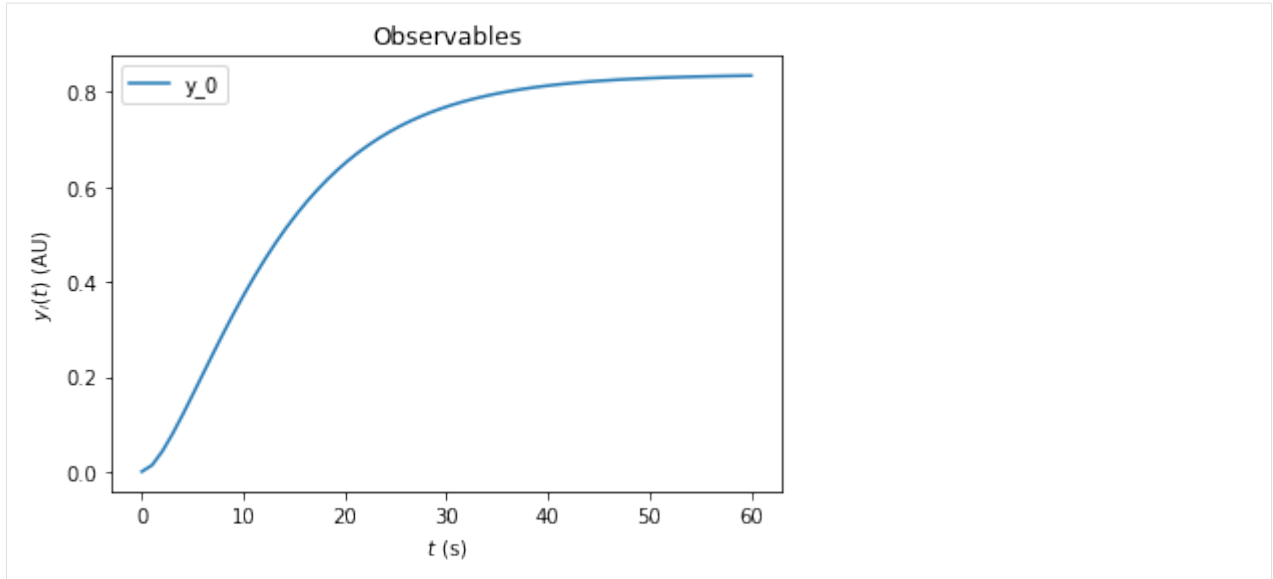


On top of preequilibration, we can also specify presimulation. This option can be used to specify pretreatments where the system is not assumed to reach steadystate. Presimulation can be activated by specifying `t_presim` and `edata.fixedParametersPresimulation`. If both `fixedParametersPresimulation` and `fixedParametersPreequilibration` are specified, preequilibration will be performed first, followed by presimulation, followed by regular simulation. For this example we specify `DRUG_0=10` and `KIN_0=0` for the presimulation and `DRUG_0=10` and `KIN_0=2` for the regular simulation. We do not overwrite the `DRUG_0=3` and `KIN_0=0` that was previously specified for preequilibration.

```
[14]: edata.t_presim = 10
      edata.fixedParametersPresimulation = [10.0, 0.0]
      edata.fixedParameters = [10.0, 2.0]
      print(edata.fixedParametersPreequilibration)
      print(edata.fixedParametersPresimulation)
      print(edata.fixedParameters)
      rdata = amici.runAmiciSimulation(model, solver, edata)
      amici.plotting.plotObservableTrajectories(rdata)

(3.0, 0.0)
(10.0, 0.0)
(10.0, 2.0)
```





### 10.2.5 AMICI documentation example of the steady state solver logic

This is an example to document the internal logic of the steady state solver, which is used in preequilibration and postequilibration.

#### Steady states of dynamical system

Not every dynamical system needs to run into a steady state. Instead, it may exhibit

- continuous growth, e.g.,

$$\dot{x} = x, \quad x_0 = 1$$

- a finite-time blow up, e.g.,

$$\dot{x} = x^2, \quad x_0 = 1$$

- oscillations, e.g.,

$$\ddot{x} = -x, \quad x_0 = 1$$

- chaotic behaviour, e.g., the Lorentz attractor

If the considered dynamical system has a steady state for positive times, then integrating the ODE long enough will equilibrate the system to this steady state. However, this may be computationally more demanding than other approaches and may fail, if the maximum number of integration steps is exceeded before reaching the steady state.

In general, Newton's method will find the steady state faster than forward simulation. However, it only converges if started close enough to the steady state. Moreover, it will not work, if the dynamical system has conserved quantities which were not removed prior to steady state computation: Conserved quantities will cause singularities in the Jacobian of the right hand side of the system, such that the linear problem within each step of Newton's method can not be solved.

### Logic of the steady state solver

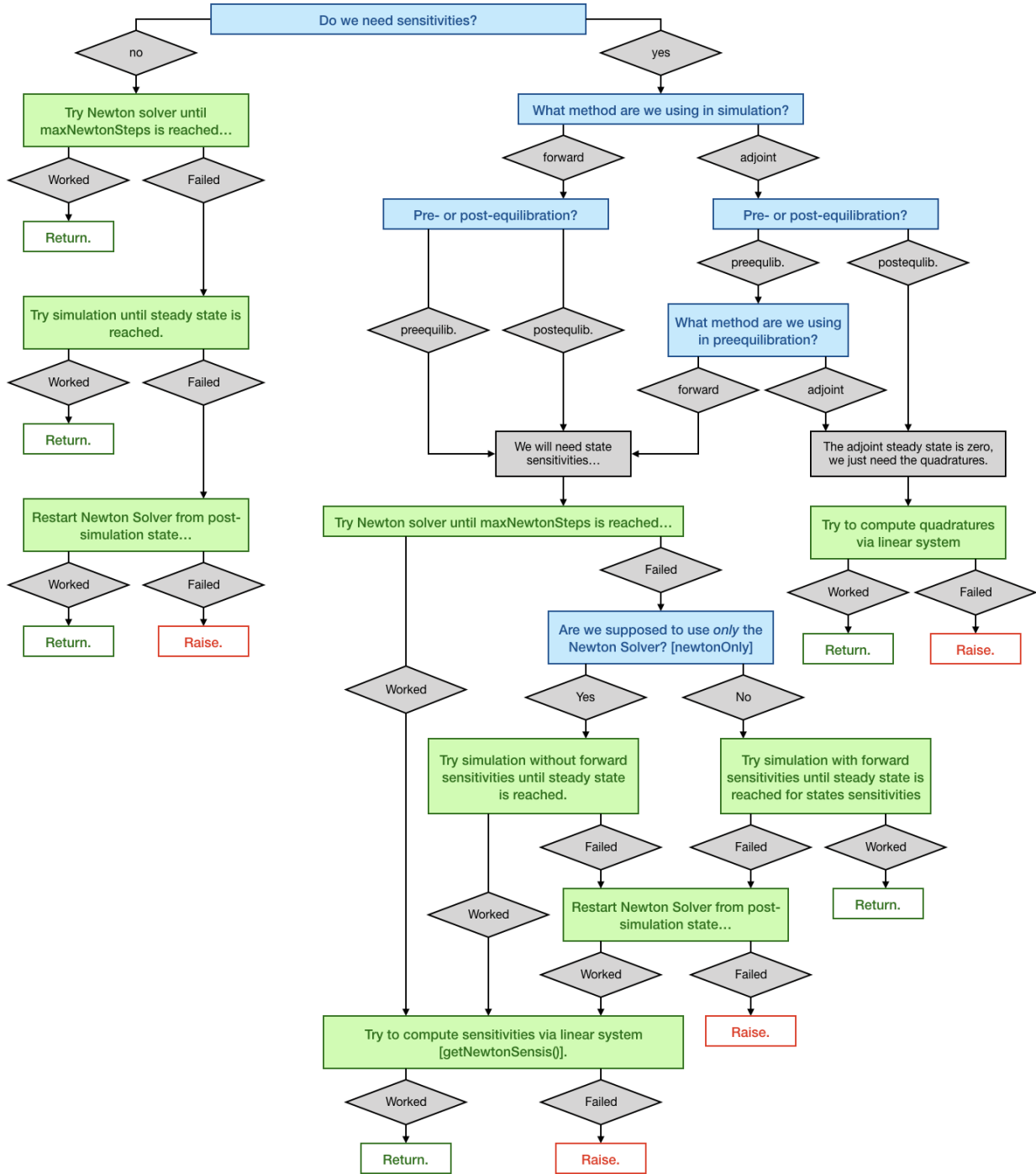
If AMICI has to equilibrate a dynamical system, it can do this either via simulating until the right hand side of the system becomes small, or it can try to find the steady state directly by Newton's method. Amici decides automatically which approach is chosen and how forward or adjoint sensitivities are computed, if requested. However, the user can influence this behavior, if prior knowledge about the dynamical is available.

The logic which AMICI will follow to equilibrate the system works as follows:

```
[1]: from IPython.display import Image

fig = Image(
    filename="../../../documentation/gfx/steadystate_solver_workflow.png")
fig
```

[1]:



## The example model

We will use the example model `model_constant_species.xml`, which has conserved species. Those are automatically removed in the SBML import of AMICI, but they can also be kept in the model to demonstrate the failure of Newton's method due to a singular right hand side Jacobian.

```
[2]: import libsbml
import importlib
import amici
import os
import sys
import numpy as np
import matplotlib.pyplot as plt

# SBML model we want to import
sbml_file = "model_constant_species.xml"

# Name of the models that will also be the name of the python module
model_name = "model_constant_species"
model_reduced_name = model_name + "_reduced"

# Directories to which the generated model code is written
model_output_dir = model_name
model_reduced_output_dir = model_reduced_name

# Read the model and give some output
sbml_reader = libsbml.SBMLReader()
sbml_doc = sbml_reader.readSBML(sbml_file)
sbml_model = sbml_doc.getModel()
dir(sbml_doc)

print("Species: ", [s.getId() for s in sbml_model.getListOfSpecies()])

print("\nReactions:")
for reaction in sbml_model.getListOfReactions():
    reactants = " + ".join(
        [
            "%s %s"
            % (
                int(r.getStoichiometry()) if r.getStoichiometry() > 1 else "",
                r.getSpecies(),
            )
            for r in reaction.getListOfReactants()
        ]
    )
    products = " + ".join(
        [
            "%s %s"
            % (
                int(r.getStoichiometry()) if r.getStoichiometry() > 1 else "",
                r.getSpecies(),
            )
            for r in reaction.getListOfProducts()
        ]
    )
```

(continues on next page)

(continued from previous page)

```

    ]
)
reversible = "<" if reaction.getReversible() else ""
print(
    "%3s: %10s %1s->%10s\t\t[%s]"
    % (
        reaction.getId(),
        reactants,
        reversible,
        products,
        libsbml.formulaToL3String(reaction.getKineticLaw().getMath()),
    )
)

```

Species: ['substrate', 'enzyme', 'complex', 'product']

Reactions:

```

creation:          -> substrate          [compartment * (synthesis_substrate + k_
↳create)]
binding: substrate + enzyme <-> complex      [compartment * (k_bind *
↳substrate * enzyme - k_unbind * complex)]
conversion: complex -> enzyme + product      [compartment * k_convert *
↳complex]
decay: product ->                          [compartment * k_decay * product]

```

[3]: *# Create an SbmlImporter instance for our SBML model*

```

sbml_importer = amici.SbmlImporter(sbml_file)

# specify observables and constant parameters
constantParameters = ["synthesis_substrate", "init_enzyme"]
observables = {
    "observable_product": {"name": "", "formula": "product"},
    "observable_substrate": {"name": "", "formula": "substrate"},
}
sigmas = {"observable_product": 1.0, "observable_substrate": 1.0}

# import the model
sbml_importer.sbml2amici(
    model_reduced_name,
    model_reduced_output_dir,
    observables=observables,
    constant_parameters=constantParameters,
    sigmas=sigmas,
)
sbml_importer.sbml2amici(
    model_name,
    model_output_dir,
    observables=observables,
    constant_parameters=constantParameters,
    sigmas=sigmas,
    compute_conservation_laws=False,
)

```

```
[4]: # import the models and run some test simulations
model_reduced_module = amici.import_model_module(
    model_reduced_name, os.path.abspath(model_reduced_output_dir)
)
model_reduced = model_reduced_module.getModel()

model_module = amici.import_model_module(
    model_name, os.path.abspath(model_output_dir)
)
model = model_module.getModel()

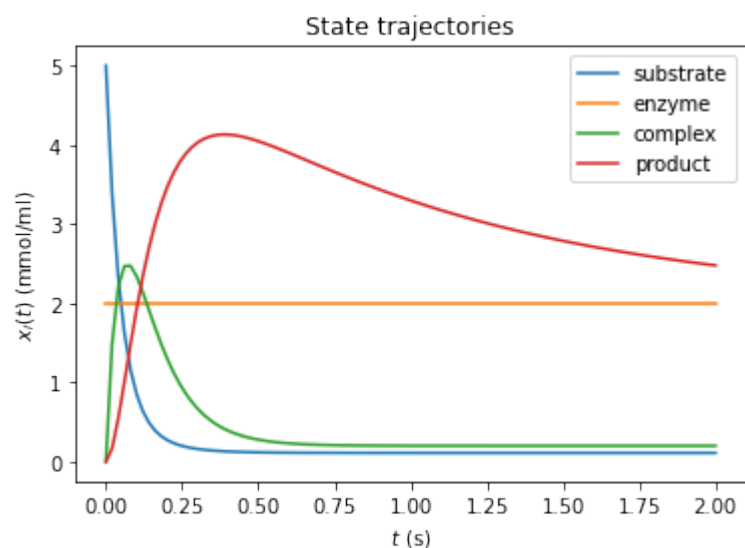
# simulate model with conservation laws
model_reduced.setTimepoints(np.linspace(0, 2, 100))
solver_reduced = model_reduced.getSolver()
rdata_reduced = amici.runAmiciSimulation(model_reduced, solver_reduced)

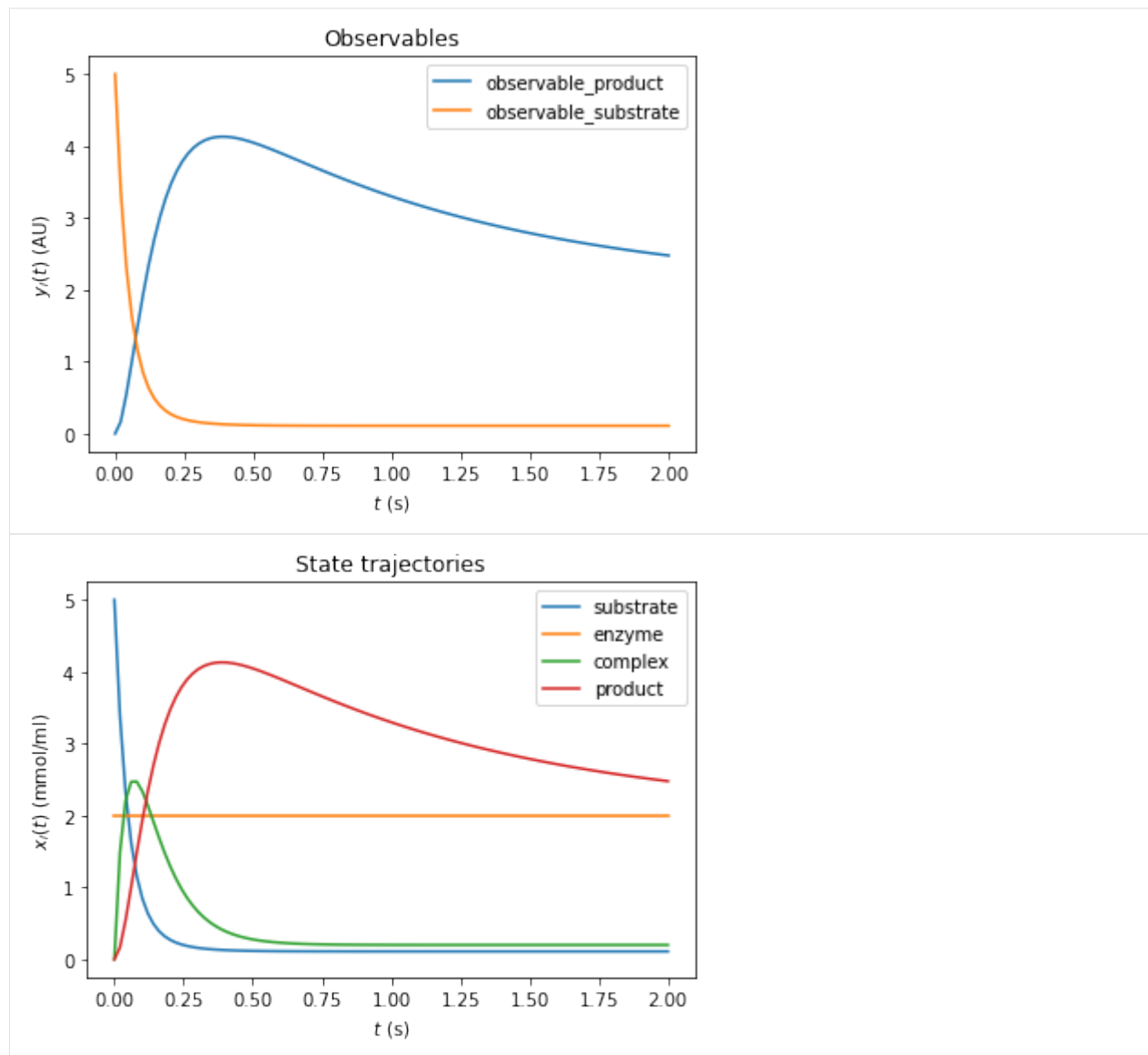
# simulate model without conservation laws
model.setTimepoints(np.linspace(0, 2, 100))
solver = model.getSolver()
rdata = amici.runAmiciSimulation(model, solver)

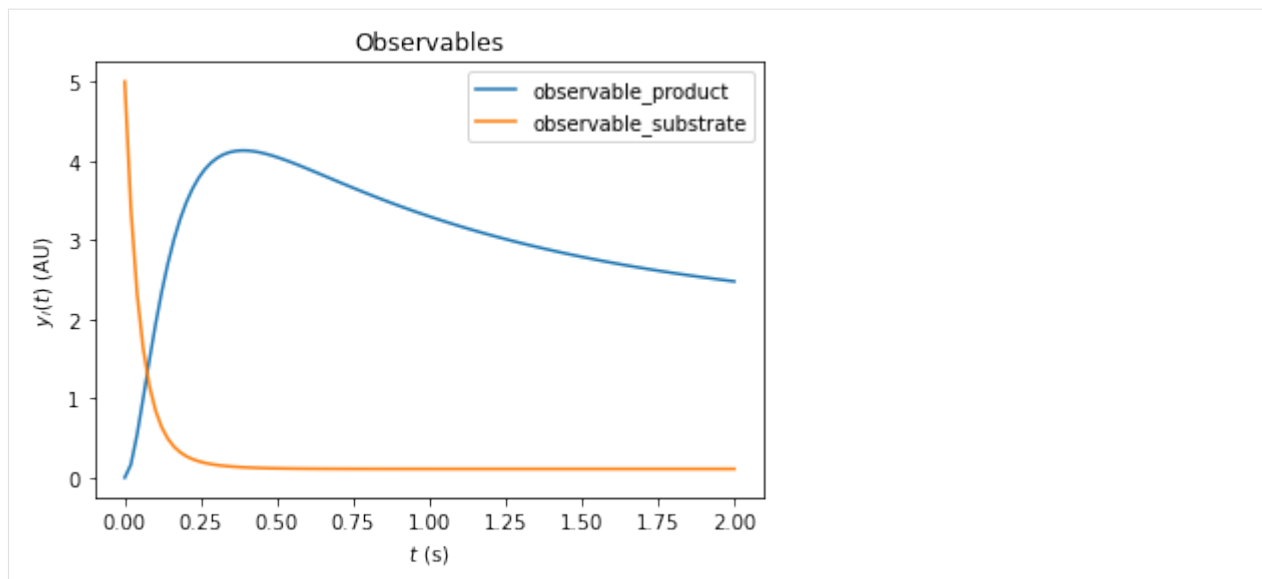
# plot trajectories
import amici.plotting

amici.plotting.plotStateTrajectories(rdata_reduced, model=model_reduced)
amici.plotting.plotObservableTrajectories(rdata_reduced, model=model_reduced)

amici.plotting.plotStateTrajectories(rdata, model=model)
amici.plotting.plotObservableTrajectories(rdata, model=model)
```







### Inferring the steady state of the system (postequilibration)

First, we want to demonstrate that Newton's method will fail with the unreduced model due to a singular right hand side Jacobian.

```
[5]: # Call postequilibration by setting an infinity timepoint
model.setTimepoints(np.full(1, np.inf))

# set the solver
solver = model.getSolver()
solver.setNewtonMaxSteps(10)
solver.setMaxSteps(1000)
rdata = amici.runAmiciSimulation(model, solver)

# np.set_printoptions(threshold=8, edgeitems=2)
for key, value in rdata.items():
    print("%12s: " % key, value)

    ts: [inf]
    x: [[0.11      2.          0.2          2.000000002]]
    x0: [5.  2.  0.  0.]
    x_ss: [nan nan nan nan]
    sx: None
    sx0: None
    sx_ss: None
    y: [[2.000000002 0.11      ]]
    sigmay: [[1.  1.]]
    sy: None
    ssigmay: None
    z: None
    rz: None
    sigmaz: None
    sz: None
```

(continues on next page)



(continued from previous page)

```

    srz: None
    ssigmaz: None
    slh: None
    s2llh: None
    J: [[-20.    0.   20.    0. ]
 [-1.1   0.    1.1   0. ]
 [ 1.    0.  -11.   10. ]
 [ 0.    0.    0.   -1. ]]
    xdot: [ 0.00000000e+00  0.00000000e+00  2.22044605e-16 -2.24170307e-08]
    status: 0.0
    llh: nan
    chi2: nan
    res: [0. 0.]
    sres: None
    FIM: None
    w: [[2.          2.          2.          2.00000002]]
    preeq_wrms: nan
    preeq_t: nan
    preeq_numlinsteps: None
    preeq_numsteps: [[0 0 0]]
    preeq_numstepsB: 12.0
    preeq_status: [[0 0 0]]
    preeq_cpu_time: 0.0
    preeq_cpu_timeB: 0.0
    posteq_wrms: 0.5604257578208488
    posteq_t: 19.2252094591474
    posteq_numlinsteps: None
    posteq_numsteps: [[ 0 417  0]]
    posteq_numstepsB: 0.0
    posteq_status: [[-3 1 0]]
    posteq_cpu_time: 2.315
    posteq_cpu_timeB: 0.0
    numsteps: None
    numrhsevals: None
    numerrtestfails: None
    numnonlinsolvconvfails: None
    order: None
    cpu_time: 0.0
    numstepsB: None
    numrhsevalsB: None
    numerrtestfailsB: None
    numnonlinsolvconvfailsB: None
    cpu_timeB: 0.0

```

The fields `posteq_status` and `posteq_numsteps` in `rdata` tells us how postequilibration worked:

- the first entry informs us about the status/number of steps in Newton's method (here 0, as Newton's method did not work)
- the second entry tells us, the status/how many integration steps were taken until steady state was reached
- the third entry informs us about the status/number of Newton steps in the second launch, after simulation

The status is encoded as an Integer flag with the following meanings:

- 1: Successful run
- 0: Did not run
- -1: Error: No further specification is given, the error message should give more information.
- -2: Error: The method did not converge to a steady state within the maximum number of steps (Newton's method or simulation).
- -3: Error: The Jacobian of the right hand side is singular (only Newton's method)
- -4: Error: The damping factor in Newton's method was reduced until it met the lower bound without success (Newton's method only)
- -5: Error: The model was simulated past the timepoint  $t=1e100$  without finding a steady state. Therefore, it is likely that the model has not steady state for the given parameter vector.

Here, only the second entry of `posteq_status` contains a positive integer: The first run of Newton's method failed due to a Jacobian, which could not be factorized, but the second run (simulation) contains the entry 1 (success). The third entry is 0, thus Newton's method was not launched for a second time. More information can be found in `posteq_numsteps`: Also here, only the second entry contains a positive integer, which is smaller than the maximum number of steps taken ( $<1000$ ). Hence steady state was reached via simulation, which corresponds to the simulated time written to `posteq_time`.

We want to demonstrate a complete failure if inferring the steady state by reducing the number of integration steps to a lower value:

```
[6]: # reduce maxsteps for integration
solver.setMaxSteps(100)
rdata = amici.runAmiciSimulation(model, solver)
print("Status of postequilibration:", rdata["posteq_status"])
print(
    "Number of steps employed in postequilibration:", rdata["posteq_numsteps"]
)

Status of postequilibration: [[-3 -2 -3]]
Number of steps employed in postequilibration: [[ 0 100  0]]

[Warning] AMICI:simulation: AMICI simulation failed:
Steady state computation failed. First run of Newton solver failed: RHS could not be
↳ factorized. Simulation to steady state failed: No convergence was achieved. Second run
↳ of Newton solver failed: RHS could not be factorized.
Error occurred in:
0      0x1060f7913 amici::SteadystateProblem::handleSteadyStateFailure(amici::Solver
↳ const*, amici::Model*) + 531
1      0x1060f6b3c amici::SteadystateProblem::findSteadyState(amici::Solver*, amici::
↳ NewtonSolver*, amici::Model*, int) + 332
2      0x1060f6882 amici::SteadystateProblem::workSteadyStateProblem(amici::Solver*,
↳ amici::Model*, int) + 322
3      0x1060a4615 amici::AmiciApplication::runAmiciSimulation(amici::Solver&, amici:
↳ :ExpData const*, amici::Model&, bool) + 405
4
```

However, the same logic works, if we use the reduced model. For sufficiently many Newton steps, postequilibration is achieved by Newton's method in the first run. In this specific example, the steady state is found within one step.

```
[7]: # Call postequilibration by setting an infinity timepoint
model_reduced.setTimepoints(np.full(1, np.inf))
```

(continues on next page)

(continued from previous page)

```
# set the solver
solver_reduced = model_reduced.getSolver()
solver_reduced.setNewtonMaxSteps(10)
solver_reduced.setMaxSteps(100)
rdata_reduced = amici.runAmiciSimulation(model_reduced, solver_reduced)

print("Status of postequilibration:", rdata_reduced["posteq_status"])
print(
    "Number of steps employed in postequilibration:",
    rdata_reduced["posteq_numsteps"],
)
```

```
Status of postequilibration: [[1 0 0]]
Number of steps employed in postequilibration: [[2 0 0]]
```

## Postequilibration with sensitivities

Equilibration is possible with forward and adjoint sensitivity analysis. As for the main simulation part, adjoint sensitivity analysis yields less information than forward sensitivity analysis, since no state sensitivities are computed. However, it has a better scaling behavior towards large model sizes.

## Postequilibration with forward sensitivities

If forward sensitivity analysis is used, then state sensitivities at the timepoint `np.inf` will be computed. This can be done in (currently) two different ways:

1. If the Jacobian  $\nabla_x f$  of the right hand side  $f$  is not (close to) singular, the most efficient approach will be solving the linear system of equations, which defines the steady state sensitivities:

$$0 = \dot{s}^x = (\nabla_x f) s^x + \frac{\partial f}{\partial \theta} \quad \Rightarrow \quad (\nabla_x f) s^x = -\frac{\partial f}{\partial \theta}$$

This approach will always be chosen by AMICI, if the option `model.SteadyStateSensitivityMode` is set to `SteadyStateSensitivityMode.newtonOnly`. Furthermore, it will also be chosen, if the steady state was found by Newton's method, as in this case, the Jacobian is at least not singular (but may still be poorly conditioned). A check for the condition number of the Jacobian is currently missing, but will soon be implemented.

2. If the Jacobian is poorly conditioned or singular, then the only way to obtain a reliable result will be integrating the state variables with state sensitivities until the norm of the right hand side becomes small. This approach will be chosen by AMICI, if the steady state was found by simulation and the option `model.SteadyStateSensitivityMode` is set to `SteadyStateSensitivityMode.simulationFSA`. This approach is numerically more stable, but the computation time for large models may be substantial.

Side remark:

A possible third way may consist in a (relaxed) Richardson iteration type approach, which interprets the entries of the right hand side  $f$  as residuals and minimizes the squared residuals  $\|f\|^2$  by a Levenberg-Marquart-type algorithm. This approach would also work for poorly conditioned (and even for singular Jacobians if additional constraints are implemented as Lagrange multipliers) while being faster than a long forward simulation.

We want to demonstrate both possibilities to find the steady state sensitivities, as well as the failure of their computation if the Jacobian is singular and the `newtonOnly` setting was used.

```
[8]: # Call simulation with singular Jacobian and integrateIfNewtonFails mode
model.setTimepoints(np.full(1, np.inf))
model.setSteadyStateSensitivityMode(
    amici.SteadyStateSensitivityMode.integrateIfNewtonFails
)
solver = model.getSolver()
solver.setNewtonMaxSteps(10)
solver.setSensitivityMethod(amici.SensitivityMethod.forward)
solver.setSensitivityOrder(amici.SensitivityOrder.first)
solver.setMaxSteps(10000)
rdata = amici.runAmiciSimulation(model, solver)

print("Status of postequilibration:", rdata["posteq_status"])
print(
    "Number of steps employed in postequilibration:", rdata["posteq_numsteps"]
)
print("Computed state sensitivities:")
print(rdata["sx"][0, :, :])
```

```
Status of postequilibration: [[-3  1  0]]
Number of steps employed in postequilibration: [[ 0 1026  0]]
Computed state sensitivities:
[[-1.10000000e-02  0.00000000e+00 -6.70507402e-18 -1.20114408e-11]
 [ 1.00000000e-02  0.00000000e+00 -8.22965063e-19  1.20114329e-11]
 [-1.00000000e-03  0.00000000e+00 -2.00000000e-02 -2.40228711e-11]
 [ 5.50000000e-02  0.00000000e+00  1.00000000e-01  9.99999999e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00 -2.00000000e+00]]
```

```
[9]: # Call simulation with singular Jacobian and newtonOnly mode (will fail)
model.setTimepoints(np.full(1, np.inf))
model.setSteadyStateSensitivityMode(
    amici.SteadyStateSensitivityMode.newtonOnly
)
solver = model.getSolver()
solver.setSensitivityMethod(amici.SensitivityMethod.forward)
solver.setSensitivityOrder(amici.SensitivityOrder.first)
solver.setMaxSteps(10000)
rdata = amici.runAmiciSimulation(model, solver)

print("Status of postequilibration:", rdata["posteq_status"])
print(
    "Number of steps employed in postequilibration:", rdata["posteq_numsteps"]
)
print("Computed state sensitivities:")
print(rdata["sx"][0, :, :])
```

```
Status of postequilibration: [[-2 -1  1]]
Number of steps employed in postequilibration: [[ 0 543  0]]
Computed state sensitivities:
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

```
[Warning] AMICI:simulation: AMICI simulation failed:
Steady state sensitivitiy computation failed due to unsuccessful factorization of RHS
↳ Jacobian
Error occurred in:
0      0x1060f698b amici::SteadystateProblem::workSteadyStateProblem(amici::Solver*,
↳ amici::Model*, int) + 587
1      0x1060a4615 amici::AmiciApplication::runAmiciSimulation(amici::Solver&, amici:
↳ :ExpData const*, amici::Model&, bool) + 405
2      0x1060a4474 amici::runAmiciSimulation(amici::Solver&, amici::ExpData const*,
↳ amici::Model&, bool) + 36
3      0x106061005 _wrap_runAmiciSimulation(_object*, _object*) + 549
4      0x1021b2309 cfunction_call_varargs + 320
5
```

```
[10]: # Call postequilibration by setting an infinity timepoint
model_reduced.setTimepoints(np.full(1, np.inf))
model.setSteadyStateSensitivityMode(
    amici.SteadyStateSensitivityMode.newtonOnly
)
solver_reduced = model_reduced.getSolver()
solver_reduced.setNewtonMaxSteps(10)
solver_reduced.setSensitivityMethod(amici.SensitivityMethod.forward)
solver_reduced.setSensitivityOrder(amici.SensitivityOrder.first)
solver_reduced.setMaxSteps(1000)
rdata_reduced = amici.runAmiciSimulation(model_reduced, solver_reduced)

print("Status of postequilibration:", rdata_reduced["posteq_status"])
print(
    "Number of steps employed in postequilibration:",
    rdata_reduced["posteq_numsteps"],
)
print("Computed state sensitivities:")
print(rdata_reduced["sx"][0, :, :])
```

```
Status of postequilibration: [[1 0 0]]
Number of steps employed in postequilibration: [[2 0 0]]
Computed state sensitivities:
[[-1.1e-02  0.0e+00 -0.0e+00 -0.0e+00]
 [ 1.0e-02  0.0e+00 -0.0e+00 -0.0e+00]
 [-1.0e-03  0.0e+00 -2.0e-02 -0.0e+00]
 [ 5.5e-02  0.0e+00  1.0e-01  1.0e+00]
 [-0.0e+00  0.0e+00 -0.0e+00 -2.0e+00]]
```

## Postequilibration with adjoint sensitivities

Postequilibration also works with adjoint sensitivities. In this case, it is exploited that the ODE of the adjoint state  $p$  will always have the steady state 0, since it's a linear ODE:

$$\frac{d}{dt}p(t) = J(x^*, \theta)^T p(t),$$

where  $x^*$  denotes the steady state of the system state. Since the Eigenvalues of the Jacobian are negative and since the Jacobian at steady state is a fixed matrix, this system has a simple algebraic solution:

$$p(t) = e^{tJ(x^*, \theta)^T} p_{\text{end}}.$$

As a consequence, the quadratures in adjoint computation also reduce to a matrix-vector product:

$$Q(x, \theta) = Q(x^*, \theta) = p_{\text{integral}} * \frac{\partial f}{\partial \theta}$$

with

$$p_{\text{integral}} = \int_0^\infty p(s) ds = (J(x^*, \theta)^T)^{-1} p_{\text{end}}.$$

However, this solution is given in terms of a linear system of equations defined by the transposed Jacobian of the right hand side. Hence, if the (transposed) Jacobian is singular, it is not applicable. In this case, standard integration must be carried out.

```
[11]: # Call adjoint postequilibration by setting an infinity timepoint
# and create an edata object, which is needed for adjoint computation
edata = amici.ExpData(2, 0, 0, np.array([float("inf")]))
edata.setObservedData([1.8] * 2)
edata.fixedParameters = np.array([3.0, 5.0])

model_reduced.setSteadyStateSensitivityMode(
    amici.SteadyStateSensitivityMode.newtonOnly
)
solver_reduced = model_reduced.getSolver()
solver_reduced.setNewtonMaxSteps(10)
solver_reduced.setSensitivityMethod(amici.SensitivityMethod.adjoint)
solver_reduced.setSensitivityOrder(amici.SensitivityOrder.first)
solver_reduced.setMaxSteps(1000)
rdata_reduced = amici.runAmiciSimulation(model_reduced, solver_reduced, edata)

print("Status of postequilibration:", rdata_reduced["posteq_status"])
print(
    "Number of steps employed in postequilibration:",
    rdata_reduced["posteq_numsteps"],
)
print(
    "Number of backward steps employed in postequilibration:",
    rdata_reduced["posteq_numstepsB"],
)
print("Computed gradient:", rdata_reduced["sllh"])

Status of postequilibration: [[1 0 0]]
Number of steps employed in postequilibration: [[2 0 0]]
Number of backward steps employed in postequilibration: 0.0
Computed gradient: [-1.859000e-02  1.690000e-02 -1.690000e-03 -3.16282e+00  1.600000e+01]
```

If we carry out the same computation with a system that has a singular Jacobian, then `posteq_numstepsB` will not be 0 any more (which indicates that the linear system solve was used to compute backward postequilibration). Now, integration is carried out and hence `posteq_numstepsB > 0`

```
[12]: # Call adjoint postequilibration with model with singular Jacobian
model.setSteadyStateSensitivityMode(
    amici.SteadyStateSensitivityMode.newtonOnly
)
solver = model.getSolver()
solver.setNewtonMaxSteps(10)
solver.setSensitivityMethod(amici.SensitivityMethod.adjoint)
solver.setSensitivityOrder(amici.SensitivityOrder.first)
rdata = amici.runAmiciSimulation(model, solver, edata)

print("Status of postequilibration:", rdata["posteq_status"])
print(
    "Number of steps employed in postequilibration:", rdata["posteq_numsteps"]
)
print(
    "Number of backward steps employed in postequilibration:",
    rdata["posteq_numstepsB"],
)
print("Computed gradient:", rdata["sllh"])

Status of postequilibration: [[-3 -1  1]]
Number of steps employed in postequilibration: [[ 0 479  0]]
Number of backward steps employed in postequilibration: 3076.0
Computed gradient: [-1.85899987e-02  1.68999988e-02 -1.690000055e-03 -3.16282001e+00
 1.600000000e+01]
```

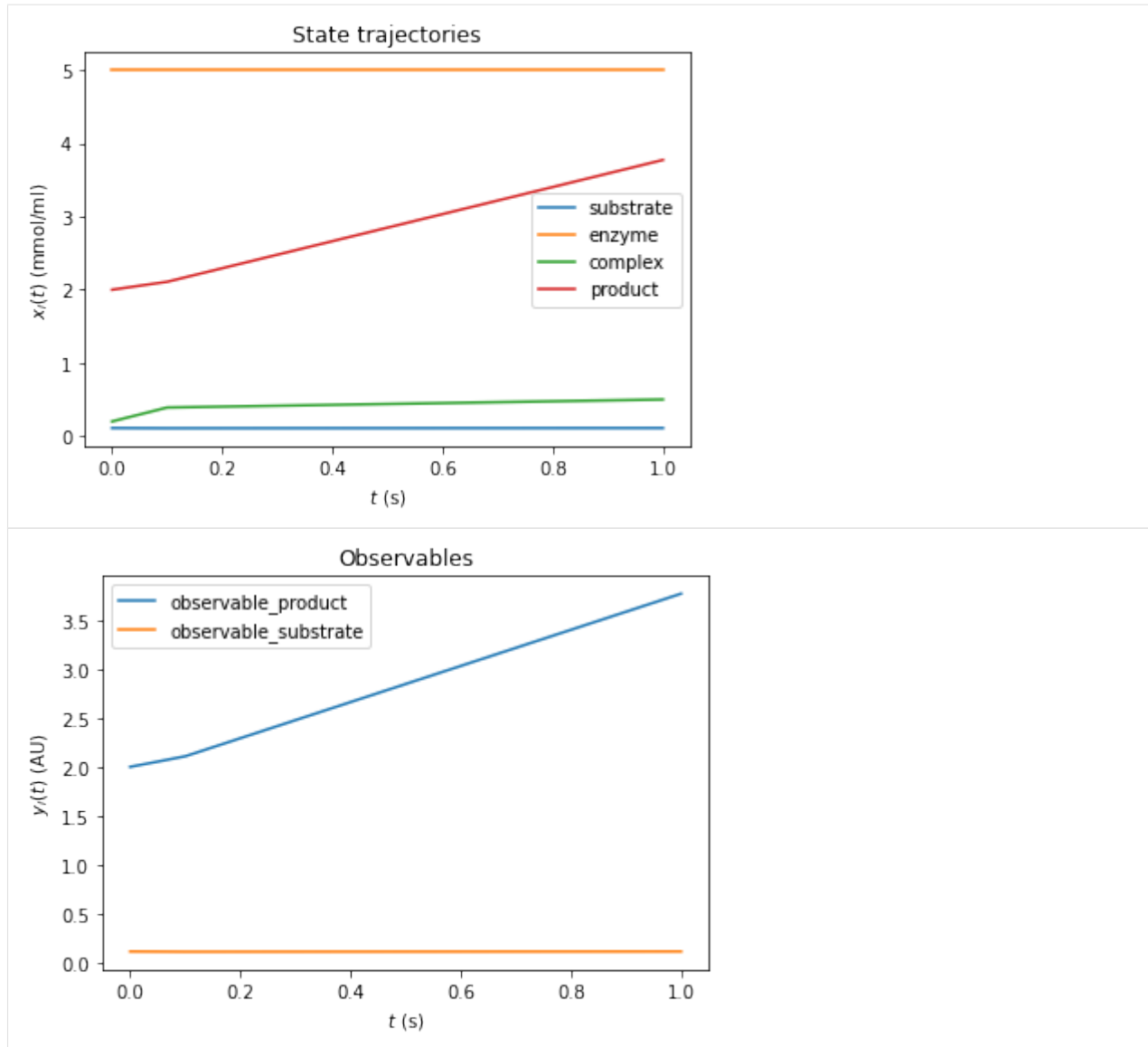
## Preequilibrating the model

Sometimes, we want to launch a solver run from a steady state which was inferred numerically, i.e., the system was preequilibrated. In order to do this with AMICI, we need to pass an `ExpData` object, which contains fixed parameter for the actual simulation and for preequilibration of the model.

```
[13]: # create edata, with 3 timepoints and 2 observables:
edata = amici.ExpData(2, 0, 0, np.array([0.0, 0.1, 1.0]))
edata.setObservedData([1.8] * 6)
edata.fixedParameters = np.array([3.0, 5.0])
edata.fixedParametersPreequilibration = np.array([0.0, 2.0])
edata.reinitializeFixedParameterInitialStates = True

[14]: # create the solver object and run the simulation
solver_reduced = model_reduced.getSolver()
solver_reduced.setNewtonMaxSteps(10)
rdata_reduced = amici.runAmiciSimulation(model_reduced, solver_reduced, edata)

amici.plotting.plotStateTrajectories(rdata_reduced, model=model_reduced)
amici.plotting.plotObservableTrajectories(rdata_reduced, model=model_reduced)
```



We can also combine pre- and postequilibration.

```
[15]: # Change the last timepoint to an infinity timepoint.
      edata.setTimepoints(np.array([0.0, 0.1, float("inf")]))

      # run the simulation
      rdata_reduced = amici.runAmiciSimulation(model_reduced, solver_reduced, edata)
```



## Preequilibration with sensitivities

Beyond the need for an ExpData object, the steady state solver logic in preequilibration is the same as in postequilibration, also if sensitivities are requested. The computation will fail for singular Jacobians, if SteadyStateSensitivityMode is set to newtonOnly, or if not enough steps can be taken. However, if forward simulation with steady state sensitivities is allowed, or if the Jacobian is not singular, it will work.

## Prequilibration with forward sensitivities

```
[16]: # No postequilibration this time.
edata.setTimepoints(np.array([0.0, 0.1, 1.0]))

# create the solver object and run the simulation, singular Jacobian, enforce Newton
↳ solver for sensitivities
model.setSteadyStateSensitivityMode(
    amici.SteadyStateSensitivityMode.newtonOnly
)
solver = model.getSolver()
solver.setNewtonMaxSteps(10)
solver.setSensitivityMethod(amici.SensitivityMethod.forward)
solver.setSensitivityOrder(amici.SensitivityOrder.first)
rdata = amici.runAmiciSimulation(model, solver, edata)

for key, value in rdata.items():
    if key[0:6] == "preeq_":
        print("%20s: " % key, value)

        preeq_wrms: 0.5604257578208488
        preeq_t: 19.2252094591474
        preeq_numlinsteps: None
        preeq_numsteps: [[ 0 417 0]]
        preeq_numstepsB: 0.0
        preeq_status: [[-3 1 0]]
        preeq_cpu_time: 1.723
        preeq_cpu_timeB: 0.0

[Warning] AMICI:simulation: AMICI simulation failed:
Steady state sensitivity computation failed due to unsuccessful factorization of RHS
↳ Jacobian
Error occurred in:
0      0x1060f698b amici::SteadystateProblem::workSteadyStateProblem(amici::Solver*,
↳ amici::Model*, int) + 587
1      0x1060a456f amici::AmiciApplication::runAmiciSimulation(amici::Solver&, amici:
↳ :ExpData const*, amici::Model&, bool) + 239
2      0x1060a4474 amici::runAmiciSimulation(amici::Solver&, amici::ExpData const*,
↳ amici::Model&, bool) + 36
3      0x106061005 _wrap_runAmiciSimulation(_object*, _object*) + 549
4      0x1021b2309 cfunction_call_varargs + 320
5
```

```
[17]: # Singular Jacobian, use simulation
model.setSteadyStateSensitivityMode(
    amici.SteadyStateSensitivityMode.integrateIfNewtonFails
```

(continues on next page)

(continued from previous page)

```
)
solver = model.getSolver()
solver.setNewtonMaxSteps(10)
solver.setSensitivityMethod(amici.SensitivityMethod.forward)
solver.setSensitivityOrder(amici.SensitivityOrder.first)
rdata = amici.runAmiciSimulation(model, solver, edata)
```

```
for key, value in rdata.items():
    if key[0:6] == "preeq_":
        print("%20s: " % key, value)
```

```

    preeq_wrms:  0.9920376238481097
    preeq_t:    21.270502326483026
preeq_numlinsteps: None
    preeq_numsteps:  [[  0 1026   0]]
    preeq_numstepsB:  0.0
    preeq_status:    [[-3  1  0]]
    preeq_cpu_time:  12.439
    preeq_cpu_timeB:  0.0
```

[18]: *# Non-singular Jacobian, use Newton solver*

```
solver_reduced = model_reduced.getSolver()
solver_reduced.setNewtonMaxSteps(10)
solver_reduced.setSensitivityMethod(amici.SensitivityMethod.forward)
solver_reduced.setSensitivityOrder(amici.SensitivityOrder.first)
rdata_reduced = amici.runAmiciSimulation(model_reduced, solver_reduced, edata)
```

```
for key, value in rdata_reduced.items():
    if key[0:6] == "preeq_":
        print("%20s: " % key, value)
```

```

    preeq_wrms:  0.0
    preeq_t:    nan
preeq_numlinsteps: None
    preeq_numsteps:  [[2 0 0]]
    preeq_numstepsB:  0.0
    preeq_status:    [[1 0 0]]
    preeq_cpu_time:  0.036
    preeq_cpu_timeB:  0.0
```

## Preequilibration with adjoint sensitivities

When using pre-equilibration, adjoint sensitivity analysis can be used for simulation. This is a particularly interesting case: Standard adjoint sensitivity analysis requires the initial state sensitivities  $\mathbf{sx}_0$  to work, at least if data is given for finite (i.e., not exclusively post-equilibration) timepoints: For each parameter, a contribution to the gradient is given by the scalar product of the corresponding state sensitivity vector at timepoint  $t = 0$ , (column in  $\mathbf{sx}_0$ ), with the adjoint state ( $p(t = 0)$ ). Hence, the matrix  $\mathbf{sx}_0$  is needed. This scalar product “closes the loop” from forward to adjoint simulation.

By default, if adjoint sensitivity analysis is called with pre-equilibration, the initial state sensitivities are computed in just the same way as if this way done for forward sensitivity analysis. The only difference in the internal logic is that, if the steady state gets inferred via simulation, a separate solver object is used in order to ensure that the steady state simulation does not interfere with the snapshotting of the forward trajectory from the actual time course.

However, also an adjoint version of preequilibration is possible: In this case, the “loop” from forward to adjoint simulation needs no closure: The simulation time is extended by preequilibration: forward from  $t = -\infty$  to  $t = 0$ , and after adjoint simulation also backward from  $t = 0$  to  $t = -\infty$ . Similar to adjoint postequilibration, the steady state of the adjoint state (at  $t = -\infty$ ) is  $p = 0$ , hence the scalar product (at  $t = -\infty$ ) for the initial state sensitivities of preequilibration with the adjoint state vanishes. Instead, this gradient contribution is covered by additional quadratures  $\int_{-\infty}^0 p(s) ds \cdot \frac{\partial f}{\partial \theta}$ . In order to compute these quadratures correctly, the adjoint state from the main adjoint simulation must be passed on to the initial adjoint state of backward preequilibration.

However, as the adjoint state must be passed on from backward computation to preequilibration, it is currently not allowed to alter (reinitialize) states of the model at  $t = 0$ , unless these states are constant, as otherwise this alteration would lead to a discontinuity in the adjoints state as well and hence to an incorrect gradient.

```
[19]: # Non-singular Jacobian, use Newton solver and adjoints with initial state sensitivities
solver_reduced = model_reduced.getSolver()
solver_reduced.setNewtonMaxSteps(10)
solver_reduced.setSensitivityMethod(amici.SensitivityMethod.adjoint)
solver_reduced.setSensitivityOrder(amici.SensitivityOrder.first)
rdata_reduced = amici.runAmiciSimulation(model_reduced, solver_reduced, edata)

for key, value in rdata_reduced.items():
    if key[0:6] == "preeq_":
        print("%20s: " % key, value)
print("Gradient:", rdata_reduced["sllh"])

      preeq_wrms:  0.0
      preeq_t:    nan
preeq_numlinsteps: None
      preeq_numsteps:  [[2  0  0]]
      preeq_numstepsB:  0.0
      preeq_status:    [[1  0  0]]
      preeq_cpu_time:  0.039
      preeq_cpu_timeB:  0.0
Gradient: [-0.05528395  0.0461776 -0.03354519 -2.34602219  6.314481  ]
```

```
[20]: # Non-singular Jacobian, use simulation solver and adjoints with initial state_
      ↪sensitivities
solver_reduced = model_reduced.getSolver()
solver_reduced.setNewtonMaxSteps(0)
solver_reduced.setSensitivityMethod(amici.SensitivityMethod.adjoint)
solver_reduced.setSensitivityOrder(amici.SensitivityOrder.first)
rdata_reduced = amici.runAmiciSimulation(model_reduced, solver_reduced, edata)

for key, value in rdata_reduced.items():
    if key[0:6] == "preeq_":
        print("%20s: " % key, value)
print("Gradient:", rdata_reduced["sllh"])

      preeq_wrms:  0.8470065245264354
      preeq_t:    19.213162474372176
preeq_numlinsteps: None
      preeq_numsteps:  [[  0 426  0]]
      preeq_numstepsB:  0.0
      preeq_status:    [[-2  1  0]]
      preeq_cpu_time:  1.753
      preeq_cpu_timeB:  0.0
```

(continues on next page)

(continued from previous page)

```
Gradient: [-0.05528395  0.0461776 -0.03354519 -2.34602226  6.3144812 ]
```

```
[21]: # Non-singular Jacobian, use Newton solver and adjoints with fully adjoint_
↳preequilibration
solver_reduced = model_reduced.getSolver()
solver_reduced.setNewtonMaxSteps(10)
solver_reduced.setSensitivityMethod(amici.SensitivityMethod.adjoint)
solver_reduced.setSensitivityMethodPreequilibration(
    amici.SensitivityMethod.adjoint
)
solver_reduced.setSensitivityOrder(amici.SensitivityOrder.first)
rdata_reduced = amici.runAmiciSimulation(model_reduced, solver_reduced, edata)

for key, value in rdata_reduced.items():
    if key[0:6] == "preeq_":
        print("%20s: " % key, value)
print("Gradient:", rdata_reduced["slh"])

    preeq_wrms: 0.0
    preeq_t: nan
    preeq_numlinsteps: None
    preeq_numsteps: [[2 0 0]]
    preeq_numstepsB: 0.0
    preeq_status: [[1 0 0]]
    preeq_cpu_time: 0.042
    preeq_cpu_timeB: 0.009
Gradient: [-0.05528395  0.0461776 -0.03354519 -2.34602219  6.314481  ]
```

As for postquilibration, adjoint preequilibration has an analytic solution (via the linear system), which will be preferred. If used for models with singular Jacobian, numerical integration will be carried out, which is indicated by `preeq_numstepsB`.

```
[22]: # Non-singular Jacobian, use Newton solver and adjoints with fully adjoint_
↳preequilibration
solver = model.getSolver()
solver.setNewtonMaxSteps(10)
solver.setSensitivityMethod(amici.SensitivityMethod.adjoint)
solver.setSensitivityMethodPreequilibration(amici.SensitivityMethod.adjoint)
solver.setSensitivityOrder(amici.SensitivityOrder.first)
rdata = amici.runAmiciSimulation(model, solver, edata)

for key, value in rdata.items():
    if key[0:6] == "preeq_":
        print("%20s: " % key, value)
print("Gradient:", rdata["slh"])

    preeq_wrms: 0.9986067660342685
    preeq_t: 36.94272314329062
    preeq_numlinsteps: None
    preeq_numsteps: [[ 0 417 0]]
    preeq_numstepsB: 1371.0
    preeq_status: [[-3 1 0]]
    preeq_cpu_time: 2.488
```

(continues on next page)

(continued from previous page)

```
preeq_cpu_timeB: 5.016
Gradient: [-0.05528395  0.04617759 -0.03354518 -2.34602224  6.3144811 ]
```

## Controlling the error tolerances in pre- and postequilibration

When solving ODEs or DAEs, AMICI uses the default logic of CVODES and IDAS to control error tolerances. This means that error weights are computed based on the absolute error tolerances and the product of current state variables of the system and their respective relative error tolerances. If this error combination is then controlled.

The respective tolerances for equilibrating a system with AMICI can be controlled by the user via the getter/setter functions `[get|set][Absolute|Relative]ToleranceSteadyState[Sensi]`:

```
[23]: # Non-singular Jacobian, use simulaiton
model_reduced.setSteadyStateSensitivityMode(
    amici.SteadyStateSensitivityMode.integrateIfNewtonFails
)
solver_reduced = model_reduced.getSolver()
solver_reduced.setNewtonMaxSteps(0)
solver_reduced.setSensitivityMethod(amici.SensitivityMethod.forward)
solver_reduced.setSensitivityOrder(amici.SensitivityOrder.first)

# run with lax tolerances
solver_reduced.setRelativeToleranceSteadyState(1e-2)
solver_reduced.setAbsoluteToleranceSteadyState(1e-3)
solver_reduced.setRelativeToleranceSteadyStateSensi(1e-2)
solver_reduced.setAbsoluteToleranceSteadyStateSensi(1e-3)
rdata_reduced_lax = amici.runAmiciSimulation(
    model_reduced, solver_reduced, edata
)

# run with strict tolerances
solver_reduced.setRelativeToleranceSteadyState(1e-12)
solver_reduced.setAbsoluteToleranceSteadyState(1e-16)
solver_reduced.setRelativeToleranceSteadyStateSensi(1e-12)
solver_reduced.setAbsoluteToleranceSteadyStateSensi(1e-16)
rdata_reduced_strict = amici.runAmiciSimulation(
    model_reduced, solver_reduced, edata
)

# compare ODE outputs
print("\nODE solver steps, which were necessary to reach steady state:")
print("lax tolerances: ", rdata_reduced_lax["preeq_numsteps"])
print("strict tolerances: ", rdata_reduced_strict["preeq_numsteps"])

print("\nsimulation time corresponding to steady state:")
print(rdata_reduced_lax["preeq_t"])
print(rdata_reduced_strict["preeq_t"])

print("\ncomputation time to reach steady state:")
print(rdata_reduced_lax["preeq_cpu_time"])
print(rdata_reduced_strict["preeq_cpu_time"])
```

ODE solver steps, which were necessary to reach steady state:

lax tolerances:  $\begin{bmatrix} 0 & 733 & 0 \end{bmatrix}$

strict tolerances:  $\begin{bmatrix} 0 & 1031 & 0 \end{bmatrix}$

simulation time corresponding to steady state:

6.002011407974004

31.0689293433781

computation time to reach steady state:

7.646

7.837

## 10.2.6 Debugging simulation failures

**Objective:** Demonstrate common simulation failures and give some hints for interpreting, debugging, and fixing them.

```
[1]: %matplotlib inline
import os
import amici
from amici.petab_import import import_petab_problem
from amici.petab_objective import simulate_petab, RDATAS, EDATAS
from amici.plotting import plot_state_trajectories, plot_jacobian
import petab
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
from contextlib import suppress

try:
    import benchmark_models_petab
except ModuleNotFoundError:
    # install `benchmark_models_petab` if necessary
    %pip install -q -e "git+https://github.com/Benchmarking-Initiative/Benchmark-Models-
    ↪PEtab.git@master#subdirectory=src/python&egg=benchmark_models_petab"
    try:
        import benchmark_models_petab
    except ModuleNotFoundError:
        print("*** Please restart the kernel. ***")
```

## Overview

In the following, we will simulate models contained in the [PEtab Benchmark Collection](#) to demonstrate a number of simulation failures to analyze and fix them. We use the P<sub>ET</sub>ab format, as it makes model import and simulation much easier, but everything illustrated here, also applies to plain SBML or PySB import.

Note that, due to numerical issues, the examples below may not be fully reproducible on every system.

If any simulation failures occur, they will be printed via Python logging.

Programmatically, simulation success can be checked via `ReturnDataView.status`. In case of a successful simulation, and only then, this value corresponds to `amici.AMICI_SUCCESS`. In case of a simulation error, all quantities in `ReturnData/ReturnDataView` will be reported up to the time of failure, the rest will be NaN. The likelihood and it's gradient will always be NaN in case of failure.

### AMICI\_TOO\_MUCH\_WORK - mxstep steps taken before reaching tout

Let's run a simulation:

```
[2]: petab_problem = benchmark_models_petab.get_problem("Fujita_SciSignal2010")
    amici_model = import_petab_problem(
        petab_problem, verbose=False, force_compile=False
    )

    np.random.seed(2991)
    problem_parameters = dict(
        zip(
            petab_problem.x_free_ids,
            petab_problem.sample_parameter_startpoints(n_starts=1)[0],
        )
    )
    res = simulate_petab(
        petab_problem=petab_problem,
        amici_model=amici_model,
        problem_parameters=problem_parameters,
        scaled_parameters=True,
    )
    print(
        "Status:",
        [amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]],
    )
    assert [
        amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]
    ] == [
        "AMICI_SUCCESS",
        "AMICI_SUCCESS",
        "AMICI_SUCCESS",
        "AMICI_TOO_MUCH_WORK",
        "AMICI_NOT_RUN",
        "AMICI_NOT_RUN",
    ]
```

```

2023-11-21 19:19:25.079 - amici.swig_wrappers - DEBUG - [condition_step_03_0][CVODES:
↳ CNode:TOO_MUCH_WORK] AMICI ERROR: in module CVODES in function CNode : At t = 3031.8,
↳ mxstep steps taken before reaching tout.
2023-11-21 19:19:25.080 - amici.swig_wrappers - ERROR - [condition_step_03_0][FORWARD_
↳ FAILURE] AMICI forward simulation failed at t = 3031.8: AMICI failed to integrate the
↳ forward problem

Status: ['AMICI_SUCCESS', 'AMICI_SUCCESS', 'AMICI_SUCCESS', 'AMICI_TOO_MUCH_WORK',
↳ 'AMICI_NOT_RUN', 'AMICI_NOT_RUN']

```

### What happened?

AMICI failed to integrate the forward problem. The problem occurred for only one simulation condition, `condition_step_03_0`. The issue occurred at  $t = 3031.8$ , where the CVODES reached the maximum number of steps.

### How to address?

The number of steps the solver has to take is closely related to the chosen error tolerance. More accurate results, more steps. Therefore, this problem can be solved in two ways:

1. Increasing the maximum number of steps via `amici.Solver.setMaxSteps`. Note that this will increase the time required for simulation, and that simulation may still fail eventually. Sometimes it may be preferable to not increase this limit but rather fail fast. Also note that increasing the number of allowed steps increase RAM requirements (even if fewer steps are actually taken), so don't set this to ridiculously large values in order to avoid this error.
2. Reducing the number of steps CVODES has to take. This is determined by the required error tolerance. There are various solver error tolerances that can be adjusted. The most relevant ones are those controlled via `amici.Solver.setRelativeTolerance()` and `amici.Solver.setAbsoluteTolerance()`.

So, let's fix that:

```

[3]: # let's increase the allowed number of steps by 10x:
print("Increasing allowed number of steps ...")
amici_solver = amici_model.getSolver()
amici_solver.setMaxSteps(10 * amici_solver.getMaxSteps())

res = simulate_petab(
    petab_problem=petab_problem,
    amici_model=amici_model,
    problem_parameters=problem_parameters,
    scaled_parameters=True,
    solver=amici_solver,
)

print(
    "Status:",
    [amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]],
)
assert all(rdata.status == amici.AMICI_SUCCESS for rdata in res[RDATAS])
print("Simulations finished successfully.")
print()

# let's relax the relative error tolerance by a factor of 50

```

(continues on next page)



(continued from previous page)

```

print("Relaxing relative error tolerance ...")
amici_solver = amici_model.getSolver()
amici_solver.setRelativeTolerance(50 * amici_solver.getRelativeTolerance())

res = simulate_petab(
    petab_problem=petab_problem,
    amici_model=amici_model,
    problem_parameters=problem_parameters,
    scaled_parameters=True,
    solver=amici_solver,
)
print(
    "Status:",
    [amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]],
)
assert all(rdata.status == amici.AMICI_SUCCESS for rdata in res[RDATAS])
print("Simulations finished succesfully.")

```

Increasing allowed number of steps ...  
 Status: ['AMICI\_SUCCESS', 'AMICI\_SUCCESS', 'AMICI\_SUCCESS', 'AMICI\_SUCCESS', 'AMICI\_↵SUCCESS', 'AMICI\_SUCCESS']  
 Simulations finished succesfully.

Relaxing relative error tolerance ...  
 Status: ['AMICI\_SUCCESS', 'AMICI\_SUCCESS', 'AMICI\_SUCCESS', 'AMICI\_SUCCESS', 'AMICI\_↵SUCCESS', 'AMICI\_SUCCESS']  
 Simulations finished succesfully.

Internal  $t = [...]$  and  $h = [...]$  are such that  $t + h = t$  on the next step

Let's run a simulation:

```

[4]: petab_problem = benchmark_models_petab.get_problem("Crauste_CellSystems2017")
amici_model = import_petab_problem(petab_problem, verbose=False)

np.random.seed(1)
problem_parameters = dict(
    zip(
        petab_problem.x_free_ids,
        petab_problem.sample_parameter_startpoints(n_starts=1)[0],
    )
)
res = simulate_petab(
    petab_problem=petab_problem,
    amici_model=amici_model,
    problem_parameters=problem_parameters,
    scaled_parameters=True,
)
print(
    "Status:",
    [amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]],
)

```

(continues on next page)

(continued from previous page)

```

)
assert [
    amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]
] == ["AMICI_TOO_MUCH_WORK"]

```

2023-11-21 19:19:25.674 - amici.swig\_wrappers - DEBUG - [model1\_data1][CVODES:Cvode:  
 ↳WARNING] AMICI ERROR: in module CVODES in function Cvode : Internal t = 0.330112 and h\_  
 ↳= 2.06577e-17 are such that t + h = t on the next step. The solver will continue\_  
 ↳anyway.

2023-11-21 19:19:25.675 - amici.swig\_wrappers - DEBUG - [model1\_data1][CVODES:Cvode:  
 ↳WARNING] AMICI ERROR: in module CVODES in function Cvode : Internal t = 0.330112 and h\_  
 ↳= 2.06577e-17 are such that t + h = t on the next step. The solver will continue\_  
 ↳anyway.

2023-11-21 19:19:25.676 - amici.swig\_wrappers - DEBUG - [model1\_data1][CVODES:Cvode:  
 ↳WARNING] AMICI ERROR: in module CVODES in function Cvode : Internal t = 0.330112 and h\_  
 ↳= 2.06577e-17 are such that t + h = t on the next step. The solver will continue\_  
 ↳anyway.

2023-11-21 19:19:25.677 - amici.swig\_wrappers - DEBUG - [model1\_data1][CVODES:Cvode:  
 ↳WARNING] AMICI ERROR: in module CVODES in function Cvode : Internal t = 0.330112 and h\_  
 ↳= 2.06577e-17 are such that t + h = t on the next step. The solver will continue\_  
 ↳anyway.

2023-11-21 19:19:25.678 - amici.swig\_wrappers - DEBUG - [model1\_data1][CVODES:Cvode:  
 ↳WARNING] AMICI ERROR: in module CVODES in function Cvode : Internal t = 0.330112 and h\_  
 ↳= 2.06577e-17 are such that t + h = t on the next step. The solver will continue\_  
 ↳anyway.

2023-11-21 19:19:25.679 - amici.swig\_wrappers - DEBUG - [model1\_data1][CVODES:Cvode:  
 ↳WARNING] AMICI ERROR: in module CVODES in function Cvode : Internal t = 0.330112 and h\_  
 ↳= 2.06577e-17 are such that t + h = t on the next step. The solver will continue\_  
 ↳anyway.

2023-11-21 19:19:25.680 - amici.swig\_wrappers - DEBUG - [model1\_data1][CVODES:Cvode:  
 ↳WARNING] AMICI ERROR: in module CVODES in function Cvode : Internal t = 0.330112 and h\_  
 ↳= 2.06577e-17 are such that t + h = t on the next step. The solver will continue\_  
 ↳anyway.

2023-11-21 19:19:25.681 - amici.swig\_wrappers - DEBUG - [model1\_data1][CVODES:Cvode:  
 ↳WARNING] AMICI ERROR: in module CVODES in function Cvode : Internal t = 0.330112 and h\_  
 ↳= 2.06577e-17 are such that t + h = t on the next step. The solver will continue\_  
 ↳anyway.

2023-11-21 19:19:25.682 - amici.swig\_wrappers - DEBUG - [model1\_data1][CVODES:Cvode:  
 ↳WARNING] AMICI ERROR: in module CVODES in function Cvode : Internal t = 0.330112 and h\_  
 ↳= 2.06577e-17 are such that t + h = t on the next step. The solver will continue\_  
 ↳anyway.

2023-11-21 19:19:25.683 - amici.swig\_wrappers - DEBUG - [model1\_data1][CVODES:Cvode:  
 ↳WARNING] AMICI ERROR: in module CVODES in function Cvode : Internal t = 0.330112 and h\_  
 ↳= 2.06577e-17 are such that t + h = t on the next step. The solver will continue\_  
 ↳anyway.

2023-11-21 19:19:25.683 - amici.swig\_wrappers - DEBUG - [model1\_data1][CVODES:Cvode:  
 ↳WARNING] AMICI ERROR: in module CVODES in function Cvode : The above warning has been\_  
 ↳issued mxhnil times and will not be issued again for this problem.

2023-11-21 19:19:25.684 - amici.swig\_wrappers - DEBUG - [model1\_data1][CVODES:Cvode:TOO\_  
 ↳MUCH\_WORK] AMICI ERROR: in module CVODES in function Cvode : At t = 0.330112, mxstep\_  
 ↳steps taken before reaching tout.

2023-11-21 19:19:25.685 - amici.swig\_wrappers - ERROR - [model1\_data1][FORWARD\_FAILURE]\_  
 ↳AMICI forward simulation failed at t = 0.330112: AMICI failed to integrate the forward\_  
 ↳problem

(continues on next page)

(continued from previous page)

```
Status: ['AMICI_TOO_MUCH_WORK']
```

### What happened?

The forward simulation failed because the AMICI solver exceeded the maximum number of steps. Unlike in the previous case of `mxstep` steps taken before reaching `tout` (see above), here we got several additional warnings that the current step size  $h$  is numerically zero.

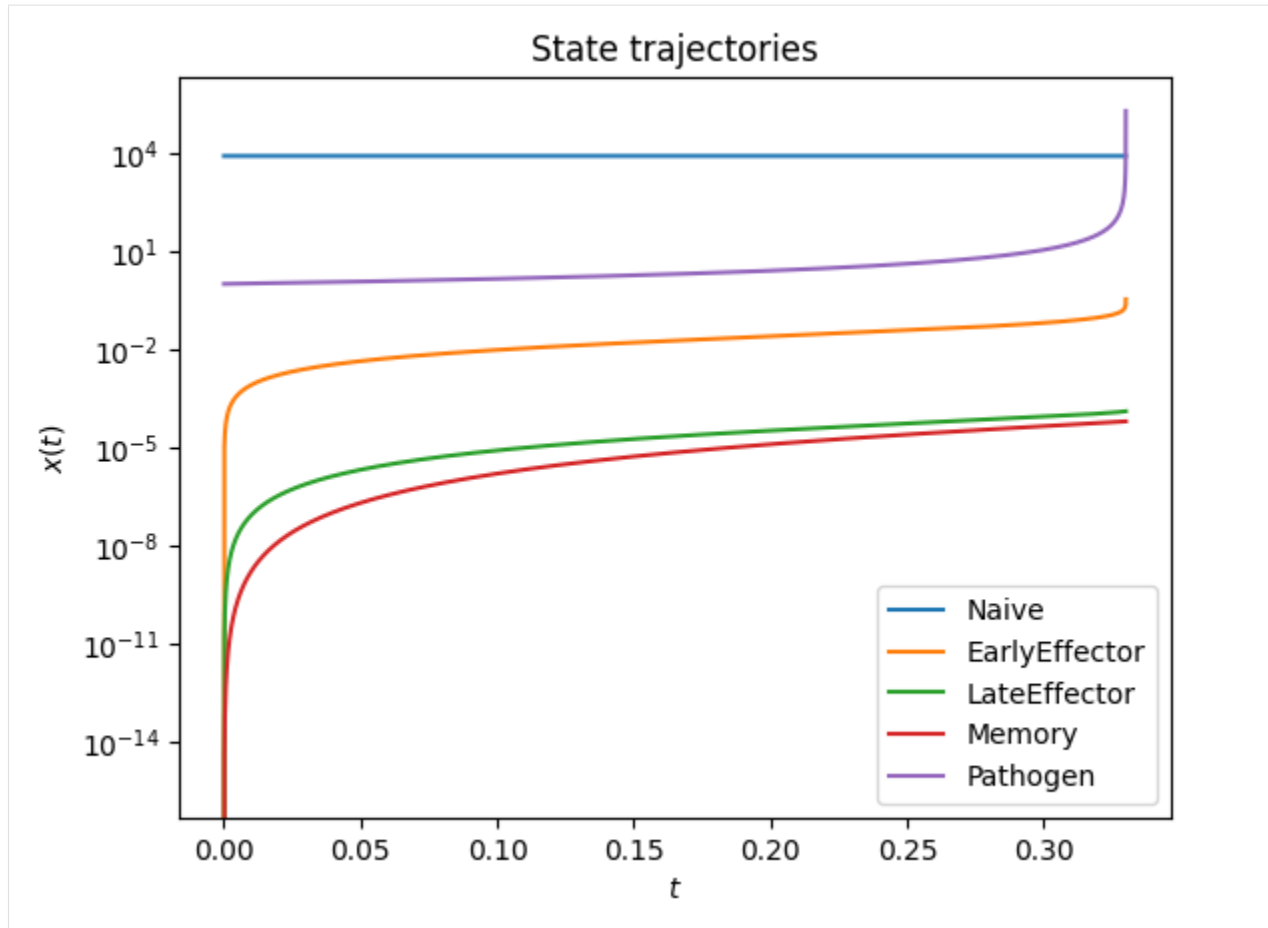
### How to address?

The warning `Internal t = [...] and h = [...] are such that t + h = t on the next step` tells us that the solver is not able to move forward. The solver may be able to recover from that, but not always.

Let's look at the state trajectories to see what's going on. Such a tiny step size is usually related to very fast dynamics. We repeat the simulation with additional timepoints before the point of failure:

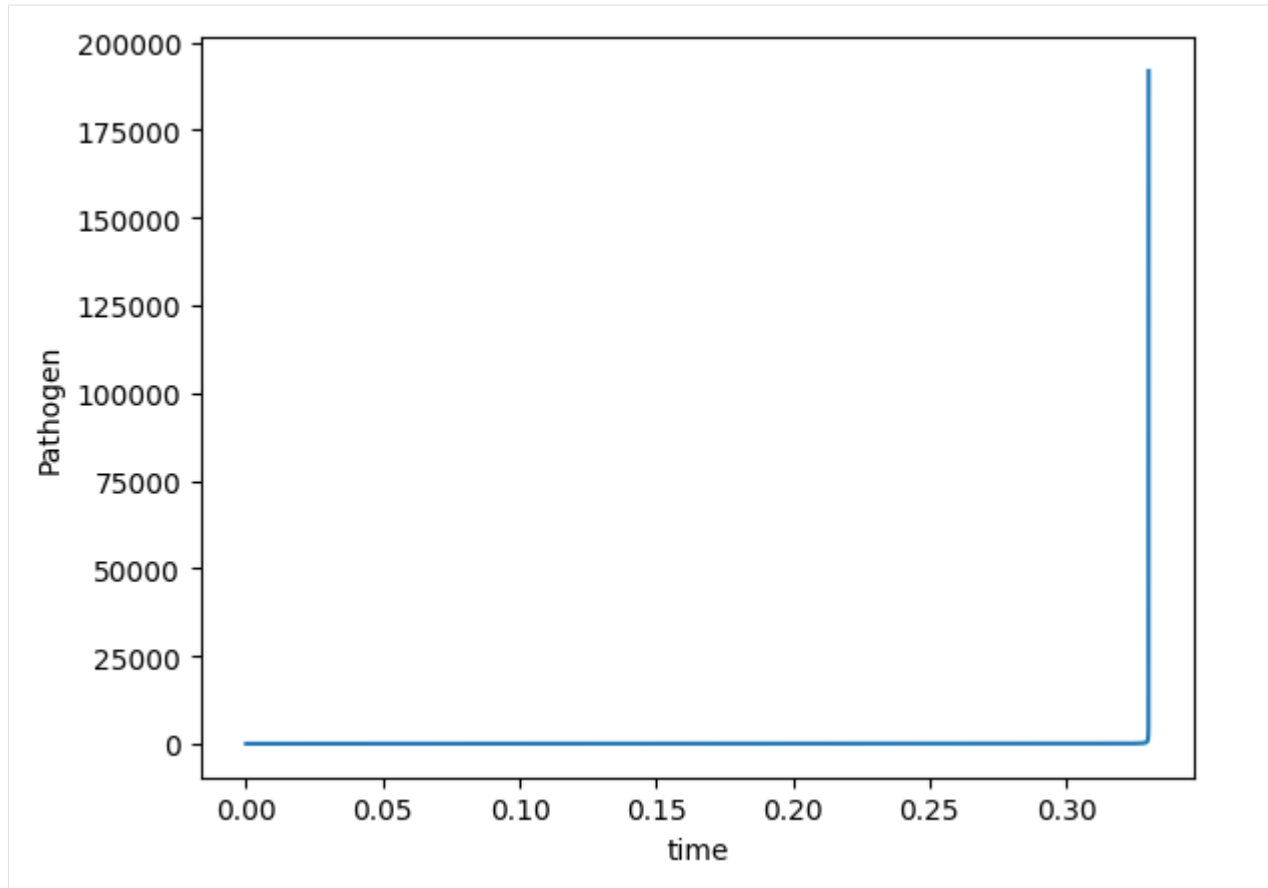
```
[5]: # Create a copy of this simulation condition
edata = amici.ExpData(res[EDATAS][0])
edata.setTimepoints(np.linspace(0, 0.33011, 5000))
amici_solver = amici_model.getSolver()
rdata = amici.runAmiciSimulation(amici_model, amici_solver, edata)

# Visualize state trajectories
plot_state_trajectories(rdata, model=amici_model)
plt.yscale("log")
```



We can see a steep increase for Pathogen just before the error occurs. Let's zoom in:

```
[6]: plt.plot(rdata.t, rdata.by_id("Pathogen"))
plt.xlabel("time")
plt.ylabel("Pathogen");
```



The solver is unable to handle such a steep increase. There is not much we can do. Increasing the tolerances will let the solver proceed a bit further, but this is usually not enough. Most likely there is a problem in the model or in the choice of parameter values.

**the error test failed repeatedly or with  $|h| = h_{\min}$**

Let's run a simulation:

```
[7]: petab_problem = benchmark_models_petab.get_problem("Fujita_SciSignal2010")
    amici_model = import_petab_problem(petab_problem, verbose=False)

    np.random.seed(4920)
    problem_parameters = dict(
        zip(
            petab_problem.x_free_ids,
            petab_problem.sample_parameter_startpoints(n_starts=1)[0],
        )
    )
    res = simulate_petab(
        petab_problem=petab_problem,
        amici_model=amici_model,
        problem_parameters=problem_parameters,
        scaled_parameters=True,
    )
```

(continues on next page)

(continued from previous page)

```

print(
    "Status:",
    [amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]],
)

assert [
    amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]
] == [
    "AMICI_SUCCESS",
    "AMICI_ERR_FAILURE",
    "AMICI_NOT_RUN",
    "AMICI_NOT_RUN",
    "AMICI_NOT_RUN",
    "AMICI_NOT_RUN",
]

```

```

2023-11-21 19:19:27.007 - amici.swig_wrappers - DEBUG - [condition_step_00_3][CVODES:
↳ CNode:ERR_FAILURE] AMICI ERROR: in module CVODES in function CNode : At t = 429.232,
↳ and h = 7.75194e-05, the error test failed repeatedly or with |h| = hmin.
2023-11-21 19:19:27.008 - amici.swig_wrappers - ERROR - [condition_step_00_3][FORWARD_
↳ FAILURE] AMICI forward simulation failed at t = 429.232: AMICI failed to integrate the
↳ forward problem

```

```

Status: ['AMICI_SUCCESS', 'AMICI_ERR_FAILURE', 'AMICI_NOT_RUN', 'AMICI_NOT_RUN', 'AMICI_
↳ NOT_RUN', 'AMICI_NOT_RUN']

```

### What happened?

AMICI failed to integrate the forward problem. The problem occurred for only one simulation condition, `condition_step_00_3`. The issue occurred at  $t = 429.232$ , where the error test failed. This means, the solver is unable to take a step of non-zero size without violating the chosen error tolerances.

### How to address?

The step size is computed based on the Jacobian. Inspecting `ReturnData.J` shows us that we have rather large values in the Jacobian:

```

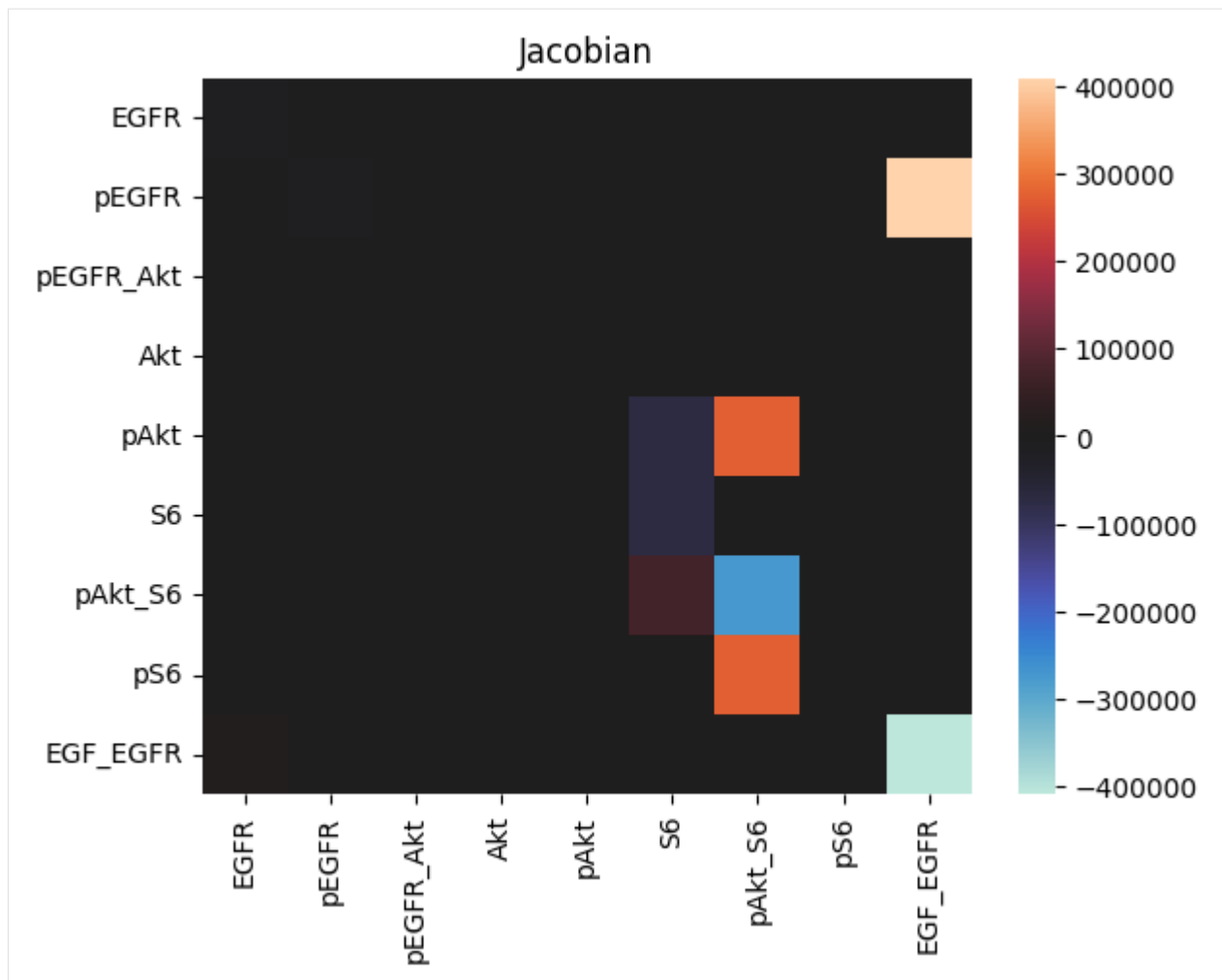
[8]: rdata = res[RDATAS][1]

# Show Jacobian as heatmap
plot_jacobian(rdata)

print(f"largest absolute Jacobian value: {np.max(np.abs(rdata.J)):.3g}")

largest absolute Jacobian value: 4.09e+05

```



In this case, the default relative error tolerance may be too high and lead too large absolute errors.

Let's retry simulation using stricter tolerances:

```
[9]: # set stricter relative error tolerance
amici_solver = amici_model.getSolver()
amici_solver.setRelativeTolerance(amici_solver.getRelativeTolerance() / 10)

res = simulate_petab(
    petab_problem=petab_problem,
    amici_model=amici_model,
    problem_parameters=problem_parameters,
    scaled_parameters=True,
    solver=amici_solver,
)
print(
    "Status:",
    [amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]],
)
assert all(rdata.status == amici.AMICI_SUCCESS for rdata in res[RDATAS])
print("Simulations finished successfully.")
```

```
Status: ['AMICI_SUCCESS', 'AMICI_SUCCESS', 'AMICI_SUCCESS', 'AMICI_SUCCESS', 'AMICI_
→SUCCESS', 'AMICI_SUCCESS']
Simulations finished succesfully.
```

### Cvode routine CVode returned a root after reinitialization

Let's run a simulation:

```
[10]: petab_problem = benchmark_models_petab.get_problem("Weber_BMC2015")
      amici_model = import_petab_problem(
          petab_problem, verbose=False, force_compile=False
      )

      np.random.seed(4)
      problem_parameters = dict(
          zip(
              petab_problem.x_free_ids,
              petab_problem.sample_parameter_startpoints(n_starts=1)[0],
          )
      )
      res = simulate_petab(
          petab_problem=petab_problem,
          amici_model=amici_model,
          problem_parameters=problem_parameters,
          scaled_parameters=True,
      )
      print(
          "Status:",
          [amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]],
      )
      assert [
          amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]
      ] == [
          "AMICI_ERROR",
          "AMICI_NOT_RUN",
      ]
```

```
2023-11-21 19:19:27.852 - amici.swig_wrappers - ERROR - [model1_data2+model1_
→data1][OTHER] AMICI simulation failed: Cvode routine CVode returned a root after_
→reinitialization. The initial step-size after the event or heaviside function is too_
→small. To fix this, increase absolute and relative tolerances! failed with error code 2
2023-11-21 19:19:27.852 - amici.swig_wrappers - DEBUG - [model1_data2+model1_
→data1][BACKTRACE] The previous error occurred at:
2      0x7f1bb4daeb98 amici::CvodeException::CvodeException(int, char const*) + 24
3      0x7f1bb4d9b82f /home/docs/checkouts/readthedocs.org/user_builds/amici/checkouts/
→v0.20.0/documentation/amici_models/Weber_BMC2015/Weber_BMC2015/_Weber_BMC2015.cpython-
→39-x86_64-linux-gnu.so(+0x4b82f) [0x7f1bb4d9b82f]
4      0x7f1bb4de9926 amici::CvodeSolver::solve(double, int) const + 86
5      0x7f1bccfce626 amici::Solver::run(double) const + 198
6      0x7f1bcd00af70 amici::ForwardProblem::workForward
```

```
Status: ['AMICI_ERROR', 'AMICI_NOT_RUN']
```

What happened?



The simulation failed because the initial step-size after an event or heaviside function was too small. The error occurred during simulation of condition `model1_data1` after successful preequilibration (`model1_data2`).

### How to address?

The error message already suggests a fix for this situation, so let's try increasing the relative tolerance:

```
[11]: amici_solver = amici_model.getSolver()
      amici_solver.setRelativeTolerance(200 * amici_solver.getRelativeTolerance())

      np.random.seed(4)
      problem_parameters = dict(
          zip(
              petab_problem.x_free_ids,
              petab_problem.sample_parameter_startpoints(n_starts=1)[0],
          )
      )
      res = simulate_petab(
          petab_problem=petab_problem,
          amici_model=amici_model,
          problem_parameters=problem_parameters,
          scaled_parameters=True,
          solver=amici_solver,
      )
      print(
          "Status:",
          [amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]],
      )
      assert all(rdata.status == amici.AMICI_SUCCESS for rdata in res[RDATAS])

      Status: ['AMICI_SUCCESS', 'AMICI_SUCCESS']
```

### AMICI encountered a NaN / Inf value for [...]

Let's run a simulation:

```
[12]: petab_problem = benchmark_models_petab.get_problem("Borghans_BiophysChem1997")
      amici_model = import_petab_problem(petab_problem, verbose=False)

      np.random.seed(18)
      problem_parameters = dict(
          zip(
              petab_problem.x_free_ids,
              petab_problem.sample_parameter_startpoints(n_starts=1)[0],
          )
      )
      res = simulate_petab(
          petab_problem=petab_problem,
          amici_model=amici_model,
          problem_parameters=problem_parameters,
          scaled_parameters=True,
      )
      print(
          "Status:",
```

(continues on next page)

(continued from previous page)

```
[amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]],
)
assert [
    amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]
] == ["AMICI_FIRST_RHSFUNC_ERR"]
```

2023-11-21 19:19:28.274 - amici.swig\_wrappers - WARNING - [model1\_data1][AMICI:NaN]␣  
 ↳AMICI encountered a NaN value for xdot[2] (A\_state)  
 2023-11-21 19:19:28.275 - amici.swig\_wrappers - WARNING - [model1\_data1][AMICI:NaN]␣  
 ↳AMICI encountered a NaN value for w[6] (flux\_v7\_v\_6)  
 2023-11-21 19:19:28.276 - amici.swig\_wrappers - DEBUG - [model1\_data1][CVODES:CVode:  
 ↳FIRST\_RHSFUNC\_ERR] AMICI ERROR: in module CVODES in function CVode : The right-hand␣  
 ↳side routine failed at the first call.  
 2023-11-21 19:19:28.277 - amici.swig\_wrappers - ERROR - [model1\_data1][FORWARD\_FAILURE]␣  
 ↳AMICI forward simulation failed at t = 0: AMICI failed to integrate the forward problem

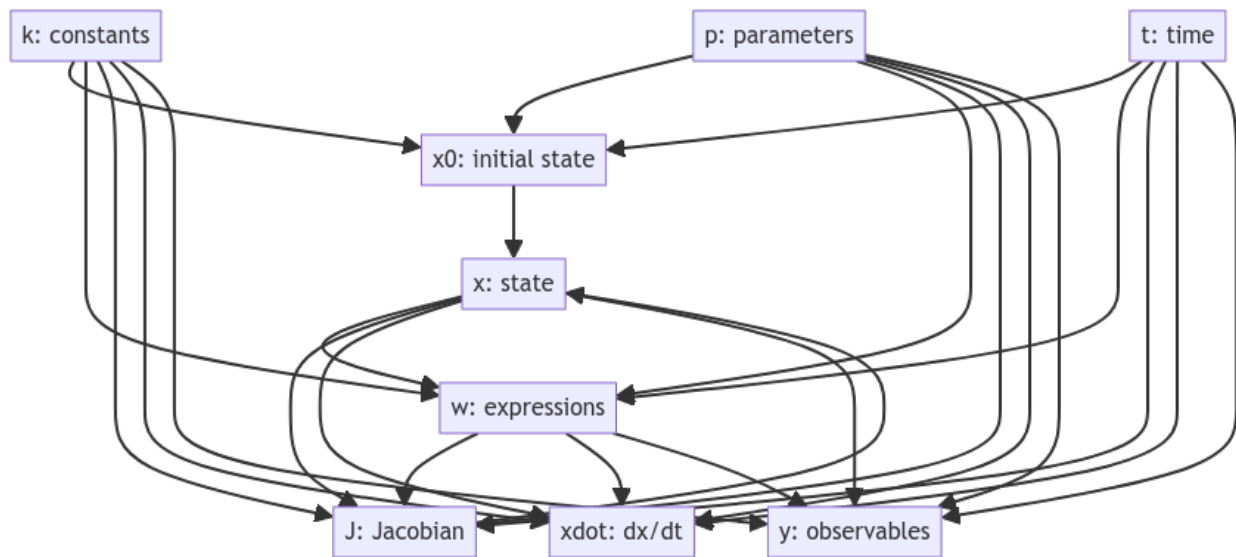
Status: ['AMICI\_FIRST\_RHSFUNC\_ERR']

### What happened?

The forward simulation failed because AMICI encountered a NaN value when simulating condition `model1_data1`. Then NaNs occurred in  $\dot{x}$  and  $w$  (model expressions, such as reaction fluxes or assignment rules). Furthermore, the failure occurred at the first call, so at  $t = t_0$  (here:  $t = 0$ ).

### How to address?

The NaN in  $\dot{x}$  is most likely a consequence of the one in  $w$ . (A subset of) the dependency tree looks something like:



Always look for the most basic (furthest up) model quantities. In cases where non-finite values occur in expressions further down, rerunning the simulation after calling `Model.setAlwaysCheckFinite(True)` may give some further hints on where the issue originates.

The NaN in  $w$  occurred for `flux_v7_v_6` (see error log), i.e., when computing the reaction flux for reaction `v7_v_6`. As  $w$  only depends on  $(t, p, k, x)$  and no non-finite values have been reported for those, the issue has to be in the respective flux equation.

Let's look at that expression. This can either be done by inspecting the underlying SBML model (e.g., using COPASI), or by checking the generated model code:

```
[13]: # model source code location
model_src_dir = Path(amici_model.module.__file__).parents[1]

# find the problematic expression in the model source code
!grep flux_v7_v_6 {model_src_dir}/w.cpp

    flux_v7_v_6 = 1.0*std::pow(A_state, 2)*Vd*std::pow(Z_state, n_par)/((std::pow(A_
↪state, 2) + std::pow(Kp, 2))*(std::pow(Kd, n_par) + std::pow(Z_state, n_par))); //
↪w[6]
```

What could go wrong? We can obtain NaN from any of these symbols being NaN, or through division by zero.

Let's let's check the denominator first:

$$(A_{state}^2 + Kp^2) * (Kd^{n_{par}} + Z_{state}^{n_{par}})$$

A\_state and Z\_state are state variables, Kd, K\_p, and n\_par are parameters.

As the error occurred at  $t = t_0$ , let's ensure the initial state is non-zero and finite:

```
[14]: rdata = res[RDATAS][0]
edata = res[EDATAS][0]
# check initial states
x0 = dict(zip(amici_model.getStateIds(), rdata.x0))
print(f"{x0=}")

x0={'Z_state': 0.6869701913398437, 'Y_state': 0.2977237418558598, 'A_state': 0.
↪1116031306650328}
```

The initial states are fine - the first multiplicand is non-zero, as  $x_0$  was non-zero.

So let's check the parameter values occurring in the second multiplicand:

```
[15]: # we have to account for the chosen parameter scale
from itertools import starmap

unscaled_parameter = dict(
    zip(
        amici_model.getParameterIds(),
        starmap(
            amici.getUnscaledParameter, zip(edata.parameters, edata.pscale)
        ),
    )
)
print(dict((p, unscaled_parameter[p]) for p in ("Kd", "Kp", "n_par")))

{'Kd': 0.028491925689008366, 'Kp': 1002.513636749445, 'n_par': 7816.430091706722}
```

Considering that n\_par occurs as exponent, it's magnitude looks pretty high. This term is very likely causing the problem - let's check:

```
[16]: print(
    f"{x0['Z_state']**unscaled_parameter['n_par'] + unscaled_parameter['Kd']**unscaled_
↪parameter['n_par']=}"
)
```

```
x0['Z_state']**unscaled_parameter['n_par'] + unscaled_parameter['Kd']**unscaled_
↪parameter['n_par']=0.0
```

Indeed, no way we can fix this for the given model. This was most likely an unrealistic parameter value, originating from a too high upper parameter bound for `n_par`. Therefore, if this error occurs during optimization, a first step could be adapting the respective parameter bounds. In other cases, this may be a result of unfortunate arrangement of model expressions, which can sometimes be solved by passing a suitable simplification function to the model import.

### Steady state sensitivity computation failed due to unsuccessful factorization of RHS Jacobian

Let's run a simulation:

```
[17]: petab_problem = benchmark_models_petab.get_problem("Blasi_CellSystems2016")
with suppress(KeyError):
    del os.environ["AMICI_EXPERIMENTAL_SBML_NONCONST_CLS"]
amici_model = import_petab_problem(
    petab_problem,
    verbose=False,
    force_compile=True,
    model_name="Blasi_CellSystems2016_1",
)

amici_solver = amici_model.getSolver()
amici_solver.setSensitivityMethod(amici.SensitivityMethod.forward)
amici_solver.setSensitivityOrder(amici.SensitivityOrder.first)
amici_model.setSteadyStateSensitivityMode(
    amici.SteadyStateSensitivityMode.newtonOnly
)

np.random.seed(2020)
problem_parameters = dict(
    zip(
        petab_problem.x_free_ids,
        petab_problem.sample_parameter_startpoints(n_starts=1)[0],
    )
)
res = simulate_petab(
    petab_problem=petab_problem,
    amici_model=amici_model,
    problem_parameters=problem_parameters,
    scaled_parameters=True,
    solver=amici_solver,
)
print(
    "Status:",
    [amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]],
)

# hard to reproduce on GHA
if os.getenv("GITHUB_ACTIONS") is None:
    assert [
```

(continues on next page)

(continued from previous page)

```

    amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]
] == ["AMICI_ERROR"]

```

Visualization table not available. Skipping.

```

2023-11-21 19:19:53.041 - amici.swig_wrappers - ERROR - [control][OTHER] AMICI_
↳simulation failed: Steady state sensitivity computation failed due to unsuccessful_
↳factorization of RHS Jacobian
2023-11-21 19:19:53.042 - amici.swig_wrappers - DEBUG - [control][BACKTRACE] The_
↳previous error occurred at:
2      0x7f1bcc6cf87 /home/docs/checkouts/readthedocs.org/user_builds/amici/checkouts/
↳v0.20.0/python/sdist/amici/_amici.cpython-39-x86_64-linux-gnu.so(+0xc6cf87)_
↳[0x7f1bcc6cf87]
3      0x7f1bccfbbaad amici::runAmiciSimulation(amici::Solver&, amici::ExpData const*,_
↳amici::Model&, bool) + 3341
4      0x7f1bccfbb527 /home/docs/checkouts/readthedocs.org/user_builds/amici/checkouts/
↳v0.20.0/python/sdist/amici/_amici.cpython-39-x86_64-linux-gnu.so(+0x1bb527)_
↳[0x7f1bccfbb527]
5      0x7f1bccd87a16

```

```
Status: ['AMICI_ERROR']
```

### What happened?

AMICI failed to compute steady-state sensitivities, because it was not able to factorize the Jacobian.

### How to address?

This is most likely a result of a singular Jacobian. Let's check the condition number:

```
[18]: rdata = res[RDATAS][0]
      np.linalg.cond(rdata.J)
```

```
[18]: 3.01187419527941e+19
```

Indeed, the condition number shows that the Jacobian is numerically singular. If this happens consistently, it is usually due to conserved quantities in the model.

There are two ways we can address that:

1. Use numerical integration to compute sensitivities, for which a singular Jacobian is not an issue. This is, usually, slower, though.
2. Remove any conserved quantities.

Let's try both approaches:

```
[19]: # use numerical integration
      amici_model.setSteadyStateSensitivityMode(
          amici.SteadyStateSensitivityMode.integrationOnly
      )

      res = simulate_petab(
          petab_problem=petab_problem,
          amici_model=amici_model,
          problem_parameters=problem_parameters,
          scaled_parameters=True,
          solver=amici_solver,

```

(continues on next page)

(continued from previous page)

```

)
print(
    "Status:",
    [amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]],
)
assert all(rdata.status == amici.AMICI_SUCCESS for rdata in res[RDATAS])

```

Status: ['AMICI\_SUCCESS']

```

[20]: # Remove conserved quantities - this requires re-importing the model

# this is enabled by the `AMICI_EXPERIMENTAL_SBML_NONCONST_CLS` environment variable
os.environ["AMICI_EXPERIMENTAL_SBML_NONCONST_CLS"] = "1"
amici_model = import_petab_problem(
    petab_problem,
    verbose=False,
    # we need a different model name if we import the model again
    # we cannot load a model with the same name as an already loaded model
    model_name="Blasi_CellSystems2016_2",
    force_compile=True,
)
del os.environ["AMICI_EXPERIMENTAL_SBML_NONCONST_CLS"]

amici_solver = amici_model.getSolver()
amici_solver.setSensitivityMethod(amici.SensitivityMethod.forward)
amici_solver.setSensitivityOrder(amici.SensitivityOrder.first)

res = simulate_petab(
    petab_problem=petab_problem,
    amici_model=amici_model,
    problem_parameters=problem_parameters,
    scaled_parameters=True,
    solver=amici_solver,
)
print(
    "Status:",
    [amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]],
)
assert all(rdata.status == amici.AMICI_SUCCESS for rdata in res[RDATAS])

```

Visualization table not available. Skipping.

Status: ['AMICI\_SUCCESS']

## Steady state computation failed

Let's run a simulation:

```
[21]: petab_problem = benchmark_models_petab.get_problem("Brannmark_JBC2010")
      amici_model = import_petab_problem(
          petab_problem,
          verbose=False,
      )

      amici_solver = amici_model.getSolver()

      np.random.seed(1851)
      problem_parameters = dict(
          zip(
              petab_problem.x_free_ids,
              petab_problem.sample_parameter_startpoints(n_starts=1)[0],
          )
      )
      res = simulate_petab(
          petab_problem=petab_problem,
          amici_model=amici_model,
          problem_parameters=problem_parameters,
          scaled_parameters=True,
          solver=amici_solver,
      )

      print(
          "Status:",
          [amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]],
      )

      # hard to reproduce on GH
      if os.getenv("GITHUB_ACTIONS") is None:
          assert [
              amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]
          ] == [
              "AMICI_ERROR",
              "AMICI_NOT_RUN",
              "AMICI_NOT_RUN",
              "AMICI_NOT_RUN",
              "AMICI_NOT_RUN",
              "AMICI_NOT_RUN",
              "AMICI_NOT_RUN",
              "AMICI_NOT_RUN",
              "AMICI_NOT_RUN",
          ]

2023-11-21 19:20:18.733 - amici.swig_wrappers - DEBUG - [Dose_0+Dose_0][CVODES:CNode:ERR_
↳ FAILURE] AMICI ERROR: in module CVODES in function CNode : At t = 736.713 and h = 0.
↳ 000145837, the error test failed repeatedly or with |h| = hmin.
2023-11-21 19:20:18.734 - amici.swig_wrappers - DEBUG - [Dose_0+Dose_0][OTHER] AMICI_
↳ equilibration failed at t=736.713.
2023-11-21 19:20:18.735 - amici.swig_wrappers - ERROR - [Dose_0+Dose_0][OTHER] AMICI_
↳ simulation failed: Steady state computation failed. Simulation to steady state failed
(see page)
```

(continued from previous page)

```

2023-11-21 19:20:18.736 - amici.swig_wrappers - DEBUG - [Dose_0+Dose_0][BACKTRACE] The
↳previous error occurred at:
2      0x7f1bcd0100b2 amici::SteadystateProblem::handleSteadyStateFailure(bool, bool,
↳bool) + 146
3      0x7f1bcd01084c amici::SteadystateProblem::findSteadyState(amici::Solver const&,
↳amici::Model&, int) + 524
4      0x7f1bcd01089f amici::SteadystateProblem::workSteadyStateProblem(amici::Solver
↳const&, amici::Model&, int) + 63
5      0x7f1bccfb9f1b amici::runAmiciSimulation(amici::Solver&, amici::ExpData const*,
↳amici::Model&, bool) + 379
6      0x7f1bccfbb527 /home/docs/checkouts/readthe

```

```

Status: ['AMICI_ERROR', 'AMICI_NOT_RUN', 'AMICI_NOT_RUN', 'AMICI_NOT_RUN', 'AMICI_NOT_RUN
↳', 'AMICI_NOT_RUN', 'AMICI_NOT_RUN', 'AMICI_NOT_RUN']

```

### What happened?

All given experimental conditions require pre-equilibration, i.e., finding a steady state. AMICI first tries to find a steady state using the Newton solver, if that fails, it tries simulating until steady state, if that also fails, it tries the Newton solver from the end of the simulation. In this case, all three failed. Neither Newton's method nor simulation yielded a steady state satisfying the required tolerances.

This can also be seen in `ReturnDataView.preeq_status` (the three statuses corresponds to Newton #1, Simulation, Newton #2):

```

[22]: rdata = res[RDATAS][0]
      list(map(amici.SteadyStateStatus, rdata.preeq_status.flatten()))

[22]: [<SteadyStateStatus.not_run: 0>,
      <SteadyStateStatus.failed: -1>,
      <SteadyStateStatus.not_run: 0>]

```

### How to address?

There are several ways to address that:

1. Stricter integration tolerances (preferred if affordable - higher accuracy, but generally slower)
2. Looser steady-state tolerances (lower accuracy, generally faster)
3. Increase the number of allowed steps for Newton's method

Let's try all of them:

```

[23]: # Reduce relative tolerance for integration
      amici_solver = amici_model.getSolver()
      amici_solver.setRelativeTolerance(
          1 / 100 * amici_solver.getRelativeTolerance()
      )

      res = simulate_petab(
          petab_problem=petab_problem,
          amici_model=amici_model,
          problem_parameters=problem_parameters,
          scaled_parameters=True,
          solver=amici_solver,
      )

```

(continues on next page)



(continued from previous page)

```

print(
    "status:",
    [amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]],
)

rdata = res[RDATAS][0]
print(
    f"preeq_status={list(map(amici.SteadyStateStatus, rdata.preeq_status.flatten()))}"
)
print(f"{rdata.preeq_numsteps=}")

# hard to reproduce on GHA
if os.getenv("GITHUB_ACTIONS") is None:
    assert all(rdata.status == amici.AMICI_SUCCESS for rdata in res[RDATAS])

status: ['AMICI_SUCCESS', 'AMICI_SUCCESS', 'AMICI_SUCCESS', 'AMICI_SUCCESS', 'AMICI_
→SUCCESS', 'AMICI_SUCCESS', 'AMICI_SUCCESS', 'AMICI_SUCCESS']
preeq_status=[<SteadyStateStatus.not_run: 0>, <SteadyStateStatus.success: 1>,
→<SteadyStateStatus.not_run: 0>]
rdata.preeq_numsteps=array([[ 0, 3150, 0]])

```

```

[24]: # Increase relative steady state tolerance
for log10_relaxation_factor in range(1, 10):
    print(f"Relaxing tolerances by factor {10 ** log10_relaxation_factor}")
    amici_solver = amici_model.getSolver()
    amici_solver.setRelativeToleranceSteadyState(
        amici_solver.getRelativeToleranceSteadyState()
        * 10**log10_relaxation_factor
    )

    res = simulate_petab(
        petab_problem=petab_problem,
        amici_model=amici_model,
        problem_parameters=problem_parameters,
        scaled_parameters=True,
        solver=amici_solver,
    )
    if all(rdata.status == amici.AMICI_SUCCESS for rdata in res[RDATAS]):
        print(
            f"-> Succeeded with relative steady state tolerance {amici_solver.
→getRelativeToleranceSteadyState()}\n"
        )
        break
    else:
        print("-> Failed.\n")

print(
    "status:",
    [amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]],
)

rdata = res[RDATAS][0]

```

(continues on next page)

(continued from previous page)

```

print(
    f"preeq_status={list(map(amici.SteadyStateStatus, rdata.preeq_status.flatten()))}"
)
print(f"{rdata.preeq_numsteps=}")
assert all(rdata.status == amici.AMICI_SUCCESS for rdata in res[RDATAS])

```

Relaxing tolerances by factor 10

```

2023-11-21 19:20:19.195 - amici.swig_wrappers - DEBUG - [Dose_0+Dose_0][CVODES:CNode:ERR_
↳FAILURE] AMICI ERROR: in module CVODES in function CNode : At t = 736.713 and h = 0.
↳000145837, the error test failed repeatedly or with |h| = hmin.
2023-11-21 19:20:19.196 - amici.swig_wrappers - DEBUG - [Dose_0+Dose_0][OTHER] AMICI_
↳equilibration failed at t=736.713.
2023-11-21 19:20:19.197 - amici.swig_wrappers - ERROR - [Dose_0+Dose_0][OTHER] AMICI_
↳simulation failed: Steady state computation failed. Simulation to steady state failed.
2023-11-21 19:20:19.198 - amici.swig_wrappers - DEBUG - [Dose_0+Dose_0][BACKTRACE] The_
↳previous error occurred at:
2      0x7f1bcd0100b2 amici::SteadystateProblem::handleSteadyStateFailure(bool, bool, _
↳bool) + 146
3      0x7f1bcd01084c amici::SteadystateProblem::findSteadyState(amici::Solver const&, _
↳amici::Model&, int) + 524
4      0x7f1bcd01089f amici::SteadystateProblem::workSteadyStateProblem(amici::Solver_
↳const&, amici::Model&, int) + 63
5      0x7f1bccfb9f1b amici::runAmiciSimulation(amici::Solver&, amici::ExpData const*, _
↳amici::Model&, bool) + 379
6      0x7f1bccfbb527 /home/docs/checkouts/readthe

```

-> Failed.

Relaxing tolerances by factor 100

```

2023-11-21 19:20:19.389 - amici.swig_wrappers - DEBUG - [Dose_0+Dose_0][CVODES:CNode:ERR_
↳FAILURE] AMICI ERROR: in module CVODES in function CNode : At t = 736.713 and h = 0.
↳000145837, the error test failed repeatedly or with |h| = hmin.
2023-11-21 19:20:19.390 - amici.swig_wrappers - DEBUG - [Dose_0+Dose_0][OTHER] AMICI_
↳equilibration failed at t=736.713.
2023-11-21 19:20:19.391 - amici.swig_wrappers - ERROR - [Dose_0+Dose_0][OTHER] AMICI_
↳simulation failed: Steady state computation failed. Simulation to steady state failed.
2023-11-21 19:20:19.392 - amici.swig_wrappers - DEBUG - [Dose_0+Dose_0][BACKTRACE] The_
↳previous error occurred at:
2      0x7f1bcd0100b2 amici::SteadystateProblem::handleSteadyStateFailure(bool, bool, _
↳bool) + 146
3      0x7f1bcd01084c amici::SteadystateProblem::findSteadyState(amici::Solver const&, _
↳amici::Model&, int) + 524
4      0x7f1bcd01089f amici::SteadystateProblem::workSteadyStateProblem(amici::Solver_
↳const&, amici::Model&, int) + 63
5      0x7f1bccfb9f1b amici::runAmiciSimulation(amici::Solver&, amici::ExpData const*, _
↳amici::Model&, bool) + 379
6      0x7f1bccfbb527 /home/docs/checkouts/readthe

```

-> Failed.

Relaxing tolerances by factor 1000

```

2023-11-21 19:20:19.585 - amici.swig_wrappers - DEBUG - [Dose_0+Dose_0][CVODES:CNode:ERR_
↳FAILURE] AMICI ERROR: in module CVODES in function CNode : At t = 736.713 and h = 0.
↳000145837, the error test failed repeatedly or with |h| = hmin.
2023-11-21 19:20:19.586 - amici.swig_wrappers - DEBUG - [Dose_0+Dose_0][OTHER] AMICI_
↳equilibration failed at t=736.713.
2023-11-21 19:20:19.587 - amici.swig_wrappers - ERROR - [Dose_0+Dose_0][OTHER] AMICI_
↳simulation failed: Steady state computation failed. Simulation to steady state failed.
2023-11-21 19:20:19.588 - amici.swig_wrappers - DEBUG - [Dose_0+Dose_0][BACKTRACE] The_
↳previous error occurred at:
2      0x7f1bcd0100b2 amici::SteadystateProblem::handleSteadyStateFailure(bool, bool, _
↳bool) + 146
3      0x7f1bcd01084c amici::SteadystateProblem::findSteadyState(amici::Solver const&, _
↳amici::Model&, int) + 524
4      0x7f1bcd01089f amici::SteadystateProblem::workSteadyStateProblem(amici::Solver_
↳const&, amici::Model&, int) + 63
5      0x7f1bccfb9f1b amici::runAmiciSimulation(amici::Solver&, amici::ExpData const*, _
↳amici::Model&, bool) + 379
6      0x7f1bccfbb527 /home/docs/checkouts/readthe

```

-> Failed.

Relaxing tolerances by factor 10000

-> Succeeded with relative steady state tolerance 0.01

```

status: ['AMICI_SUCCESS', 'AMICI_SUCCESS', 'AMICI_SUCCESS', 'AMICI_SUCCESS', 'AMICI_
↳SUCCESS', 'AMICI_SUCCESS', 'AMICI_SUCCESS', 'AMICI_SUCCESS']
preeq_status=[<SteadyStateStatus.not_run: 0>, <SteadyStateStatus.success: 1>,
↳<SteadyStateStatus.not_run: 0>]
rdata.preeq_numsteps=array([[ 0, 577,  0]])

```

That fixed the error, and took only a quarter of the number steps as the previous run, but at the cost of much lower accuracy.

```

[25]: # Let's try increasing the number of Newton steps
# (this is 0 by default, so the Newton solver wasn't used before,
# as can be seen from the 0 in `rdata.preeq_numsteps[0]`)
amici_solver = amici_model.getSolver()
amici_solver.setNewtonMaxSteps(10**4)

res = simulate_petab(
    petab_problem=petab_problem,
    amici_model=amici_model,
    problem_parameters=problem_parameters,
    scaled_parameters=True,
    solver=amici_solver,
)
print(
    "status:",
    [amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]],
)

rdata = res[RDATAS][0]
print(

```

(continues on next page)

(continued from previous page)

```

    f"preeq_status={list(map(amici.SteadyStateStatus, rdata.preeq_status.flatten()))}"
)
print(f"{rdata.preeq_numsteps=}")
# hard to reproduce on GHA
if os.getenv("GITHUB_ACTIONS") is None:
    assert [
        amici.simulation_status_to_str(rdata.status) for rdata in res[RDATAS]
    ] == [
        "AMICI_ERROR",
        "AMICI_NOT_RUN",
        "AMICI_NOT_RUN",
        "AMICI_NOT_RUN",
        "AMICI_NOT_RUN",
        "AMICI_NOT_RUN",
        "AMICI_NOT_RUN",
        "AMICI_NOT_RUN",
    ]

```

```

2023-11-21 19:20:19.987 - amici.swig_wrappers - DEBUG - [Dose_0+Dose_0][CVODES:CVode:ERR_
↳ FAILURE] AMICI ERROR: in module CVODES in function CVode : At t = 736.713 and h = 0.
↳ 000145837, the error test failed repeatedly or with |h| = hmin.
2023-11-21 19:20:19.987 - amici.swig_wrappers - DEBUG - [Dose_0+Dose_0][OTHER] AMICI_
↳ equilibration failed at t=736.713.
2023-11-21 19:20:19.989 - amici.swig_wrappers - ERROR - [Dose_0+Dose_0][OTHER] AMICI_
↳ simulation failed: Steady state computation failed. First run of Newton solver failed:
↳ RHS could not be factorized. Simulation to steady state failed. Second run of Newton_
↳ solver failed: RHS could not be factorized.
2023-11-21 19:20:19.989 - amici.swig_wrappers - DEBUG - [Dose_0+Dose_0][BACKTRACE] The_
↳ previous error occurred at:
2      0x7f1bcd0100b2 amici::SteadystateProblem::handleSteadyStateFailure(bool, bool,
↳ bool) + 146
3      0x7f1bcd01084c amici::SteadystateProblem::findSteadyState(amici::Solver const&,
↳ amici::Model&, int) + 524
4      0x7f1bcd01089f amici::SteadystateProblem::workSteadyStateProblem(amici::Solver_
↳ const&, amici::Model&, int) + 63
5      0x7f1bccfb9f1b amici::runAmiciSimulation(amici::Solver&, amici::ExpData const*,
↳ amici::Model&, bool) + 379
6      0x7f1bccfbb527 /home/docs/checkouts/readthe

```

```

status: ['AMICI_ERROR', 'AMICI_NOT_RUN', 'AMICI_NOT_RUN', 'AMICI_NOT_RUN', 'AMICI_NOT_RUN
↳ ', 'AMICI_NOT_RUN', 'AMICI_NOT_RUN', 'AMICI_NOT_RUN']
preeq_status=[<SteadyStateStatus.failed_factorization: -3>, <SteadyStateStatus.failed: -
↳ 1>, <SteadyStateStatus.failed_factorization: -3>]
rdata.preeq_numsteps=array([[ 0, 1105,  0]])

```

Increasing the maximum number of Newton steps doesn't seem to help here. The Jacobian was numerically singular and its factorization failed. This can be a result of conserved quantities in the model. Section *Steady state sensitivity computation failed due to unsuccessful factorization of RHS Jacobian* shows how to address that.

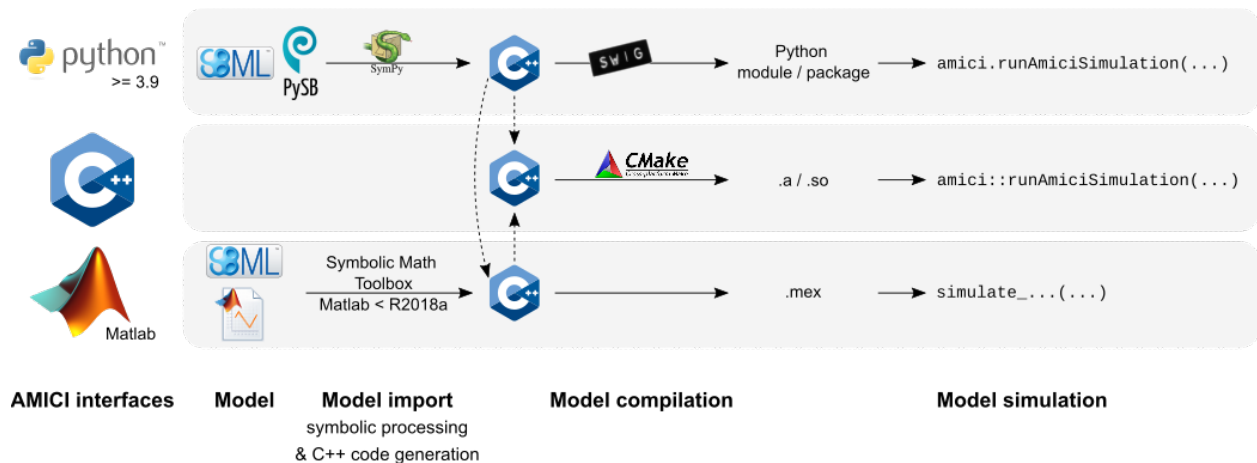
## 10.2.7 Speeding up model import and simulation - with a focus on large models

**Objective:** Give some hints to speed up import and simulation of larger models

This notebook gives some hints that may help to speed up import and simulation of (mostly) larger models. While some of these settings may also yield slight performance improvements for smaller models, other settings may make things slower. The impact may be highly model-dependent (number of states, number of parameters, rate expressions) or system-dependent and it's worthwhile doing some benchmarking.

To simulate models in AMICI, a model specified in a high-level format needs to be imported first, as shown in the following figure. This roughly involves the following steps:

1. Generating the ODEs
2. Computing derivatives
3. Generating C++ code
4. Compiling the generated code
5. Simulating the model



There are various options to speed up individual steps of this process. Generally, faster import comes with slower simulation and vice versa. During parameter estimation, a model is often imported only once, and then millions of simulations are run. Therefore, faster simulation will easily compensate for slower import (one-off cost). In other cases, many models may have to be imported, but only few simulations will be executed. In this case, faster import may be more relevant.

In the following, we will present various settings that (may) influence import and simulation time. We will follow the order of steps outlined above.

Since many of the following demonstrations take quite some time to compute, this notebook mostly shows pre-generated results.

```
[1]: from IPython.core.pylabtools import figsize, getfigs
import matplotlib.pyplot as plt
import pandas as pd

plt.rcParams.update({"font.size": 12})
```

## Examples

The demos below make use of the following models contained in the [PEtab benchmark collection](#) and other publications:

Model	# parameters	# states
Chen_MSB2009	155	500
Froehlich_CellSystems2018	4231	1396
FröhlichGer2022 (RTKERK__base)	105	2272
hello_pysb	4	3

All data has been generated with AMICI v0.15.0 or v0.16.0 unless stated otherwise.

## Model import

### Symbolic processing

#### Parameters as constants

By default, AMICI will generate sensitivity equations with respect to all model parameters. If it is clear up-front, that sensitivities with respect to certain parameters will not be required, their IDs can be passed to `amici.sbml_import.SbmlImporter.sbml2amici` or `amici.pysb_import.pysb2amici` via the `constant_parameters` argument to not generate the respective equations. This will reduce CPU time and RAM requirements during import and simulation. The PEPetab import will automatically pass all parameters with `petab.ESTIMATE==False` as `constant_parameters` arguments.

See also the following section for the case that no sensitivities are required at all.

#### Not generating sensitivity code

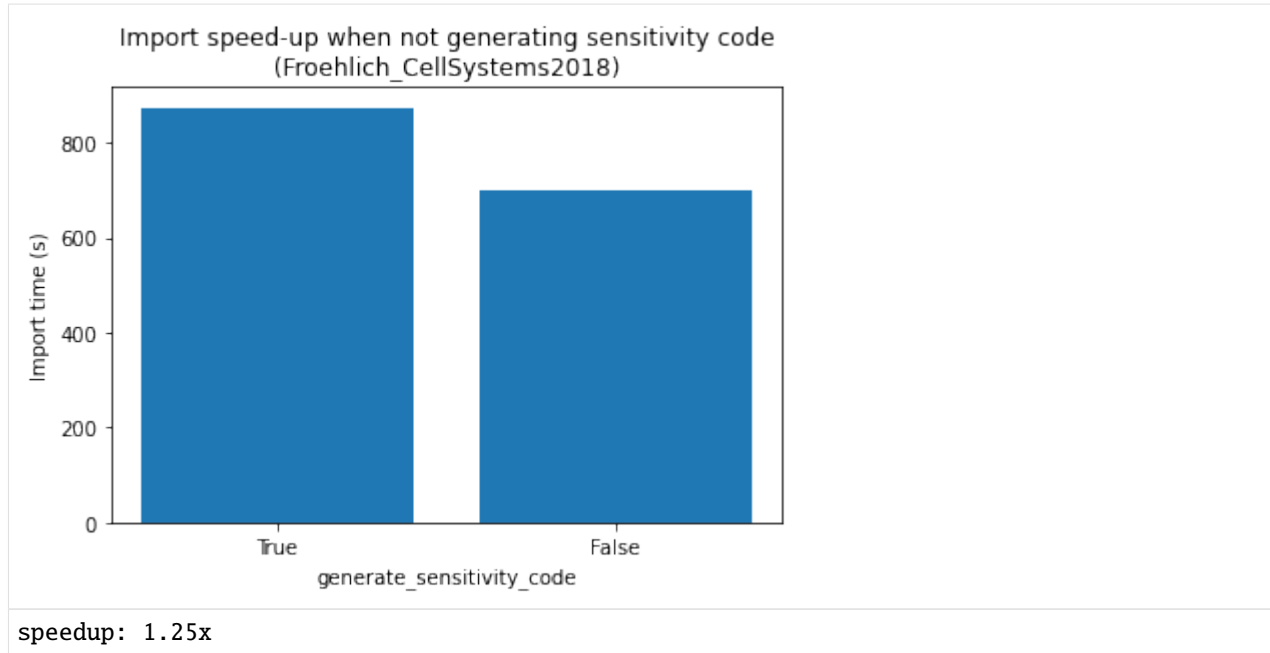
If only forward simulations of a model are required, a modest import speedup can be obtained from not generating sensitivity code. This can be enabled via the `generate_sensitivity_code` argument of `amici.sbml_import.SbmlImporter.sbml2amici` or `amici.pysb_import.pysb2amici`.

Example:

```
petab_yaml="https://raw.githubusercontent.com/Benchmarking-Initiative/Benchmark-Models-
↳PEtab/master/Benchmark-Models/Froehlich_CellSystems2018/Froehlich_CellSystems2018.yaml"
/usr/bin/time -v amici_import_petab -y "$petab_yaml" --no-compile
# vs.
/usr/bin/time -v amici_import_petab -y "$petab_yaml" --no-compile --no-sensitivities
```

```
[2]: figsize(4, 4)
plt.bar(["True", "False"], [873.54, 697.85])
plt.xlabel("generate_sensitivity_code")
plt.ylabel("Import time (s)")
plt.title(
    "Import speed-up when not generating sensitivity code\n(Froehlich_CellSystems2018)"
)
plt.show()

print(f"speedup: {873.54/697.85:.2f}x")
```



### Extracting common subexpressions

For some models, the size of the generated model code can be significantly reduced by extracting common subexpressions. This can yield substantial reductions of compile times and RAM-requirements. Very large models might not compile without this option. Extracting common subexpressions can be enabled by setting an environment variable `AMICI_EXTRACT_CSE=1` before model import. The downside is, that the generated model code becomes rather unreadable. The increase in import time when enabling this feature is usually <15%, the effect on code size and compile time is highly model dependent. Mostly models with tightly coupled ODEs, as obtained from complex rate laws or spatial discretizations of ODEs, seem to benefit. For models with mass action or similar kinetics, this option seems to not be helpful and rather increases compile time (e.g., for FröhlichGer2022, the compile time doubles).

Benchmark result from [here](#) (SBML import, `AMICI_IMPORT_NPROCS=2`, sequential compilation with clang14, `CFLAGS=-O2`):

	default	AMICI_EXTRACT_CSE=1
import time	160 min (100%)	164 min (103%)
code size	89 MB (100%)	27 MB (30%)
compile time	169 min (100%)	90 min (53%)
compile RAM	7.49 GB (100%)	1.18 GB (16%)
simulation time (*)	100%	97%

(\*) lowest out of 20 identical simulations using ASA

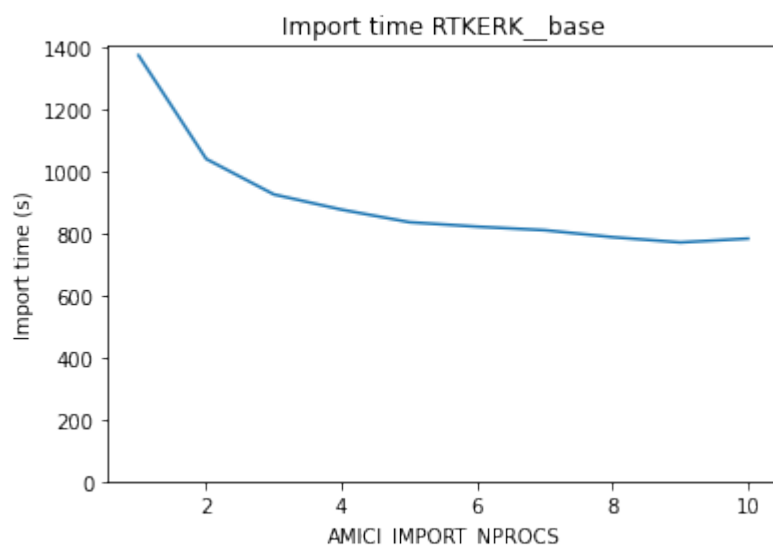
## Parallelization

For large models or complex model expressions, symbolic computation of the derivatives can be quite time consuming. This can be parallelized by setting the environment variable `AMICI_IMPORT_NPROCS` to the number of parallel processes that should be used. The impact strongly depends on the model. Note that setting this value too high may have a negative performance impact (benchmark!).

Impact for a large and a tiny model:

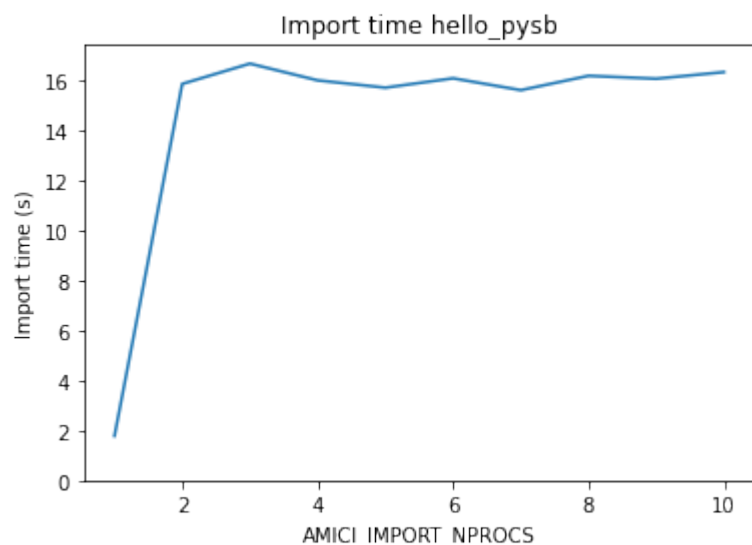
```
[3]: df_import = pd.read_csv("results_import.tsv", sep="\t")
for model_name, df in df_import.groupby("model_name"):
    plt.plot(df.nprocs, df.time)
    plt.title(f"Import time {model_name}")
    plt.xlabel("AMICI_IMPORT_NPROCS")
    plt.ylabel("Import time (s)")
    plt.ylim(ymin=0)
    plt.show()

import_times = df.sort_values("nprocs")["time"].values
percent_change = (
    (import_times[0] - min(import_times[1:])) / import_times[0] * 100
)
if percent_change > 0:
    print(f"Import time decreased by up to ~{percent_change:.0f}%.")
else:
    print(f"Import time increased by at least ~{-percent_change:.0f}%.")
```



Import time decreased by up to ~44%.





Import time increased by at least ~774%.

### (No) simplification of model expressions

By default, AMICI will try to perform some basic simplification of model expressions. For complex model expressions, or for large models, this can become costly. It very much depends on the model expressions, whether the benefits outweigh the cost.

Simplification of model expressions can be disabled by passing `simplify=None` to `amici.sbml_import.SbmlImporter.sbml2amici` or `amici.pysb_import.pysb2amici`.

Depending on the given model, different simplification schemes may be cheaper or more beneficial than the default. SymPy's simplification functions are [well documented](#).

## Compilation

### Choice of compiler

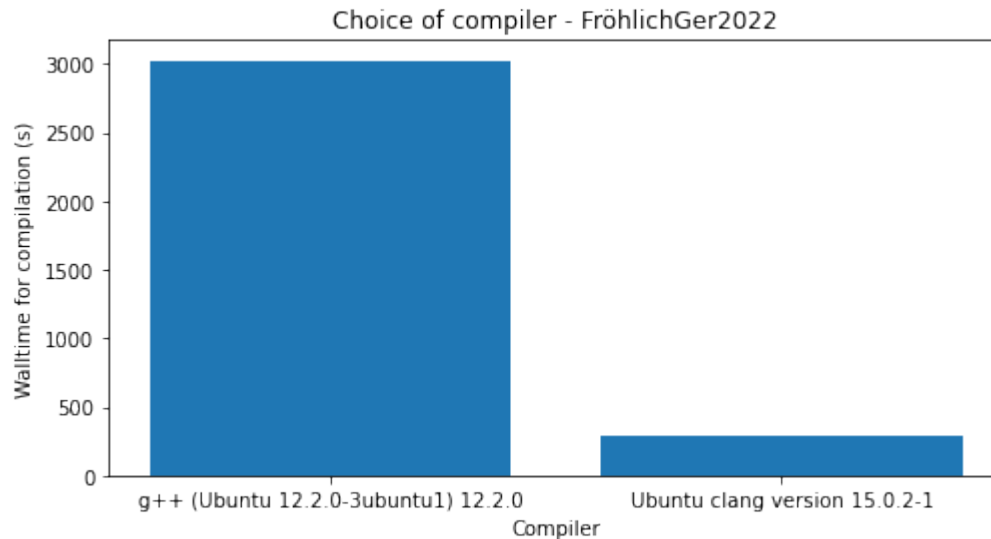
From own experience, clang seems to handle larger models, or more specifically, their large source files, better than g++, both in terms of memory requirement and compile time. You can use a different compiler by setting the CC and CXX environment variables to, e.g., `CC=clang`, `CXX=clang`.

```
[4]: figsize(8, 4)
      compilation_time_s = [3022.453, 289.518]
      labels = [
          "g++ (Ubuntu 12.2.0-3ubuntu1) 12.2.0",
          "Ubuntu clang version 15.0.2-1",
      ]
      plt.bar(labels, compilation_time_s)
      plt.ylim(ymin=0)
      plt.title("Choice of compiler - FröhlichGer2022")
      plt.xlabel("Compiler")
      plt.ylabel("Walltime for compilation (s)")
```

(continues on next page)

(continued from previous page)

```
plt.show()
print(
    f"Clang was ~{compilation_time_s[0] / compilation_time_s[1]:.0f}x as fast as g++."
)
```



Clang was ~10x as fast as g++.

## Parallel compilation

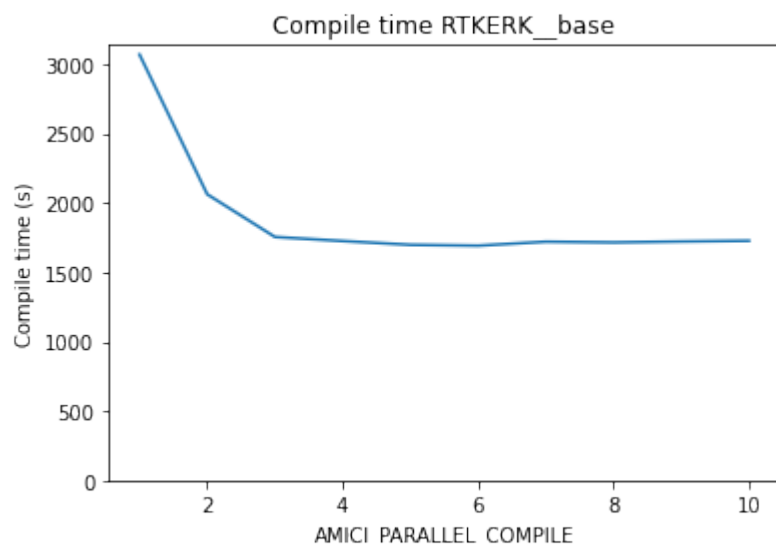
It's possible to compile multiple source files in parallel by specifying the number of parallel processes via the `AMICI_PARALLEL_COMPILE` environment variable. This is also beneficial for small models. Note, however, that for large models, this may require significant amounts of RAM.

Example for a large and tiny model:

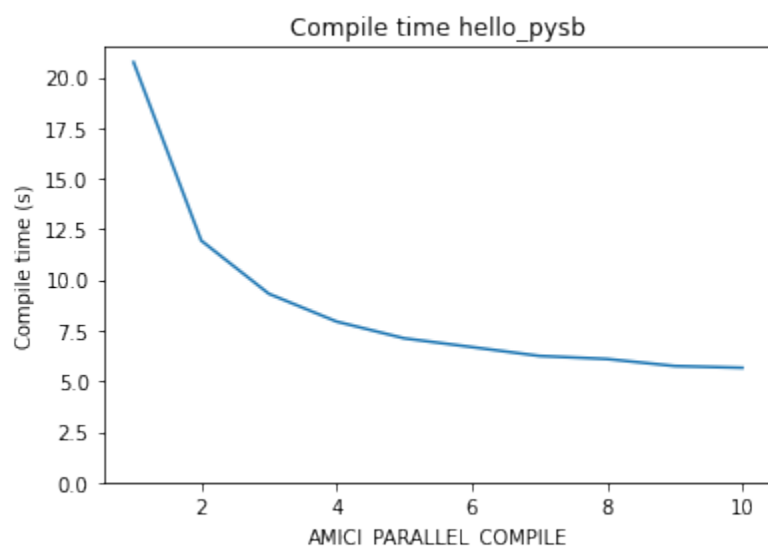
```
[5]: df_compile = pd.read_csv("results_compile.tsv", sep="\t")
figsize(6, 4)

for model_name, df in df_compile.groupby("model_name"):
    plt.plot(df.nprocs, df.time)
    plt.title(f"Compile time {model_name}")
    plt.xlabel("AMICI_PARALLEL_COMPILE")
    plt.ylabel("Compile time (s)")
    plt.ylim(ymin=0)
    plt.show()

    compilation_time_s = df.sort_values("nprocs")["time"].values
    print(
        "We were able to reduce compile time by up to "
        f"~{(compilation_time_s[0] - min(compilation_time_s[1:])) / compilation_time_
        ↪s[0] * 100:.0f}%."
    )
```



We were able to reduce compile time by up to ~45%.



We were able to reduce compile time by up to ~73%.

## Compiler flags

For most compilers, different machine code optimizations can be enabled/disabled by the `-O0`, `-O1`, `-O2`, `-O3` flags, where a higher number enables more optimizations. For fastest simulation, `-O3` should be used. However, these optimizations come at the cost of increased compile times. If models grow very large, some optimizations (especially with `g++`, see above) become prohibitively slow. In this case, a lower optimization level may be necessary to be able to compile models at all.

Another potential performance gain can be obtained from using CPU-specific instructions using `-march=native`. The disadvantage is, that the compiled model extension will only run on CPUs supporting the same instruction set. This may become problematic when attempting to use an AMICI model on a machine other than on which it was compiled (e.g. on heterogeneous compute clusters).

These compiler flags should be set for both, AMICI installation and model compilation.

For AMICI installation, e.g.,

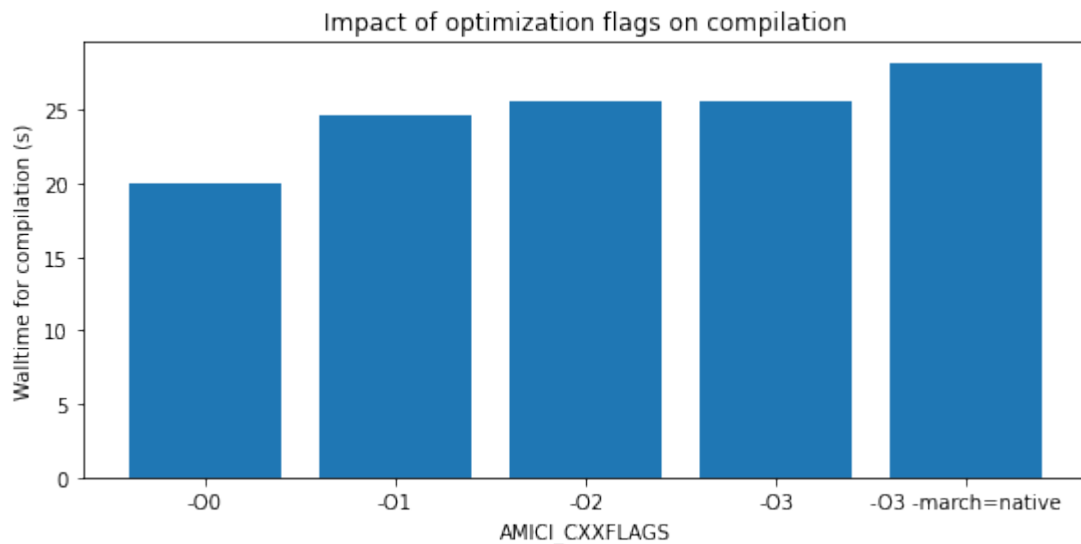
```
CFLAGS="-O3 -march=native" pip install amici
```

For model compilation, flags can be passed via the `AMICI_CXXFLAGS` environment variable.

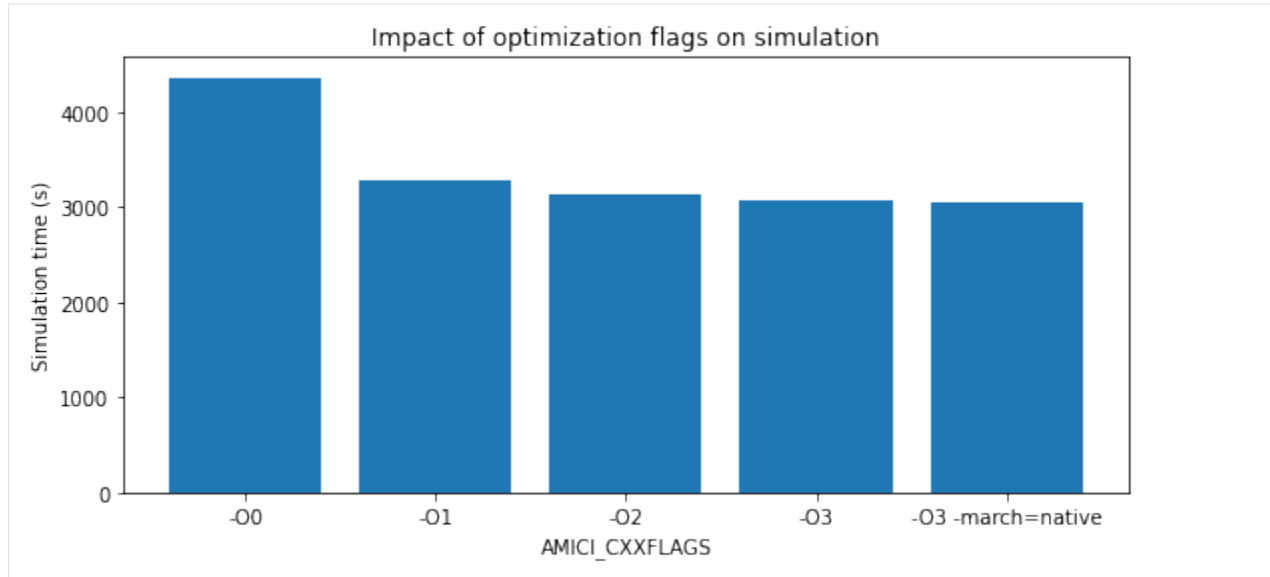
Example:

```
petab_yaml="https://raw.githubusercontent.com/Benchmarking-Initiative/Benchmark-Models-
↳ Petab/master/Benchmark-Models/Chen_MSB2009/Chen_MSB2009.yaml"
amici_import_petab -y "${petab_yaml}" --no-compile
cd Chen_MSB2009-amici0.16.0/
for cflags in "-O0" "-O1" "-O2" "-O3" "-O3 -march=native"
  # this line only builds the model extension, and is normally performed automatically.
↳ during import
  do AMICI_PARALLEL_COMPILE=1 AMICI_CXXFLAGS=${cflags} /usr/bin/time -v python setup.
↳ py build_ext --force --build-lib .
done
```

```
[6]: compilation_time_s = [20.01, 24.62, 25.59, 25.63, 28.21]
labels = ["-O0", "-O1", "-O2", "-O3", "-O3 -march=native"]
figsize(9, 4)
plt.bar(labels, compilation_time_s)
plt.title("Impact of optimization flags on compilation")
plt.xlabel("AMICI_CXXFLAGS")
plt.ylabel("Walltime for compilation (s)");
```



```
[7]: plt.bar(
    ["-O0", "-O1", "-O2", "-O3", "-O3 -march=native"],
    [4357.768, 3276.873, 3140.092, 3069.855, 3039.262],
)
plt.title("Impact of optimization flags on simulation")
plt.xlabel("AMICI_CXXFLAGS")
plt.ylabel("Simulation time (s)");
```



### Using some optimized BLAS

You might have access to some custom [BLAS](#) optimized for your hardware which might speed up your simulations somewhat. We are not aware of any systematic evaluation and cannot make any recommendation. You pass the respective compiler and linker flags via the environment variables `BLAS_CFLAGS` and `BLAS_LIBS`, respectively.

### Model simulation

A major determinant of simulation time for a given model is the required accuracy and the selected solvers. This has been evaluated, for example, in <https://doi.org/10.1038/s41598-021-82196-2> and is not covered further here.

### Adjoint vs. forward sensitivities

If only the objective function gradient is required, adjoint sensitivity analysis are often preferable over forward sensitivity analysis. As a rule of thumb, adjoint sensitivity analysis seems to outperform forward sensitivity analysis for models with more than 20 parameters:

CC BY 4.0 Fröhlich et al., DOI:10.1371/journal.pcbi.1005331

### Sensitivities w.r.t. a subset of parameters

If only sensitivities with respect to a subset of model parameters are of interest to you (see also *Parameters as constants* above), you can speed up the simulation by selecting the relevant parameter indices via `amici.Model.setParameterList`.

## Parallel simulation of multiple conditions

Whenever there are multiple independent simulations to perform, you can use `amici.runAmiciSimulations(..., num_threads=...)` instead of `amici.runAmiciSimulations(...)` to run them in parallel. Note that all simulation results have to be kept in memory, which may become problematic for very large numbers of simulations. Parallelization is based on OpenMP and does not come with the issues associated with Python's multiprocessing or multithreading (spawning extra processes or limitations related to the global interpreter lock).

## Reporting mode

During model simulation, many quantities are calculated, but not all might be of interest for you. For example, for parameter estimation you might only be interested in the likelihood and gradient. In this case, you can save time and memory using

```
amici.Solver.setReturnDataReportingMode(amici.RDataReporting.likelihood).
```

The purpose of this guide is to showcase how AMICI can be combined with differentiable programming in JAX. We will do so by reimplementing the parameter transformations available in AMICI in JAX and comparing it to the native implementation.

```
[1]: import jax
import jax.numpy as jnp
```

### 10.2.8 Preparation

To get started we will import a model using the `petab`. To this end, we will use the `benchmark` collection, which features a variety of different models. For more details about petab import, see the respective notebook `petab notebook`.

```
[2]: !git clone --depth 1 https://github.com/Benchmarking-Initiative/Benchmark-Models-PETab.
↪git tmp/benchmark-models || (cd tmp/benchmark-models && git pull)
from pathlib import Path
```

```
folder_base = Path(".") / "tmp" / "benchmark-models" / "Benchmark-Models"
```

```
Cloning into 'tmp/benchmark-models'...
remote: Enumerating objects: 336, done.
remote: Counting objects: 100% (336/336), done.
remote: Compressing objects: 100% (285/285), done.
remote: Total 336 (delta 88), reused 216 (delta 39), pack-reused 0
Receiving objects: 100% (336/336), 2.11 MiB | 7.48 MiB/s, done.
Resolving deltas: 100% (88/88), done.
```

From the benchmark collection, we now import the Boehm model.

```
[3]: import petab

model_name = "Boehm_JProteomeRes2014"
yaml_file = folder_base / model_name / (model_name + ".yaml")
petab_problem = petab.Problem.from_yaml(yaml_file)
```

The petab problem includes information about parameter scaling in it's the parameter table. For the boehm model, all estimated parameters (petab. ESTIMATE column equal to 1) have a petab.LOG10 as parameter scaling.

[4]: petab\_problem.parameter\_df

```
[4]:
```

	parameterName	parameterScale	lowerBound	\
parameterId				
Epo_degradation_BaF3	EPO_{degradation,BaF3}	log10	0.00001	
k_exp_hetero	k_{exp,hetero}	log10	0.00001	
k_exp_homo	k_{exp,homo}	log10	0.00001	
k_imp_hetero	k_{imp,hetero}	log10	0.00001	
k_imp_homo	k_{imp,homo}	log10	0.00001	
k_phos	k_{phos}	log10	0.00001	
ratio	ratio	lin	-5.00000	
sd_pSTAT5A_rel	\sigma_{pSTAT5A,rel}	log10	0.00001	
sd_pSTAT5B_rel	\sigma_{pSTAT5B,rel}	log10	0.00001	
sd_rSTAT5A_rel	\sigma_{rSTAT5A,rel}	log10	0.00001	
specC17	specC17	lin	-5.00000	

	upperBound	nominalValue	estimate
parameterId			
Epo_degradation_BaF3	100000	0.026983	1
k_exp_hetero	100000	0.000010	1
k_exp_homo	100000	0.006170	1
k_imp_hetero	100000	0.016368	1
k_imp_homo	100000	97749.379402	1
k_phos	100000	15766.507020	1
ratio	5	0.693000	0
sd_pSTAT5A_rel	100000	3.852612	1
sd_pSTAT5B_rel	100000	6.591478	1
sd_rSTAT5A_rel	100000	3.152713	1
specC17	5	0.107000	0

We now import the petab problem using ``amici.petab_import`` <[https://amici.readthedocs.io/en/latest/generated/amici.petab\\_import.html#amici.petab\\_import.import\\_petab\\_problem](https://amici.readthedocs.io/en/latest/generated/amici.petab_import.html#amici.petab_import.import_petab_problem)>`\_.

[5]: `from amici.petab_import import import_petab_problem`

```
amici_model = import_petab_problem(petab_problem, force_compile=True)
```

```
2023-02-16 12:37:18.049 - amici.petab_import - INFO - Importing model ...
2023-02-16 12:37:18.050 - amici.petab_import - INFO - Validating PETab problem ...
2023-02-16 12:37:18.343 - amici.petab_import - INFO - Model name is 'Boehm_
↳ JProteomeRes2014'.
Writing model code to '/Users/fabian/Documents/projects/AMICI/documentation/amici_models/
↳ Boehm_JProteomeRes2014'.
2023-02-16 12:37:18.344 - amici.petab_import - INFO - Species: 8
2023-02-16 12:37:18.344 - amici.petab_import - INFO - Global parameters: 9
2023-02-16 12:37:18.344 - amici.petab_import - INFO - Reactions: 9
2023-02-16 12:37:18.353 - amici.petab_import - INFO - Observables: 3
2023-02-16 12:37:18.353 - amici.petab_import - INFO - Sigmas: 3
2023-02-16 12:37:18.357 - amici.petab_import - DEBUG - Adding output parameters to model:
↳ ['noiseParameter1_pSTAT5A_rel', 'noiseParameter1_pSTAT5B_rel', 'noiseParameter1_
↳ rSTAT5A_rel']
2023-02-16 12:37:18.357 - amici.petab_import - DEBUG - Adding initial assignments for []
```

(continues on next page)

(continued from previous page)

```

2023-02-16 12:37:18.363 - amici.petab_import - DEBUG - Condition table: (1, 1)
2023-02-16 12:37:18.364 - amici.petab_import - DEBUG - Fixed parameters are ['ratio',
↳ 'specC17']
2023-02-16 12:37:18.364 - amici.petab_import - INFO - Overall fixed parameters: 2
2023-02-16 12:37:18.364 - amici.petab_import - INFO - Variable parameters: 10
2023-02-16 12:37:18.398 - amici.sbml_import - DEBUG - Finished gathering local SBML
↳ symbols ++ (1.08E-02s)
2023-02-16 12:37:18.403 - amici.sbml_import - DEBUG - Finished processing SBML
↳ parameters ++ (2.27E-03s)
2023-02-16 12:37:18.405 - amici.sbml_import - DEBUG - Finished processing SBML
↳ compartments ++ (1.18E-04s)
2023-02-16 12:37:18.411 - amici.sbml_import - DEBUG - Finished processing SBML species
↳ initials +++ (2.19E-03s)
2023-02-16 12:37:18.413 - amici.sbml_import - DEBUG - Finished processing SBML rate
↳ rules +++ (1.46E-05s)
2023-02-16 12:37:18.413 - amici.sbml_import - DEBUG - Finished processing SBML species
↳ ++ (5.96E-03s)
2023-02-16 12:37:18.417 - amici.sbml_import - DEBUG - Finished processing SBML reactions
↳ ++ (1.01E-03s)
2023-02-16 12:37:18.420 - amici.sbml_import - DEBUG - Finished processing SBML rules
↳ ++ (1.15E-03s)
2023-02-16 12:37:18.422 - amici.sbml_import - DEBUG - Finished processing SBML initial
↳ assignments++ (4.30E-05s)
2023-02-16 12:37:18.424 - amici.sbml_import - DEBUG - Finished processing SBML species
↳ references ++ (1.96E-04s)
2023-02-16 12:37:18.426 - amici.sbml_import - DEBUG - Finished processing SBML events
↳ ++ (3.40E-05s)
2023-02-16 12:37:18.426 - amici.sbml_import - DEBUG - Finished importing SBML
↳ + (4.08E-02s)
2023-02-16 12:37:18.457 - amici.sbml_import - DEBUG - Finished processing SBML
↳ observables + (2.87E-02s)
2023-02-16 12:37:18.460 - amici.sbml_import - DEBUG - Finished processing SBML event
↳ observables + (1.12E-06s)
2023-02-16 12:37:18.476 - amici.ode_export - DEBUG - Finished running smart_multiply
↳ ++ (1.06E-03s)
2023-02-16 12:37:18.499 - amici.ode_export - DEBUG - Finished importing SbmlImporter
↳ + (2.54E-02s)
2023-02-16 12:37:18.520 - amici.ode_export - DEBUG - Finished simplifying Jy
↳ ++++ (1.33E-02s)
2023-02-16 12:37:18.521 - amici.ode_export - DEBUG - Finished computing Jy
↳ +++ (1.53E-02s)
2023-02-16 12:37:18.549 - amici.ode_export - DEBUG - Finished simplifying y
↳ ++++ (2.28E-02s)
2023-02-16 12:37:18.549 - amici.ode_export - DEBUG - Finished computing y
↳ +++ (2.48E-02s)
2023-02-16 12:37:18.554 - amici.ode_export - DEBUG - Finished simplifying sigmay
↳ ++++ (7.19E-05s)
2023-02-16 12:37:18.554 - amici.ode_export - DEBUG - Finished computing sigmay
↳ +++ (2.17E-03s)
2023-02-16 12:37:18.570 - amici.ode_export - DEBUG - Finished writing Jy.cpp
↳ ++ (6.58E-02s)
2023-02-16 12:37:18.592 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳ ++++ (1.63E-02s)

```

(continues on next page)



(continued from previous page)

```

2023-02-16 12:37:18.601 - amici.ode_export - DEBUG - Finished simplifying dJydsigma
↳      +++++ (5.78E-03s)
2023-02-16 12:37:18.601 - amici.ode_export - DEBUG - Finished computing dJydsigma
↳      +++ (2.73E-02s)
2023-02-16 12:37:18.604 - amici.ode_export - DEBUG - Finished writing dJydsigma.cpp
↳      ++ (3.18E-02s)
2023-02-16 12:37:18.619 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳      +++++ (9.30E-03s)
2023-02-16 12:37:18.630 - amici.ode_export - DEBUG - Finished simplifying dJydy
↳      +++++ (7.63E-03s)
2023-02-16 12:37:18.630 - amici.ode_export - DEBUG - Finished computing dJydy
↳      +++ (2.19E-02s)
2023-02-16 12:37:18.635 - amici.ode_export - DEBUG - Finished writing dJydy.cpp
↳      ++ (2.79E-02s)
2023-02-16 12:37:18.640 - amici.ode_export - DEBUG - Finished simplifying Jz
↳      +++++ (3.06E-05s)
2023-02-16 12:37:18.641 - amici.ode_export - DEBUG - Finished computing Jz
↳      +++ (1.71E-03s)
2023-02-16 12:37:18.643 - amici.ode_export - DEBUG - Finished computing z
↳      +++ (6.17E-05s)
2023-02-16 12:37:18.647 - amici.ode_export - DEBUG - Finished simplifying sigmaz
↳      +++++ (2.92E-05s)
2023-02-16 12:37:18.647 - amici.ode_export - DEBUG - Finished computing sigmaz
↳      +++ (2.01E-03s)
2023-02-16 12:37:18.647 - amici.ode_export - DEBUG - Finished writing Jz.cpp
↳      ++ (9.91E-03s)
2023-02-16 12:37:18.653 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳      +++++ (2.68E-05s)
2023-02-16 12:37:18.655 - amici.ode_export - DEBUG - Finished simplifying dJzdsigma
↳      +++++ (3.75E-05s)
2023-02-16 12:37:18.656 - amici.ode_export - DEBUG - Finished computing dJzdsigma
↳      +++ (4.13E-03s)
2023-02-16 12:37:18.656 - amici.ode_export - DEBUG - Finished writing dJzdsigma.cpp
↳      ++ (6.25E-03s)
2023-02-16 12:37:18.661 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳      +++++ (2.62E-05s)
2023-02-16 12:37:18.663 - amici.ode_export - DEBUG - Finished simplifying dJzdz
↳      +++++ (3.29E-05s)
2023-02-16 12:37:18.664 - amici.ode_export - DEBUG - Finished computing dJzdz
↳      +++ (3.72E-03s)
2023-02-16 12:37:18.664 - amici.ode_export - DEBUG - Finished writing dJzdz.cpp
↳      ++ (5.33E-03s)
2023-02-16 12:37:18.669 - amici.ode_export - DEBUG - Finished simplifying Jrz
↳      +++++ (3.16E-05s)
2023-02-16 12:37:18.670 - amici.ode_export - DEBUG - Finished computing Jrz
↳      +++ (1.69E-03s)
2023-02-16 12:37:18.672 - amici.ode_export - DEBUG - Finished computing rz
↳      +++ (5.45E-05s)
2023-02-16 12:37:18.672 - amici.ode_export - DEBUG - Finished writing Jrz.cpp
↳      ++ (5.46E-03s)
2023-02-16 12:37:18.677 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳      +++++ (2.60E-05s)

```

(continues on next page)

(continued from previous page)

```

2023-02-16 12:37:18.679 - amici.ode_export - DEBUG - Finished simplifying dJrzdsigma
↳      +++++ (3.63E-05s)
2023-02-16 12:37:18.679 - amici.ode_export - DEBUG - Finished computing dJrzdsigma
↳      +++++ (3.57E-03s)
2023-02-16 12:37:18.679 - amici.ode_export - DEBUG - Finished writing dJrzdsigma.cpp
↳      ++ (5.15E-03s)
2023-02-16 12:37:18.684 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳      +++++ (2.39E-05s)
2023-02-16 12:37:18.686 - amici.ode_export - DEBUG - Finished simplifying dJrzdz
↳      +++++ (3.45E-05s)
2023-02-16 12:37:18.687 - amici.ode_export - DEBUG - Finished computing dJrzdz
↳      +++++ (3.68E-03s)
2023-02-16 12:37:18.687 - amici.ode_export - DEBUG - Finished writing dJrzdz.cpp
↳      ++ (5.31E-03s)
2023-02-16 12:37:18.692 - amici.ode_export - DEBUG - Finished simplifying root
↳      +++++ (3.28E-05s)
2023-02-16 12:37:18.693 - amici.ode_export - DEBUG - Finished computing root
↳      +++++ (1.74E-03s)
2023-02-16 12:37:18.693 - amici.ode_export - DEBUG - Finished writing root.cpp
↳      ++ (3.46E-03s)
2023-02-16 12:37:18.707 - amici.ode_export - DEBUG - Finished simplifying w
↳      +++++ (7.04E-03s)
2023-02-16 12:37:18.707 - amici.ode_export - DEBUG - Finished computing w
↳      +++++ (8.93E-03s)
2023-02-16 12:37:18.719 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳      +++++ (9.60E-03s)
2023-02-16 12:37:18.726 - amici.ode_export - DEBUG - Finished simplifying dwdp
↳      +++++ (4.59E-03s)
2023-02-16 12:37:18.726 - amici.ode_export - DEBUG - Finished computing dwdp
↳      +++++ (2.93E-02s)
2023-02-16 12:37:18.730 - amici.ode_export - DEBUG - Finished writing dwdp.cpp
↳      ++ (3.43E-02s)
2023-02-16 12:37:18.743 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳      +++++ (8.10E-03s)
2023-02-16 12:37:18.749 - amici.ode_export - DEBUG - Finished simplifying dwdx
↳      +++++ (3.24E-03s)
2023-02-16 12:37:18.749 - amici.ode_export - DEBUG - Finished computing dwdx
↳      +++++ (1.54E-02s)
2023-02-16 12:37:18.753 - amici.ode_export - DEBUG - Finished writing dwdx.cpp
↳      ++ (2.02E-02s)
2023-02-16 12:37:18.762 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳      +++++ (3.16E-03s)
2023-02-16 12:37:18.767 - amici.ode_export - DEBUG - Finished simplifying dwdw
↳      +++++ (2.23E-03s)
2023-02-16 12:37:18.767 - amici.ode_export - DEBUG - Finished computing dwdw
↳      +++++ (9.56E-03s)
2023-02-16 12:37:18.769 - amici.ode_export - DEBUG - Finished writing dwdw.cpp
↳      ++ (1.30E-02s)
2023-02-16 12:37:18.780 - amici.ode_export - DEBUG - Finished simplifying xdot
↳      +++++ (3.54E-03s)
2023-02-16 12:37:18.780 - amici.ode_export - DEBUG - Finished computing xdot
↳      +++++ (5.53E-03s)

```

(continues on next page)

(continued from previous page)

```

2023-02-16 12:37:18.790 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳      +++++ (6.83E-03s)
2023-02-16 12:37:18.792 - amici.ode_export - DEBUG - Finished simplifying dxdotdw
↳      +++++ (2.24E-04s)
2023-02-16 12:37:18.793 - amici.ode_export - DEBUG - Finished computing dxdotdw
↳      +++ (1.98E-02s)
2023-02-16 12:37:18.797 - amici.ode_export - DEBUG - Finished writing dxdotdw.cpp
↳      ++ (2.55E-02s)
2023-02-16 12:37:18.804 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳      +++++ (5.02E-04s)
2023-02-16 12:37:18.806 - amici.ode_export - DEBUG - Finished simplifying dxdotdx_
↳ explicit      +++++ (3.57E-05s)
2023-02-16 12:37:18.806 - amici.ode_export - DEBUG - Finished computing dxdotdx_explicit
↳      +++ (4.47E-03s)
2023-02-16 12:37:18.806 - amici.ode_export - DEBUG - Finished writing dxdotdx_explicit.
↳ cpp      ++ (6.56E-03s)
2023-02-16 12:37:18.812 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳      +++++ (5.50E-04s)
2023-02-16 12:37:18.814 - amici.ode_export - DEBUG - Finished simplifying dxdotdp_
↳ explicit      +++++ (3.08E-05s)
2023-02-16 12:37:18.815 - amici.ode_export - DEBUG - Finished computing dxdotdp_explicit
↳      +++ (4.37E-03s)
2023-02-16 12:37:18.815 - amici.ode_export - DEBUG - Finished writing dxdotdp_explicit.
↳ cpp      ++ (6.87E-03s)
2023-02-16 12:37:18.854 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳      +++++ (3.12E-02s)
2023-02-16 12:37:18.897 - amici.ode_export - DEBUG - Finished simplifying dydx
↳      +++++ (4.02E-02s)
2023-02-16 12:37:18.898 - amici.ode_export - DEBUG - Finished computing dydx
↳      +++++ (7.70E-02s)
2023-02-16 12:37:18.903 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳      +++++ (4.23E-04s)
2023-02-16 12:37:18.905 - amici.ode_export - DEBUG - Finished simplifying dydw
↳      +++++ (2.99E-05s)
2023-02-16 12:37:18.906 - amici.ode_export - DEBUG - Finished computing dydw
↳      +++++ (4.61E-03s)
2023-02-16 12:37:18.942 - amici.ode_export - DEBUG - Finished simplifying dydx
↳      +++++ (3.36E-02s)
2023-02-16 12:37:18.942 - amici.ode_export - DEBUG - Finished computing dydx
↳      +++ (1.23E-01s)
2023-02-16 12:37:18.958 - amici.ode_export - DEBUG - Finished writing dydx.cpp
↳      ++ (1.40E-01s)
2023-02-16 12:37:18.966 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳      +++++ (3.85E-04s)
2023-02-16 12:37:18.968 - amici.ode_export - DEBUG - Finished simplifying dydp
↳      +++++ (3.45E-05s)
2023-02-16 12:37:18.968 - amici.ode_export - DEBUG - Finished computing dydp
↳      +++++ (4.15E-03s)
2023-02-16 12:37:18.971 - amici.ode_export - DEBUG - Finished simplifying dydp
↳      +++++ (3.38E-05s)
2023-02-16 12:37:18.971 - amici.ode_export - DEBUG - Finished computing dydp
↳      +++ (8.39E-03s)

```

(continues on next page)

(continued from previous page)

```

2023-02-16 12:37:18.971 - amici.ode_export - DEBUG - Finished writing dydp.cpp
↳      ++ (1.04E-02s)
2023-02-16 12:37:18.975 - amici.ode_export - DEBUG - Finished computing dzdx
↳      +++ (5.68E-05s)
2023-02-16 12:37:18.975 - amici.ode_export - DEBUG - Finished writing dzdx.cpp
↳      ++ (1.63E-03s)
2023-02-16 12:37:18.979 - amici.ode_export - DEBUG - Finished computing dzdp
↳      +++ (4.66E-05s)
2023-02-16 12:37:18.979 - amici.ode_export - DEBUG - Finished writing dzdp.cpp
↳      ++ (1.61E-03s)
2023-02-16 12:37:18.982 - amici.ode_export - DEBUG - Finished computing drzdx
↳      +++ (4.78E-05s)
2023-02-16 12:37:18.983 - amici.ode_export - DEBUG - Finished writing drzdx.cpp
↳      ++ (1.67E-03s)
2023-02-16 12:37:18.986 - amici.ode_export - DEBUG - Finished computing drzdp
↳      +++ (5.38E-05s)
2023-02-16 12:37:18.986 - amici.ode_export - DEBUG - Finished writing drzdp.cpp
↳      ++ (1.69E-03s)
2023-02-16 12:37:18.992 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳      ++++ (1.11E-04s)
2023-02-16 12:37:18.994 - amici.ode_export - DEBUG - Finished simplifying dsigmaydy
↳      ++++ (3.44E-05s)
2023-02-16 12:37:18.994 - amici.ode_export - DEBUG - Finished computing dsigmaydy
↳      +++ (3.81E-03s)
2023-02-16 12:37:18.994 - amici.ode_export - DEBUG - Finished writing dsigmaydy.cpp
↳      ++ (5.42E-03s)
2023-02-16 12:37:19.000 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳      ++++ (4.32E-04s)
2023-02-16 12:37:19.002 - amici.ode_export - DEBUG - Finished simplifying dsigmaydp
↳      ++++ (7.44E-05s)
2023-02-16 12:37:19.002 - amici.ode_export - DEBUG - Finished computing dsigmaydp
↳      +++ (4.23E-03s)
2023-02-16 12:37:19.003 - amici.ode_export - DEBUG - Finished writing dsigmaydp.cpp
↳      ++ (6.96E-03s)
2023-02-16 12:37:19.006 - amici.ode_export - DEBUG - Finished writing sigmay.cpp
↳      ++ (5.00E-04s)
2023-02-16 12:37:19.011 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳      ++++ (2.56E-05s)
2023-02-16 12:37:19.013 - amici.ode_export - DEBUG - Finished simplifying dsigmazdp
↳      ++++ (3.78E-05s)
2023-02-16 12:37:19.013 - amici.ode_export - DEBUG - Finished computing dsigmazdp
↳      +++ (3.67E-03s)
2023-02-16 12:37:19.014 - amici.ode_export - DEBUG - Finished writing dsigmazdp.cpp
↳      ++ (5.26E-03s)
2023-02-16 12:37:19.016 - amici.ode_export - DEBUG - Finished writing sigmaz.cpp
↳      ++ (1.64E-05s)
2023-02-16 12:37:19.019 - amici.ode_export - DEBUG - Finished computing stau
↳      +++ (5.81E-05s)
2023-02-16 12:37:19.019 - amici.ode_export - DEBUG - Finished writing stau.cpp
↳      ++ (1.65E-03s)
2023-02-16 12:37:19.023 - amici.ode_export - DEBUG - Finished computing deltax
↳      +++ (5.10E-05s)

```

(continues on next page)

(continued from previous page)

```

2023-02-16 12:37:19.024 - amici.ode_export - DEBUG - Finished writing deltax.cpp      ↵
↪      ++ (1.91E-03s)
2023-02-16 12:37:19.027 - amici.ode_export - DEBUG - Finished computing deltasx      ↵
↪      +++ (5.45E-05s)
2023-02-16 12:37:19.028 - amici.ode_export - DEBUG - Finished writing deltasx.cpp      ↵
↪      ++ (1.86E-03s)
2023-02-16 12:37:19.032 - amici.ode_export - DEBUG - Finished writing w.cpp      ↵
↪      ++ (2.53E-03s)
2023-02-16 12:37:19.038 - amici.ode_export - DEBUG - Finished simplifying x0      ↵
↪      ++++ (8.98E-04s)
2023-02-16 12:37:19.038 - amici.ode_export - DEBUG - Finished computing x0      ↵
↪      +++ (2.79E-03s)
2023-02-16 12:37:19.040 - amici.ode_export - DEBUG - Finished writing x0.cpp      ↵
↪      ++ (5.34E-03s)
2023-02-16 12:37:19.046 - amici.ode_export - DEBUG - Finished simplifying x0_      ↵
↪fixedParameters      ++++ (3.45E-04s)
2023-02-16 12:37:19.046 - amici.ode_export - DEBUG - Finished computing x0_      ↵
↪fixedParameters      +++ (2.33E-03s)
2023-02-16 12:37:19.047 - amici.ode_export - DEBUG - Finished writing x0_fixedParameters.      ↵
↪cpp      ++ (4.79E-03s)
2023-02-16 12:37:19.053 - amici.ode_export - DEBUG - Finished running smart_jacobian      ↵
↪      ++++ (9.02E-04s)
2023-02-16 12:37:19.055 - amici.ode_export - DEBUG - Finished simplifying sx0      ↵
↪      ++++ (3.90E-05s)
2023-02-16 12:37:19.055 - amici.ode_export - DEBUG - Finished computing sx0      ↵
↪      +++ (4.79E-03s)
2023-02-16 12:37:19.056 - amici.ode_export - DEBUG - Finished writing sx0.cpp      ↵
↪      ++ (6.54E-03s)
2023-02-16 12:37:19.061 - amici.ode_export - DEBUG - Finished running smart_jacobian      ↵
↪      ++++ (8.95E-05s)
2023-02-16 12:37:19.063 - amici.ode_export - DEBUG - Finished running smart_jacobian      ↵
↪      ++++ (1.41E-04s)
2023-02-16 12:37:19.066 - amici.ode_export - DEBUG - Finished simplifying sx0_      ↵
↪fixedParameters      ++++ (2.92E-05s)
2023-02-16 12:37:19.066 - amici.ode_export - DEBUG - Finished computing sx0_      ↵
↪fixedParameters      +++ (6.44E-03s)
2023-02-16 12:37:19.067 - amici.ode_export - DEBUG - Finished writing sx0_      ↵
↪fixedParameters.cpp      ++ (8.77E-03s)
2023-02-16 12:37:19.075 - amici.ode_export - DEBUG - Finished writing xdot.cpp      ↵
↪      ++ (5.99E-03s)
2023-02-16 12:37:19.080 - amici.ode_export - DEBUG - Finished writing y.cpp      ↵
↪      ++ (2.52E-03s)
2023-02-16 12:37:19.085 - amici.ode_export - DEBUG - Finished simplifying x_rdata      ↵
↪      ++++ (7.20E-05s)
2023-02-16 12:37:19.086 - amici.ode_export - DEBUG - Finished computing x_rdata      ↵
↪      +++ (2.21E-03s)
2023-02-16 12:37:19.087 - amici.ode_export - DEBUG - Finished writing x_rdata.cpp      ↵
↪      ++ (4.64E-03s)
2023-02-16 12:37:19.092 - amici.ode_export - DEBUG - Finished simplifying total_cl      ↵
↪      ++++ (3.50E-05s)
2023-02-16 12:37:19.093 - amici.ode_export - DEBUG - Finished computing total_cl      ↵
↪      +++ (1.79E-03s)

```

(continues on next page)

(continued from previous page)

```

2023-02-16 12:37:19.093 - amici.ode_export - DEBUG - Finished writing total_cl.cpp
↳ ++ (3.50E-03s)
2023-02-16 12:37:19.098 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳ ++++ (2.68E-05s)
2023-02-16 12:37:19.100 - amici.ode_export - DEBUG - Finished simplifying dtotal_cldp
↳ ++++ (3.98E-05s)
2023-02-16 12:37:19.100 - amici.ode_export - DEBUG - Finished computing dtotal_cldp
↳ +++ (3.62E-03s)
2023-02-16 12:37:19.100 - amici.ode_export - DEBUG - Finished writing dtotal_cldp.cpp
↳ ++ (5.25E-03s)
2023-02-16 12:37:19.105 - amici.ode_export - DEBUG - Finished simplifying dtotal_cldx_
↳ rdata ++++ (2.96E-05s)
2023-02-16 12:37:19.106 - amici.ode_export - DEBUG - Finished computing dtotal_cldx_
↳ rdata +++ (1.73E-03s)
2023-02-16 12:37:19.106 - amici.ode_export - DEBUG - Finished writing dtotal_cldx_rdata.
↳ cpp ++ (3.50E-03s)
2023-02-16 12:37:19.111 - amici.ode_export - DEBUG - Finished simplifying x_solver
↳ ++++ (7.02E-05s)
2023-02-16 12:37:19.112 - amici.ode_export - DEBUG - Finished computing x_solver
↳ +++ (1.82E-03s)
2023-02-16 12:37:19.113 - amici.ode_export - DEBUG - Finished writing x_solver.cpp
↳ ++ (4.36E-03s)
2023-02-16 12:37:19.118 - amici.ode_export - DEBUG - Finished simplifying dx_rdatadx_
↳ solver ++++ (2.82E-04s)
2023-02-16 12:37:19.118 - amici.ode_export - DEBUG - Finished computing dx_rdatadx_
↳ solver +++ (1.94E-03s)
2023-02-16 12:37:19.119 - amici.ode_export - DEBUG - Finished writing dx_rdatadx_solver.
↳ cpp ++ (4.05E-03s)
2023-02-16 12:37:19.125 - amici.ode_export - DEBUG - Finished simplifying dx_rdatadp
↳ ++++ (3.14E-04s)
2023-02-16 12:37:19.125 - amici.ode_export - DEBUG - Finished computing dx_rdatadp
↳ +++ (2.11E-03s)
2023-02-16 12:37:19.126 - amici.ode_export - DEBUG - Finished writing dx_rdatadp.cpp
↳ ++ (4.04E-03s)
2023-02-16 12:37:19.131 - amici.ode_export - DEBUG - Finished running smart_jacobian
↳ ++++ (2.58E-05s)
2023-02-16 12:37:19.133 - amici.ode_export - DEBUG - Finished simplifying dx_rdatadtcl
↳ ++++ (3.95E-05s)
2023-02-16 12:37:19.133 - amici.ode_export - DEBUG - Finished computing dx_rdatadtcl
↳ +++ (3.71E-03s)
2023-02-16 12:37:19.133 - amici.ode_export - DEBUG - Finished writing dx_rdatadtcl.cpp
↳ ++ (5.36E-03s)
2023-02-16 12:37:19.136 - amici.ode_export - DEBUG - Finished writing z.cpp
↳ ++ (1.84E-05s)
2023-02-16 12:37:19.138 - amici.ode_export - DEBUG - Finished writing rz.cpp
↳ ++ (1.69E-05s)
2023-02-16 12:37:19.147 - amici.ode_export - DEBUG - Finished generating cpp code
↳ + (6.45E-01s)
2023-02-16 12:37:31.424 - amici.ode_export - DEBUG - Finished compiling cpp code
↳ + (1.23E+01s)

```

running AmiciInstall

(continues on next page)



(continued from previous page)

```

hdf5.h found in /opt/homebrew/Cellar/hdf5/1.12.2_2/include
libhdf5.a found in /opt/homebrew/Cellar/hdf5/1.12.2_2/lib
running build_ext
Changed extra_compile_args for unix to ['-std=c++14']
Building model extension in /Users/fabian/Documents/projects/AMICI/documentation/amici_
↳models/Boehm_JProteomeRes2014
building 'Boehm_JProteomeRes2014._Boehm_JProteomeRes2014' extension
Testing SWIG executable swig4.0... FAILED.
Testing SWIG executable swig3.0... FAILED.
Testing SWIG executable swig... SUCCEEDED.
swigging swig/Boehm_JProteomeRes2014.i to swig/Boehm_JProteomeRes2014_wrap.cpp
swig -python -c++ -modern -outdir Boehm_JProteomeRes2014 -I/Users/fabian/Documents/
↳projects/AMICI/python/sdist/amici/swig -I/Users/fabian/Documents/projects/AMICI/python/
↳sdist/amici/include -o swig/Boehm_JProteomeRes2014_wrap.cpp swig/Boehm_
↳JProteomeRes2014.i
Deprecated command line option: -modern. Ignored, this option is now always on.
creating build
creating build/temp.macosx-13-arm64-cpython-310
creating build/temp.macosx-13-arm64-cpython-310/swig
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG_
↳-g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
↳I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
↳JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
↳I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
↳fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
↳Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
↳include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
↳projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
↳framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014.cpp -o build/temp.
↳macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014.o -std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG_
↳-g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
↳I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
↳JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
↳I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
↳fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
↳Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
↳include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
↳projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
↳framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_Jy.cpp -o build/
↳temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_Jy.o -std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG_
↳-g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
↳I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
↳JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
↳I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
↳fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
↳Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
↳include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
↳projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
↳framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_dJydsigma.cpp -o_
↳build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dJydsigma.o -std=c++14

```

(continues on next page)

(continued from previous page)

```

clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG \
  -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
  -I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
  -include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
  -I/projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
  -framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_dJydy.cpp -o
  -o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dJydy.o -std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG \
  -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
  -I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
  -include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
  -I/projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
  -framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_dJydy_colptrs.cpp
  -o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dJydy_colptrs.o -
  -std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG \
  -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
  -I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
  -include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
  -I/projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
  -framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_dJydy_rowvals.cpp
  -o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dJydy_rowvals.o -
  -std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG \
  -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
  -I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
  -include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
  -I/projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
  -framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_dsigmaydp.cpp -o
  -o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dsigmaydp.o -std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG \
  -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
  -I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
  -include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
  -I/projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
  -framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_dwdp.cpp -o build/
  -o temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dwdp.o -std=c++14

```



(continued from previous page)

```
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG
  -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
  -I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
  JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
  fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
  Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
  include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
  projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
  framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_dwdp_colptrs.cpp -
  -o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dwdp_colptrs.o -
  -std=c++14
```

```
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG
  -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
  -I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
  JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
  fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
  Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
  include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
  projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
  framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_dwdp_rowvals.cpp -
  -o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dwdp_rowvals.o -
  -std=c++14
```

```
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG
  -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
  -I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
  JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
  fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
  Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
  include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
  projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
  framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_dwdw.cpp -o build/
  temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dwdw.o -std=c++14
```

```
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG
  -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
  -I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
  JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
  fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
  Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
  include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
  projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
  framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_dwdw_colptrs.cpp -
  -o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dwdw_colptrs.o -
  -std=c++14
```

```
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG
  -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
  -I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
  JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
  -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
  fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
  Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
  include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
  projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
  framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_dwdw_rowvals.cpp -
  -o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dwdw_rowvals.o -
  -std=c++14
```

(continued from previous page)

```

clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG \
-g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_dwdx.cpp -o build/
temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dwdx.o -std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG \
-g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_dwdx_colptrs.cpp -
o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dwdx_colptrs.o -
std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG \
-g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_dwdx_rowvals.cpp -
o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dwdx_rowvals.o -
std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG \
-g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_dxdotdw.cpp -o
build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dxdotdw.o -std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG \
-g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_dxdotdw_colptrs.
cpp -o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dxdotdw_colptrs.o
std=c++14

```

(continued from previous page)

```

clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG_
↳ -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
↳ I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
↳ JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
↳ I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
↳ fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
↳ Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
↳ include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
↳ projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
↳ framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_dxdotdw_rowvals.
↳ cpp -o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dxdotdw_rowvals.o_
↳ -std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG_
↳ -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
↳ I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
↳ JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
↳ I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
↳ fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
↳ Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
↳ include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
↳ projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
↳ framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_dydx.cpp -o build/
↳ temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dydx.o -std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG_
↳ -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
↳ I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
↳ JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
↳ I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
↳ fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
↳ Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
↳ include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
↳ projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
↳ framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_sigmay.cpp -o_
↳ build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_sigmay.o -std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG_
↳ -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
↳ I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
↳ JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
↳ I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
↳ fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
↳ Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
↳ include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
↳ projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
↳ framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_sx0_
↳ fixedParameters.cpp -o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_
↳ sx0_fixedParameters.o -std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG_
↳ -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
↳ I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
↳ JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
↳ I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
↳ fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
↳ Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
↳ include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
↳ projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
↳ framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_w.cpp -o build/
↳ temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_w.o -std=c++14

```

(continued from previous page)

```

clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG
↳ -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
↳ I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
↳ JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
↳ I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
↳ fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
↳ Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
↳ include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
↳ projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
↳ framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_x0.cpp -o build/
↳ temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_x0.o -std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG
↳ -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
↳ I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
↳ JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
↳ I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
↳ fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
↳ Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
↳ include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
↳ projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
↳ framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_x0_
↳ fixedParameters.cpp -o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_
↳ x0_fixedParameters.o -std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG
↳ -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
↳ I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
↳ JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
↳ I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
↳ fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
↳ Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
↳ include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
↳ projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
↳ framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_x_rdata.cpp -o
↳ build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_x_rdata.o -std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG
↳ -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
↳ I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
↳ JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
↳ I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
↳ fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
↳ Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
↳ include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
↳ projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
↳ framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_x_solver.cpp -o
↳ build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_x_solver.o -std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG
↳ -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
↳ I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
↳ JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
↳ I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
↳ fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
↳ Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
↳ include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
↳ projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
↳ framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_xdot.cpp -o build/
↳ temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_xdot.o -std=c++14

```



(continued from previous page)

```

clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG_
↳ -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
↳ I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
↳ JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
↳ I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
↳ fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
↳ Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
↳ include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
↳ projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
↳ framework/Versions/3.10/include/python3.10 -c Boehm_JProteomeRes2014_y.cpp -o build/
↳ temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_y.o -std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG_
↳ -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
↳ I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
↳ JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
↳ I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
↳ fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
↳ Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
↳ include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
↳ projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
↳ framework/Versions/3.10/include/python3.10 -c swig/Boehm_JProteomeRes2014_wrap.cpp -o_
↳ build/temp.macosx-13-arm64-cpython-310/swig/Boehm_JProteomeRes2014_wrap.o -std=c++14
clang -Wno-unused-result -Wsign-compare -Wunreachable-code -fno-common -dynamic -DNDEBUG_
↳ -g -fwrapv -O3 -Wall -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX13.sdk -
↳ I/Users/fabian/Documents/projects/AMICI/documentation/amici_models/Boehm_
↳ JProteomeRes2014 -I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/include -
↳ I/Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/gsl -I/Users/
↳ fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/sundials/include -I/
↳ Users/fabian/Documents/projects/AMICI/python/sdist/amici/ThirdParty/SuiteSparse/
↳ include -I/opt/homebrew/Cellar/hdf5/1.12.2_2/include -I/Users/fabian/Documents/
↳ projects/AMICI/build/venv/include -I/opt/homebrew/opt/python@3.10/Frameworks/Python.
↳ framework/Versions/3.10/include/python3.10 -c wrapfunctions.cpp -o build/temp.macosx-
↳ 13-arm64-cpython-310/wrapfunctions.o -std=c++14
clang++ -bundle -undefined dynamic_lookup -isysroot /Library/Developer/CommandLineTools/
↳ SDKs/MacOSX13.sdk build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014.o_
↳ build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_Jy.o build/temp.macosx-
↳ 13-arm64-cpython-310/Boehm_JProteomeRes2014_dJydsigma.o build/temp.macosx-13-arm64-
↳ cpython-310/Boehm_JProteomeRes2014_dJydy.o build/temp.macosx-13-arm64-cpython-310/
↳ Boehm_JProteomeRes2014_dJydy_colptrs.o build/temp.macosx-13-arm64-cpython-310/Boehm_
↳ JProteomeRes2014_dJydy_rowvals.o build/temp.macosx-13-arm64-cpython-310/Boehm_
↳ JProteomeRes2014_dsigmaydp.o build/temp.macosx-13-arm64-cpython-310/Boehm_
↳ JProteomeRes2014_dwdp.o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_
↳ dwdp_colptrs.o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dwdp_
↳ rowvals.o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dwdw.o build/
↳ temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dwdw_colptrs.o build/temp.
↳ macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_dwdw_rowvals.o build/temp.macosx-13-
↳ arm64-cpython-310/Boehm_JProteomeRes2014_dwdx.o build/temp.macosx-13-arm64-cpython-310/
↳ Boehm_JProteomeRes2014_dwdx_colptrs.o build/temp.macosx-13-arm64-cpython-310/Boehm_
↳ JProteomeRes2014_dwdx_rowvals.o build/temp.macosx-13-arm64-cpython-310/Boehm_
↳ JProteomeRes2014_dxdotdw.o build/temp.macosx-13-arm64-cpython-310/Boehm_
↳ JProteomeRes2014_dxdotdw_colptrs.o build/temp.macosx-13-arm64-cpython-310/Boehm_
↳ JProteomeRes2014_dxdotdw_rowvals.o build/temp.macosx-13-arm64-cpython-310/Boehm_
↳ JProteomeRes2014_dydx.o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_
↳ sigmay.o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_sx0_
↳ fixedParameters.o build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_w.o_
↳ build/temp.macosx-13-arm64-cpython-310/Boehm_JProteomeRes2014_x0.o build/temp.macosx-
↳ 13-arm64-cpython-310/Boehm_JProteomeRes2014_x0_fixedParameters.o build/temp.macosx-13-
↳ arm64-cpython-310/Boehm_JProteomeRes2014_x_rdata.o build/temp.macosx-13-arm64-cpython-
↳ 310/Boehm_JProteomeRes2014_x_solver.o build/temp.macosx-13-arm64-cpython-310/Boehm_

```

(continued from previous page)

```
ld: warning: -undefined dynamic_lookup may not work with chained fixups
```

```
2023-02-16 12:37:31.673 - amici.petab_import - INFO - Finished Importing PETab model
↳ (1.36E+01s)
2023-02-16 12:37:31.684 - amici.petab_import - INFO - Successfully loaded model Boehm_
↳ JProteomeRes2014 from /Users/fabian/Documents/projects/AMICI/documentation/amici_
↳ models/Boehm_JProteomeRes2014.
```

### 10.2.9 JAX implementation

For full jax support, we would have to implement a new `primitive`, which would require quite a bit of engineering, and in the end wouldn't add much benefit since AMICI can't run on GPUs. Instead will interface AMICI using the experimental jax module `host_callback` <[https://jax.readthedocs.io/en/latest/jax.experimental.host\\_callback.html](https://jax.readthedocs.io/en/latest/jax.experimental.host_callback.html)>`\_\_`.

To do so, we define a base function that only takes a single argument (the parameters) and runs simulation using petab via `simulate_petab` <[https://amici.readthedocs.io/en/latest/generated/amici.petab\\_objective.html#amici.petab\\_objective.simulate\\_petab](https://amici.readthedocs.io/en/latest/generated/amici.petab_objective.html#amici.petab_objective.simulate_petab)>`\_\_`. To enable gradient computation later on, we create a solver object and set the sensitivity order to first order and pass it to `simulate_petab`. Moreover, `simulate_petab` expects a dictionary of parameters, so we create a dictionary using the free parameter ids from the petab problem. As we want to implement parameter transformation in JAX, we disable parameter scaling in petab by passing `scaled_parameters=True`.

```
[6]: from amici.petab_objective import simulate_petab
import amici

amici_solver = amici_model.getSolver()
amici_solver.setSensitivityOrder(amici.SensitivityOrder.first)

def amici_hcb_base(parameters: jnp.array):
    return simulate_petab(
        petab_problem,
        amici_model,
        problem_parameters=dict(zip(petab_problem.x_free_ids, parameters)),
        scaled_parameters=True,
        solver=amici_solver,
    )
```

Now we can use this base function to create two functions separate functions that compute the log-likelihood (`llh`) and it's gradient (`sllh`) in two individual routines. Note that, as we are using the same base function here, the log-likelihood computation will also run with sensitivities which is not necessary and will add some overhead. This is only out of convenience and should be fixed in an application where efficiency is important.

```
[7]: def amici_hcb_llh(parameters: jnp.array):
    return amici_hcb_base(parameters)["llh"]

def amici_hcb_sllh(parameters: jnp.array):
    sllh = amici_hcb_base(parameters)["sllh"]
    return jnp.asarray(
        tuple(sllh[par_id] for par_id in petab_problem.x_free_ids)
    )
```

Now we can finally define the JAX function that runs amici simulation using the host callback. We add a `custom_jvp` decorator so that we can define a custom jacobian vector product function in the next step. More details about custom jacobian vector product functions can be found in the [JAX documentation](#)

```
[8]: import jax.experimental.host_callback as hcb
      from jax import custom_jvp

      import numpy as np

      @custom_jvp
      def jax_objective(parameters: jnp.array):
          return hcb.call(
              amici_hcb_llh,
              parameters,
              result_shape=jax.ShapeDtypeStruct((), np.float64),
          )
```

Now we define the function that implement the jacobian vector product. This effectively just returns the objective function value (computed using the previously defined `jax_objective`) as well as the inner product of the gradient (computed using a host callback to the previously defined `amici_hcb_sllh`) and the tangents vector. Note that this implementation performs two simulation runs, one for the function value and one for the gradient, which is inefficient and could be avoided by caching solutions.

```
[9]: @jax_objective.defjvp
      def jax_objective_jvp(primals: jnp.array, tangents: jnp.array):
          (parameters,) = primals
          (x_dot,) = tangents
          llh = jax_objective(parameters)
          sllh = hcb.call(
              amici_hcb_sllh,
              parameters,
              result_shape=jax.ShapeDtypeStruct(
                  (petab_problem.parameter_df.estimate.sum(),), np.float64
              ),
          )
          return llh, sllh.dot(x_dot)
```

As last step, we implement the parameter transformation in jax. This effectively just extracts parameter scales from the petab problem, implements rescaling in jax and then passes the scaled parameters to the previously objective function we previously defined. We add the `value_and_grad` decorator such that the generated jax function returns both function value and function gradient in a tuple. Moreover, we add the `jax.jit` decorator such that the function is [just in time compiled](#) upon the first function call.

```
[10]: from jax import value_and_grad

      parameter_scales = petab_problem.parameter_df.loc[
          petab_problem.x_free_ids, petab.PARAMETER_SCALE
      ].values

      @jax.jit
      @value_and_grad
      def jax_objective_with_parameter_transform(parameters: jnp.array):
```

(continues on next page)

(continued from previous page)

```

par_scaled = jnp.asarray(
    tuple(
        value
        if scale == petab.LIN
        else jnp.log(value)
        if scale == petab.LOG
        else jnp.log10(value)
        for value, scale in zip(parameters, parameter_scales)
    )
)
return jax_objective(par_scaled)

```

### 10.2.10 Testing

We can now run the function to compute the log-likelihood and the gradient.

```

[11]: llh_jax, sllh_jax = jax_objective_with_parameter_transform(
    petab_problem.x_nominal_free
)

```

As a sanity check, we compare the computed value to native parameter transformation in amici.

```

[12]: r = simulate_petab(petab_problem, amici_model, solver=amici_solver)
# TODO remove me as soon as sllh in simulate_petab is fixed
sllh = {
    name: value / (np.log(10) * par_value)
    for (name, value), par_value in zip(
        r["sllh"].items(), petab_problem.x_nominal_free
    )
}

```

```

[13]: import pandas as pd

pd.Series(dict(amici=r["llh"], jax=float(llh_jax)))

```

```

[13]: amici    -138.221997
     jax      -138.222000
     dtype: float64

```

```

[14]: pd.DataFrame(
    index=sllh.keys(), data=dict(amici=sllh.values(), jax=np.asarray(sllh_jax))
)

```

```

[14]:
           amici           jax
Epo_degradation_BaF3 -3.546026e-01 -3.640394e-01
k_exp_hetero         -2.401005e+03 -2.401010e+03
k_exp_homo           -4.073832e-01 -4.106763e-01
k_imp_hetero         -1.432855e-01 -1.639030e-01
k_imp_homo           2.006412e-10  2.006412e-10
k_phos               -2.179950e-07 -2.089803e-07
sd_pSTAT5A_rel       -1.215545e-03 -1.222887e-03

```

(continues on next page)



(continued from previous page)

```
sd_pSTAT5B_rel    -1.583889e-03 -1.580870e-03
sd_rSTAT5A_rel    -2.643776e-03 -2.641361e-03
```

We see quite some differences in the gradient calculation. The primary reason is that running JAX in default configuration will use float32 precision for the parameters that are passed to AMICI, which uses float64, and the derivative of the parameter transformation. As AMICI simulations that run on the CPU are the most expensive operation, there is barely any tradeoff for using float32 vs float64 in JAX. Therefore we configure JAX to use float64 instead and rerun simulations.

```
[15]: jax.config.update("jax_enable_x64", True)
      llh_jax, sllh_jax = jax_objective_with_parameter_transform(
          petab_problem.x_nominal_free
      )
```

We can now evaluate the results again and see that differences between pure AMICI and AMICI/JAX implementations are now much smaller.

```
[16]: pd.Series(dict(amici=r["llh"], jax=float(llh_jax)))
```

```
[16]: amici    -138.221997
      jax      -138.221997
      dtype: float64
```

```
[17]: pd.DataFrame(
      index=sllh.keys(), data=dict(amici=sllh.values(), jax=np.asarray(sllh_jax))
      )
```

```
[17]:
```

	amici	jax
Epo_degradation_BaF3	-3.546026e-01	-3.546504e-01
k_exp_hetero	-2.401005e+03	-2.401005e+03
k_exp_homo	-4.073832e-01	-4.074248e-01
k_imp_hetero	-1.432855e-01	-1.433139e-01
k_imp_homo	2.006412e-10	2.006412e-10
k_phos	-2.179950e-07	-2.179076e-07
sd_pSTAT5A_rel	-1.215545e-03	-1.215596e-03
sd_pSTAT5B_rel	-1.583889e-03	-1.583805e-03
sd_rSTAT5A_rel	-2.643776e-03	-2.643703e-03

### 10.2.11 AMICI Python example “splines”

This is an example showing how to add spline assignment rules to a pre-existing SBML model.

#### Utility functions

```
[1]: import sys
import os
import libsbml
import amici

import numpy as np
import sympy as sp

from shutil import rmtree
from importlib import import_module
from uuid import uuid1
from tempfile import TemporaryDirectory
import matplotlib as mpl
from matplotlib import pyplot as plt

# Choose build directory
BUILD_PATH = None # temporary folder
# BUILD_PATH = 'build' # specified folder for debugging
if BUILD_PATH is not None:
    # Remove previous models
    rmtree(BUILD_PATH, ignore_errors=True)
    os.mkdir(BUILD_PATH)

def simulate(sbml_model, parameters=None, *, model_name=None, **kwargs):
    if model_name is None:
        model_name = "model_" + uuid1().hex
    if BUILD_PATH is None:
        with TemporaryDirectory() as build_dir:
            return _simulate(
                sbml_model,
                parameters,
                build_dir=build_dir,
                model_name=model_name,
                **kwargs
            )
    else:
        build_dir = os.path.join(BUILD_PATH, model_name)
        rmtree(build_dir, ignore_errors=True)
        return _simulate(
            sbml_model,
            parameters,
            build_dir=build_dir,
            model_name=model_name,
            **kwargs
        )
```

(continues on next page)

(continued from previous page)

```

def _simulate(
    sbml_model,
    parameters,
    *,
    build_dir,
    model_name,
    T=1,
    discard_annotations=False,
    plot=True
):
    if parameters is None:
        parameters = {}
    # Build the model module from the SBML file
    sbml_importer = amici.SbmlImporter(
        sbml_model, discard_annotations=discard_annotations
    )
    sbml_importer.sbml2amici(model_name, build_dir)
    # Import the model module
    sys.path.insert(0, os.path.abspath(build_dir))
    model_module = import_module(model_name)
    # Setup simulation timepoints and parameters
    model = model_module.getModel()
    for name, value in parameters.items():
        model.setParameterByName(name, value)
    if isinstance(T, (int, float)):
        T = np.linspace(0, T, 100)
    model.setTimepoints([float(t) for t in T])
    solver = model.getSolver()
    solver.setSensitivityOrder(amici.SensitivityOrder.first)
    solver.setSensitivityMethod(amici.SensitivityMethod.forward)
    # Simulate
    rdata = amici.runAmiciSimulation(model, solver)
    # Plot results
    if plot:
        fig, ax = plt.subplots()
        ax.plot(rdata["t"], rdata["x"])
        ax.set_xlabel("time")
        ax.set_ylabel("concentration")
    return model, rdata

```

## A simple SBML model

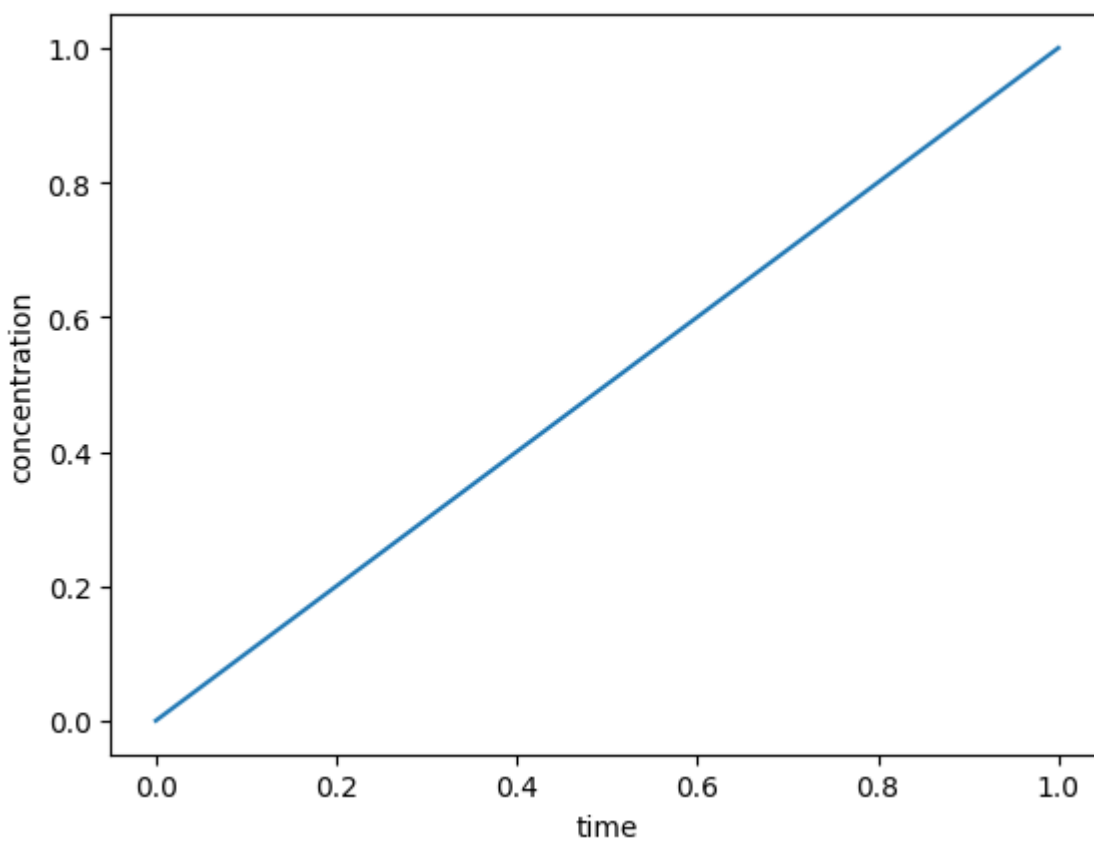
Let us consider the following SBML model:

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2/version5" level="2" version="5">
  <model id="example_splines">
    <listOfCompartments>
      <compartment id="compartment" size="1"/>
    </listOfCompartments>
    <listOfSpecies>
      <species id="x" compartment="compartment" initialAmount="0"/>
    </listOfSpecies>
    <listOfParameters>
      <parameter id="f" constant="false"/>
    </listOfParameters>
    <listOfRules>
      <rateRule variable="x">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
          <ci> f </ci>
        </math>
      </rateRule>
    </listOfRules>
  </model>
</sbml>
```

This model corresponds to the simple ODE  $\dot{x} = f$  for a species  $x$  and a parameter  $f$ .

We can easily import and simulate this model in AMICI.

```
[2]: simulate("example_splines.xml", dict(f=1));
```



### Adding a simple spline

Instead of using a constant parameter  $f$ , we want to use a smooth time-dependent function  $f(t)$  whose value is known only at a finite number of time instants. The value of  $f(t)$  outside such grid points needs to be smoothly interpolated. Several methods have been developed for this problem over the years; AMICI at the moment supports only [cubic Hermite splines](#).

We can add a spline function to an existing SBML model with the following code. The resulting time-dependent parameter  $f(t)$  will assume values  $(1, -0.5, 2)$  at the equally spaced points  $(0, 0.5, 1)$  and smoothly vary elsewhere.

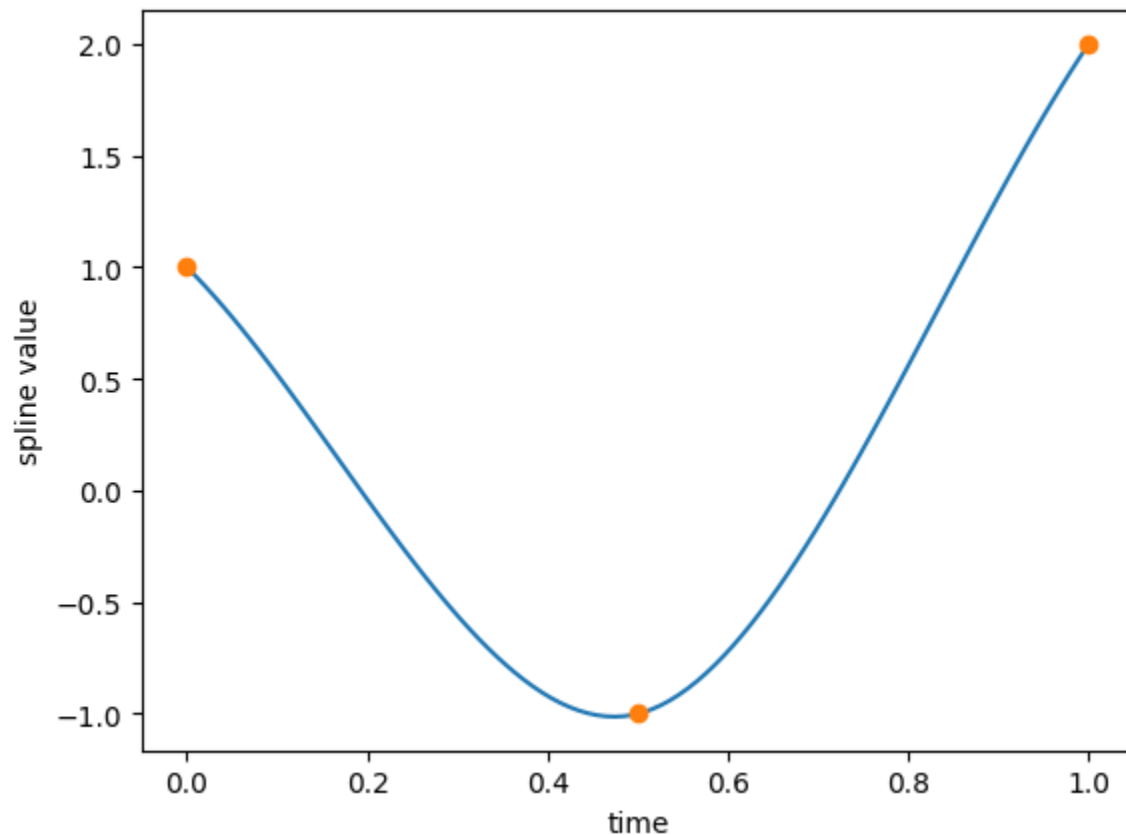
AMICI encodes the spline as a SBML assignment rule for the parameter  $f$ . Such a rule consists of a piecewise-polynomial formula which can be interpreted in any SBML-compliant software. However, such very complex formulas are computationally inefficient; e.g., in AMICI they lead to very long model creation times. To solve such problem the code below adds AMICI-specific SBML annotations to the assignment rule which can be used by AMICI to recreate the correct interpolant without reading the inefficient piecewise formula.

```
[3]: # Create a spline object
spline = amici.splines.CubicHermiteSpline(
    sbml_id="f",
    evaluate_at=amici.sbml_utils.amici_time_symbol, # the spline function is evaluated
    ↪ at the current time point
    nodes=amici.splines.UniformGrid(0, 1, number_of_nodes=3),
    values_at_nodes=[1, -1, 2],
)
```

```
[4]: # This spline object can be evaluated at any point
# and so can its derivative/integral
print(f"spline value at 0.3 = {spline.evaluate(0.3)}")
print(f"spline derivative at 0.3 = {spline.derivative(0.3)}")
print(f"spline integral between 0 and 1 = {spline.integrate(0.0, 1.0)}")
```

```
spline value at 0.3 = -0.5600000000000000
spline derivative at 0.3 = -4.6000000000000000
spline integral between 0 and 1 = 0.0416666666666672
```

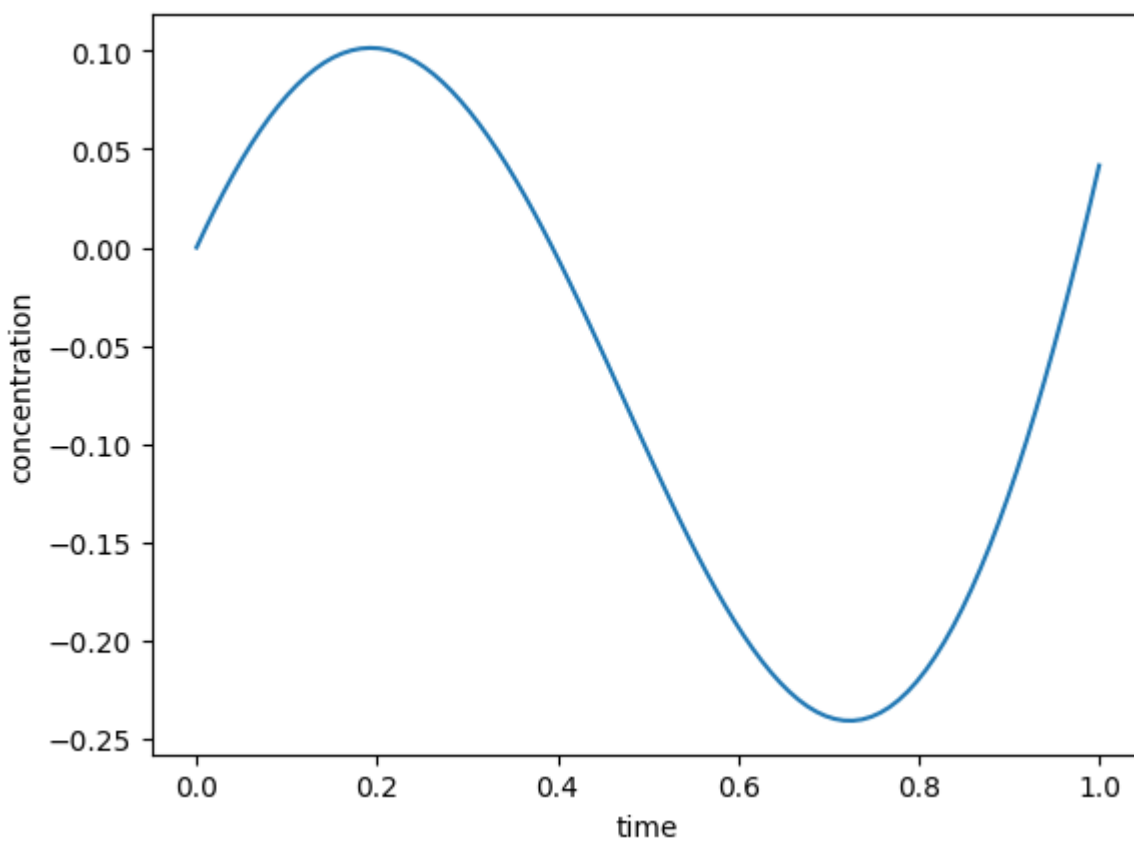
```
[5]: # Plot the spline
spline.plot(xlabel="time");
```



```
[6]: # Load SBML model using libsbml
import libsbml

sbml_doc = libsbml.SBMLReader().readSBML("example_splines.xml")
sbml_model = sbml_doc.getModel()
# We can add the spline assignment rule to the SBML model
spline.add_to_sbml_model(sbml_model)
```

```
[7]: # Finally, we can simulate it in AMICI
model, rdata = simulate(sbml_model);
```



```
[8]: # Final value should be equal to the integral computed above
assert np.allclose(rdata["x"][-1], float(spline.integrate(0.0, 1.0)))
```

The following is the SBML code for the above model

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2/version5" level="2" version="5">
  <model id="example_splines">
    <listOfCompartments>
      <compartment id="compartment" size="1"/>
    </listOfCompartments>
    <listOfSpecies>
      <species id="x" compartment="compartment" initialAmount="0"/>
    </listOfSpecies>
    <listOfParameters>
      <parameter id="f" constant="false"/>
    </listOfParameters>
    <listOfRules>
      <rateRule variable="x">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
          <ci> f </ci>
        </math>
      </rateRule>
      <assignmentRule variable="f">
```

(continues on next page)

(continued from previous page)

```

<annotation>
  <amici:spline xmlns:amici="https://github.com/AMICI-dev/AMICI" amici:spline_
↪method="cubic_hermite">
    <amici:spline_evaluation_point>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <csymbol encoding="text" definitionURL="http://www.sbml.org/sbml/symbols/
↪time"> time </csymbol>
      </math>
    </amici:spline_evaluation_point>
    <amici:spline_uniform_grid>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <cn>0</cn>
      </math>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <cn>1</cn>
      </math>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply>
          <divide/>
          <cn>1</cn>
          <cn>2</cn>
        </apply>
      </math>
    </amici:spline_uniform_grid>
    <amici:spline_values>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <cn>1</cn>
      </math>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <cn>-1</cn>
      </math>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <cn>2</cn>
      </math>
    </amici:spline_values>
  </amici:spline>
</annotation>
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <piecewise>
    ... piecewise representation of the spline ...
  </piecewise>
</math>
</assignmentRule>
</listOfRules>
</model>
</sbml>

```

The spline annotation on its own can be accessed by

```
[9]: print(spline.amici_annotation)
```

```

<amici:spline xmlns:amici="https://github.com/AMICI-dev/AMICI" amici:spline_method=
↪"cubic_hermite">

```

(continues on next page)



(continued from previous page)

```

    <amici:spline_evaluation_point>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <csymbol encoding="text" definitionURL="http://www.sbml.org/sbml/
→symbols/time"> time </csymbol>
      </math>
    </amici:spline_evaluation_point>
    <amici:spline_uniform_grid>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <cn>0</cn>
      </math>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <cn>1</cn>
      </math>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply>
          <divide/>
          <cn>1</cn>
          <cn>2</cn>
        </apply>
      </math>
    </amici:spline_uniform_grid>
    <amici:spline_values>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <cn>1</cn>
      </math>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <cn>-1</cn>
      </math>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <cn>2</cn>
      </math>
    </amici:spline_values>
  </amici:spline>

```

### Splines can be parametrized

Instead of constant values, SBML parameters can be used as spline values. These can also be automatically added to the model when adding the assignment rule.

```

[10]: spline = amici.splines.CubicHermiteSpline(
    sbml_id="f",
    evaluate_at=amici.sbml_utils.amici_time_symbol,
    nodes=amici.splines.UniformGrid(0, 1, number_of_nodes=3),
    values_at_nodes=sp.symbols("f0:3"),
)

[11]: sbml_doc = libsbml.SBMLReader().readSBML("example_splines.xml")
    sbml_model = sbml_doc.getModel()
    spline.add_to_sbml_model(
        sbml_model,

```

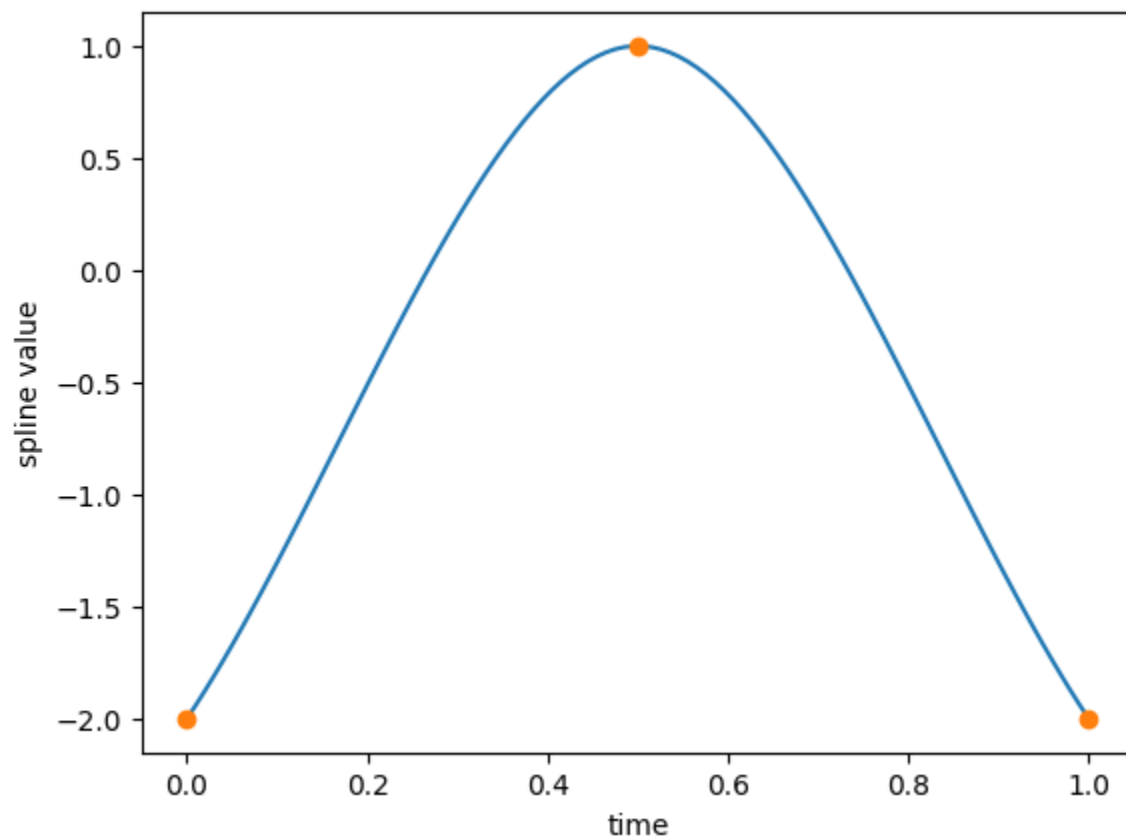
(continues on next page)

(continued from previous page)

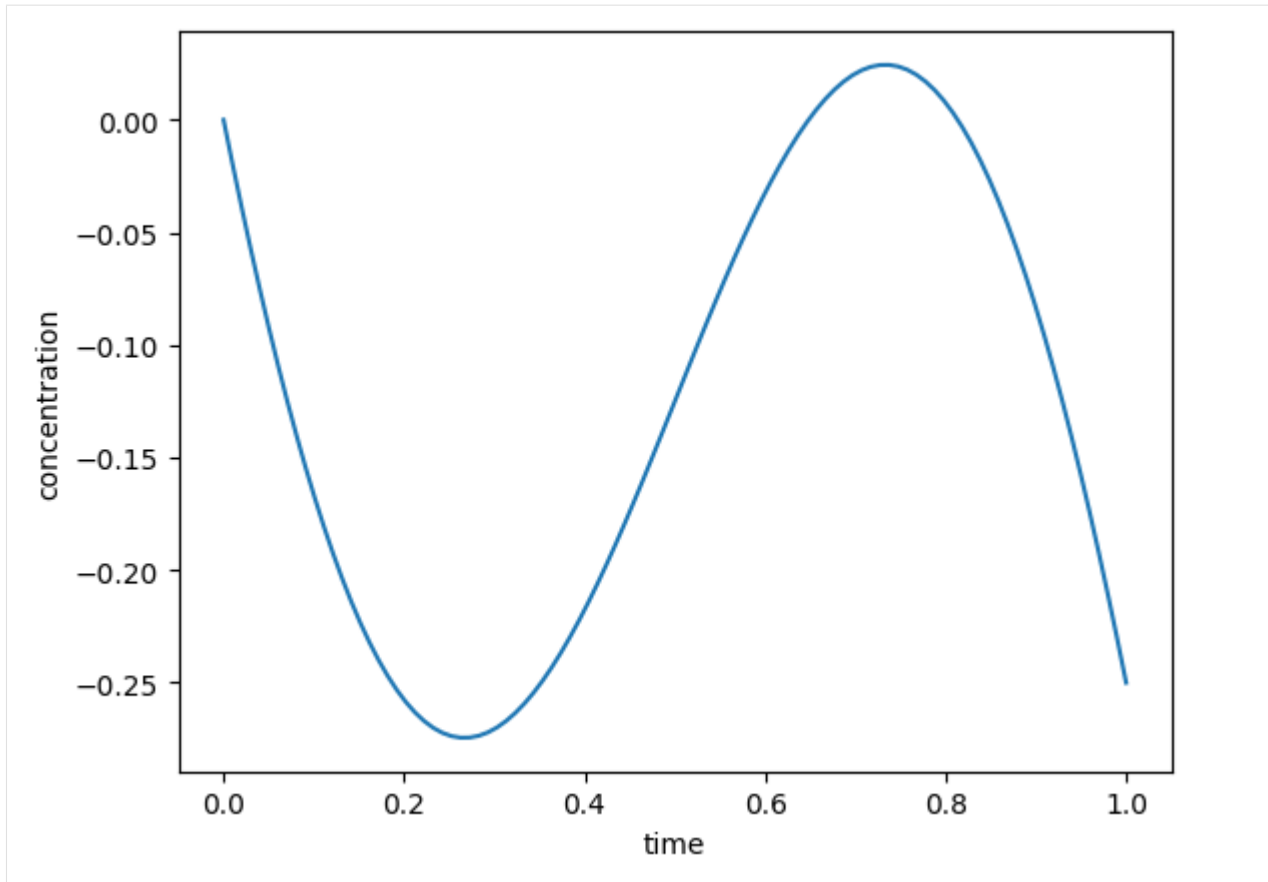
```
    auto_add=True,  
    y_nominal=[1, -0.5, 2],  
)
```

```
[12]: parameters = dict(f0=-2, f1=1, f2=-2)
```

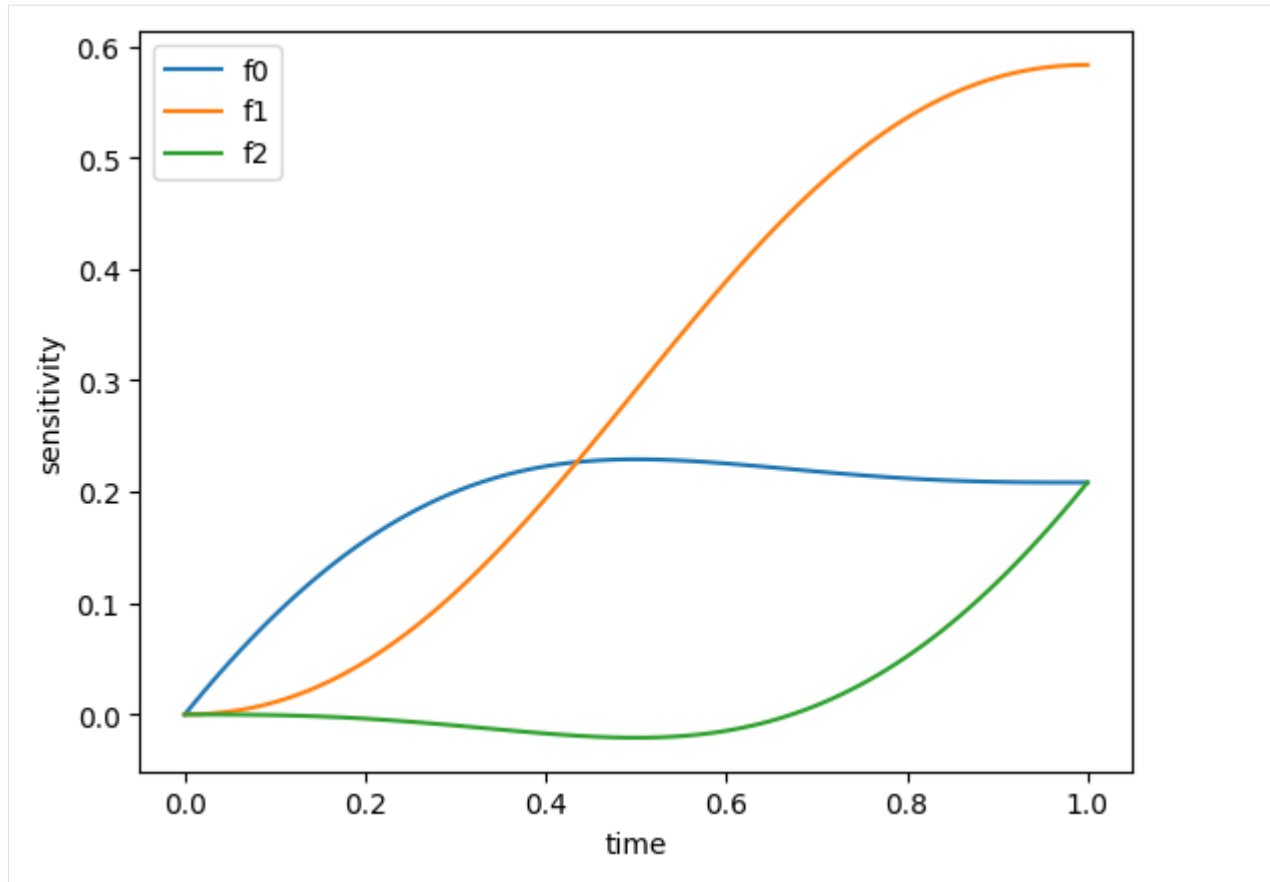
```
[13]: spline.plot(parameters, xlabel="time");
```



```
[14]: model, rdata = simulate(sbml_model, parameters)
```



```
[15]: # Sensitivities with respect to the spline values can be computed
fig, ax = plt.subplots()
ax.plot(rdata["t"], rdata.sx[:, 0], label=model.getParameterNames()[0])
ax.plot(rdata["t"], rdata.sx[:, 1], label=model.getParameterNames()[1])
ax.plot(rdata["t"], rdata.sx[:, 2], label=model.getParameterNames()[2])
ax.set_xlabel("time")
ax.set_ylabel("sensitivity")
ax.legend();
```

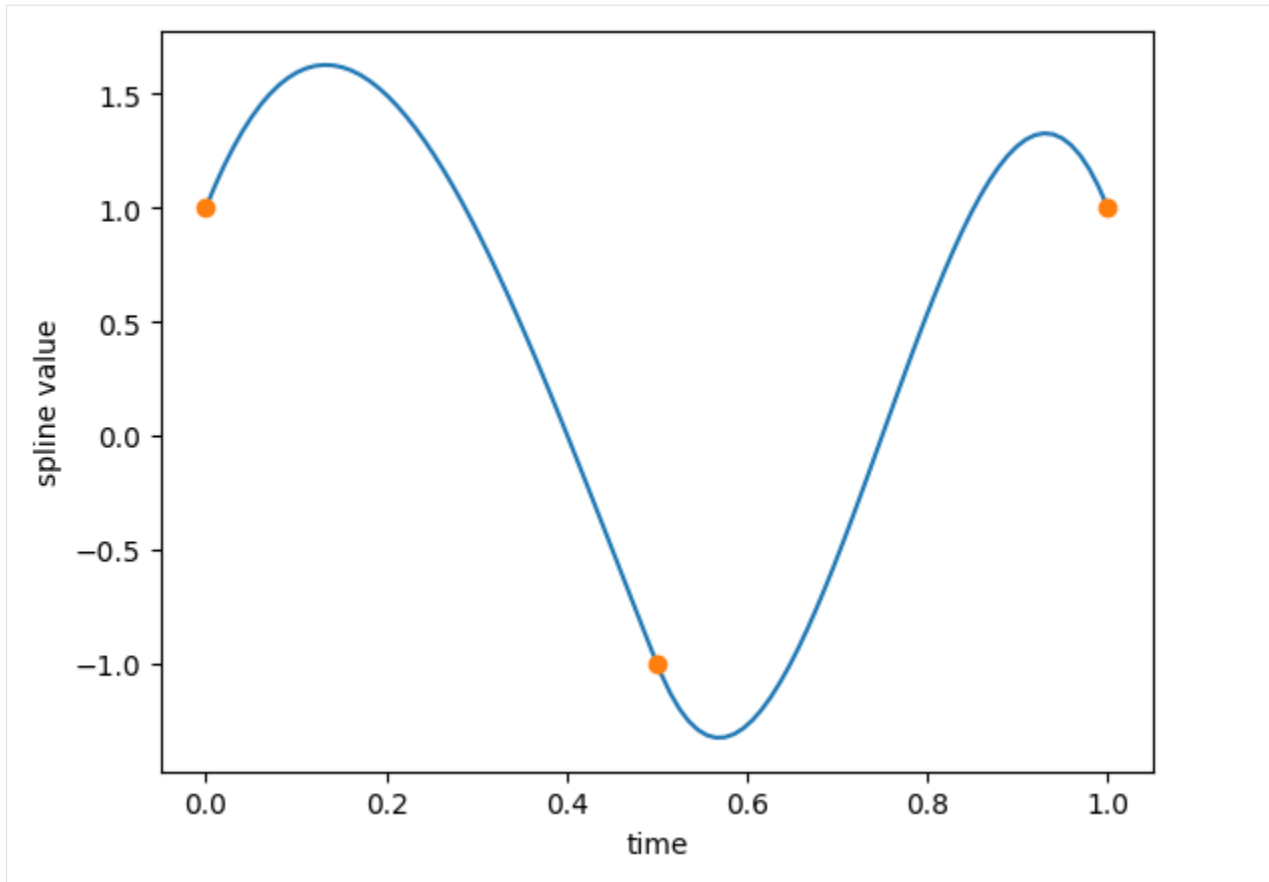


### Specifying derivatives, boundary conditions and extrapolation methods

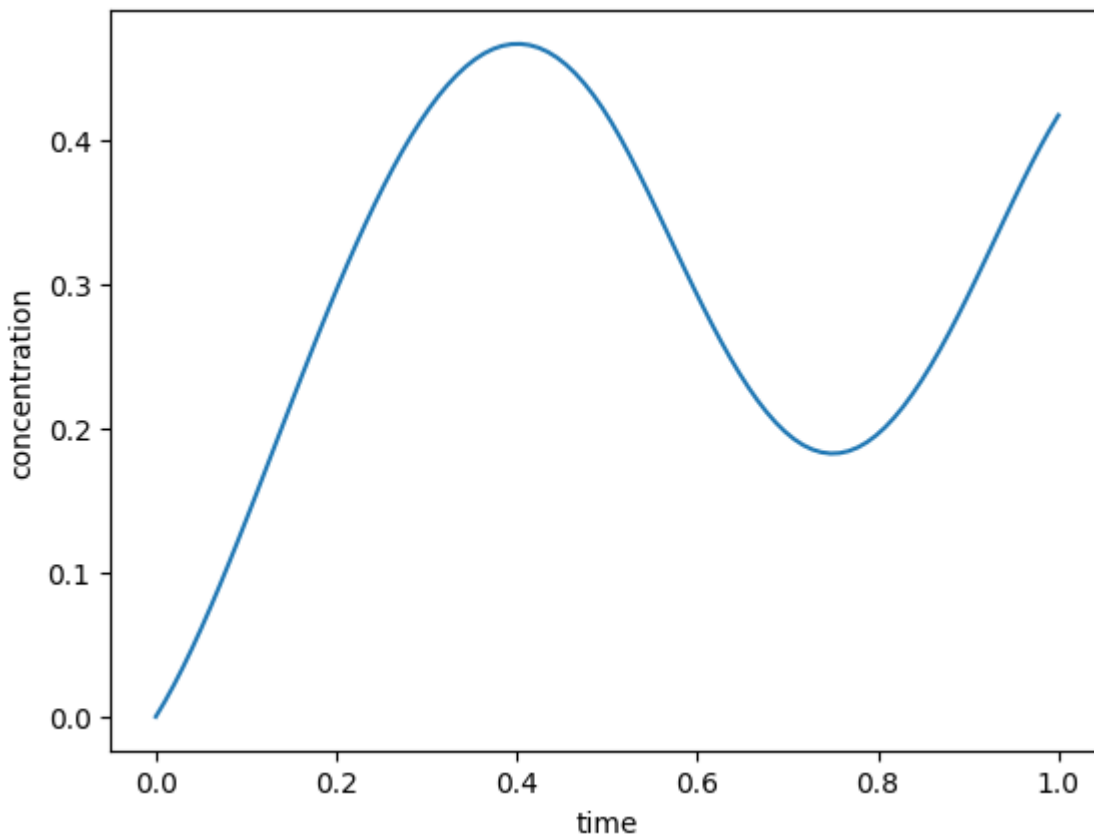
When derivatives are not specified in the `CubicHermiteSpline` constructor, they are computed automatically using finite differences and according to the boundary conditions. If their form is known a priori (e.g., they are known constants or functions of parameters), they can be passed explicitly to the spline constructor.

```
[16]: # A simple spline for which finite differencing would give a different result
spline = amici.splines.CubicHermiteSpline(
    sbml_id="f",
    evaluate_at=amici.sbml_utils.amici_time_symbol,
    nodes=amici.splines.UniformGrid(0, 1, number_of_nodes=3),
    values_at_nodes=[1.0, -1.0, 1.0],
    derivatives_at_nodes=[10.0, -10.0, -10.0],
)
```

```
[17]: spline.plot(xlabel="time");
```



```
[18]: # Simulation
sbml_doc = libsbml.SBMLReader().readSBML("example_splines.xml")
sbml_model = sbml_doc.getModel()
spline.add_to_sbml_model(sbml_model)
simulate(sbml_model, T=1);
```



The spline annotation in this case is

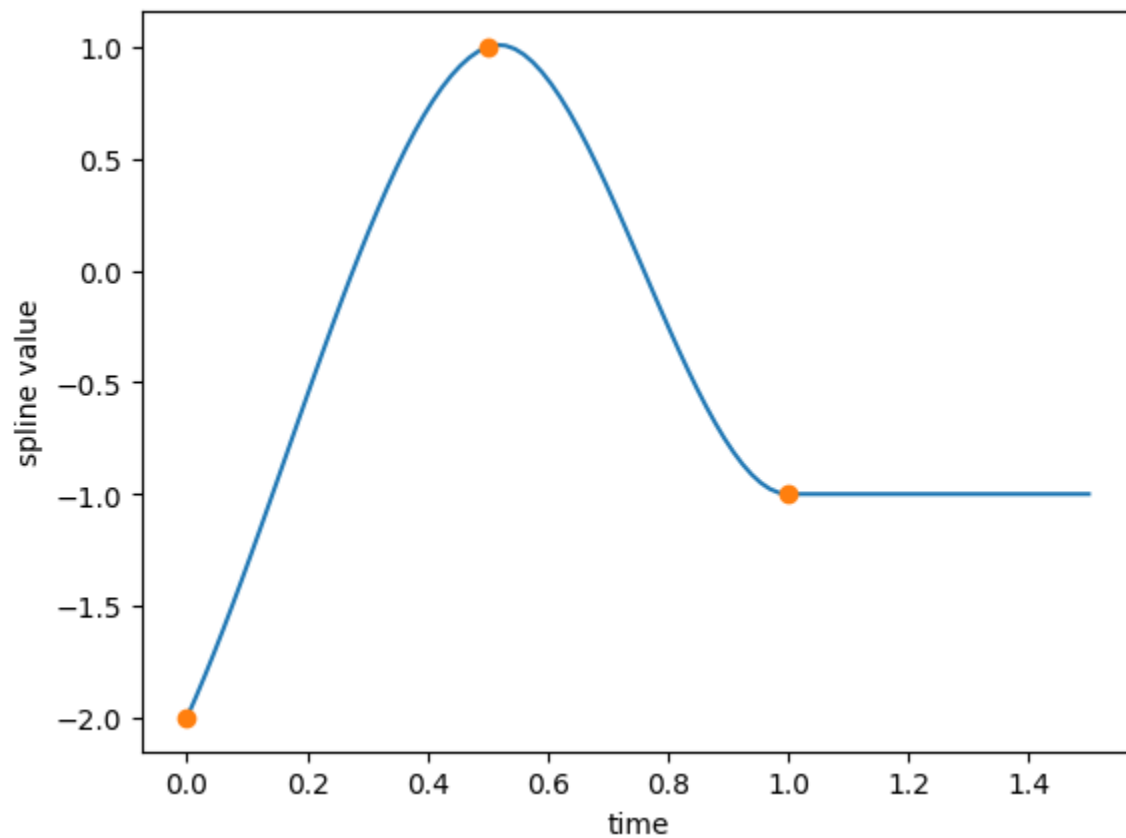
```
<amici:spline xmlns:amici="https://github.com/AMICI-dev/AMICI" amici:spline_method=
↪ "cubic_hermite">
  <amici:spline_evaluation_point> ... </amici:spline_evaluation_point>
  <amici:spline_uniform_grid> ... </amici:spline_uniform_grid>
  <amici:spline_values> ... </amici:spline_values>
  <amici:spline_derivatives>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <cn>10</cn>
    </math>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <cn>-10</cn>
    </math>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <cn>-10</cn>
    </math>
  </amici:spline_derivatives>
</amici:spline>
```

---

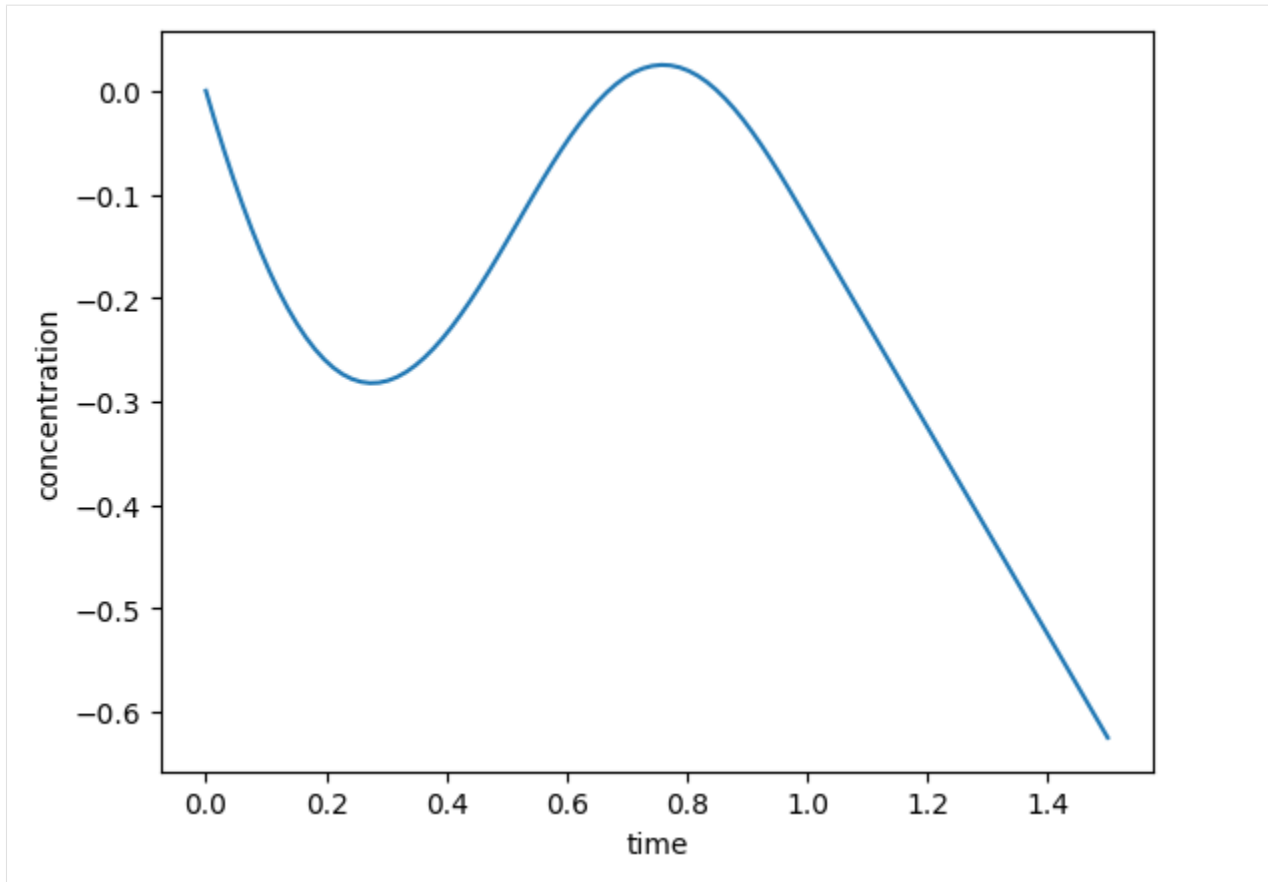
The default boundary conditions depend on the extrapolation method (which defaults to no extrapolation). For example, below we have a spline with constant extrapolation.

```
[19]: spline = amici.splines.CubicHermiteSpline(
    sbml_id="f",
    evaluate_at=amici.sbml_utils.amici_time_symbol,
    nodes=amici.splines.UniformGrid(0, 1, number_of_nodes=3),
    values_at_nodes=[-2, 1, -1],
    extrapolate=(
        None,
        "constant",
    ), # no extrapolation required on the left side
)
```

```
[20]: spline.plot(xlabel="time", xlim=(0, 1.5));
```



```
[21]: sbml_doc = libsbml.SBMLReader().readSBML("example_splines.xml")
sbml_model = sbml_doc.getModel()
spline.add_to_sbml_model(sbml_model)
simulate(sbml_model, T=1.5);
```



The spline annotation in this case is

```
<amici:spline xmlns:amici="https://github.com/AMICI-dev/AMICI" amici:spline_method=
↪ "cubic_hermite" amici:spline_bc="(no_bc, zeroderivative)" amici:spline_extrapolate=
↪ "(no_extrapolation, constant)">
  <amici:spline_evaluation_point> ... </amici:spline_evaluation_point>
  <amici:spline_uniform_grid> ... </amici:spline_uniform_grid>
  <amici:spline_values> ... </amici:spline_values>
</amici:spline>
```

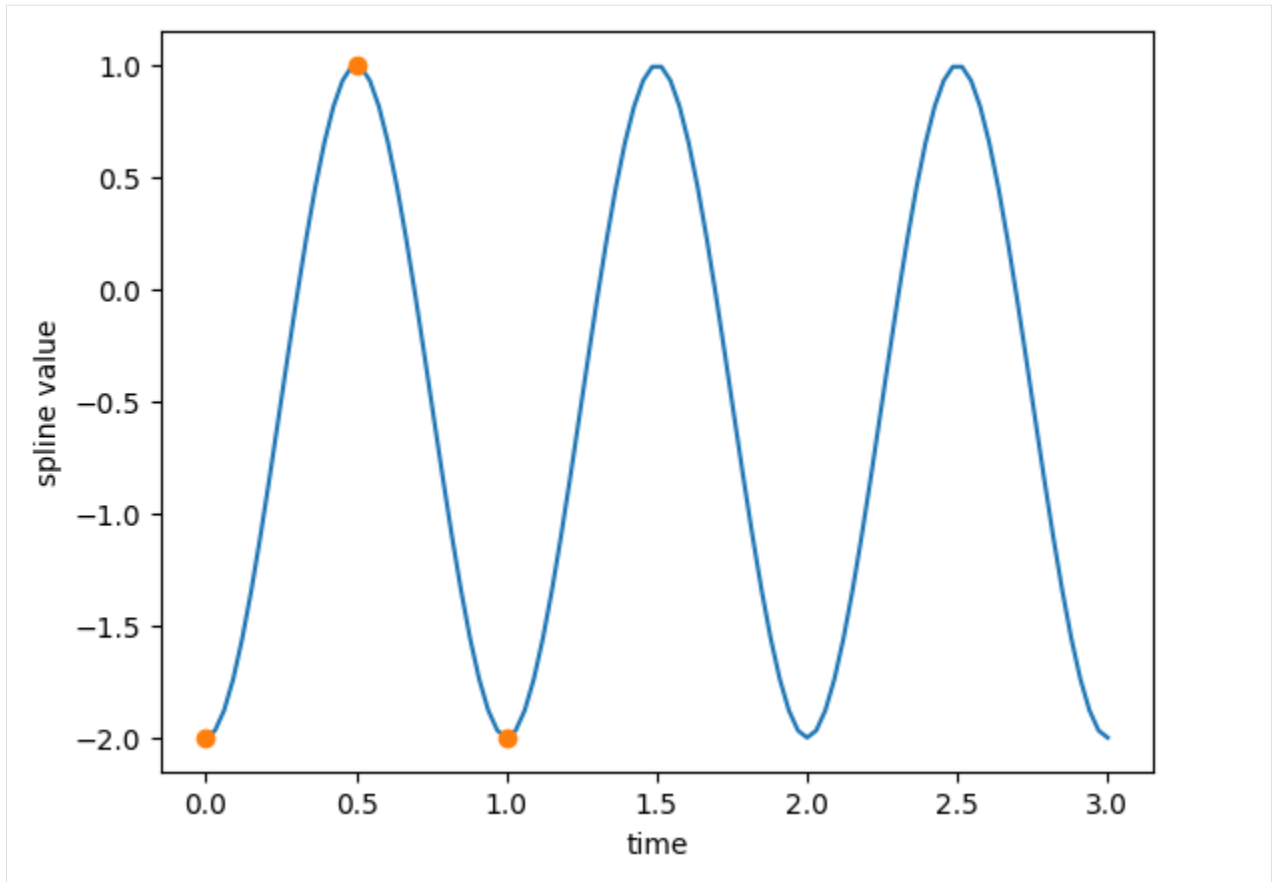
---

And here we have a periodic spline.

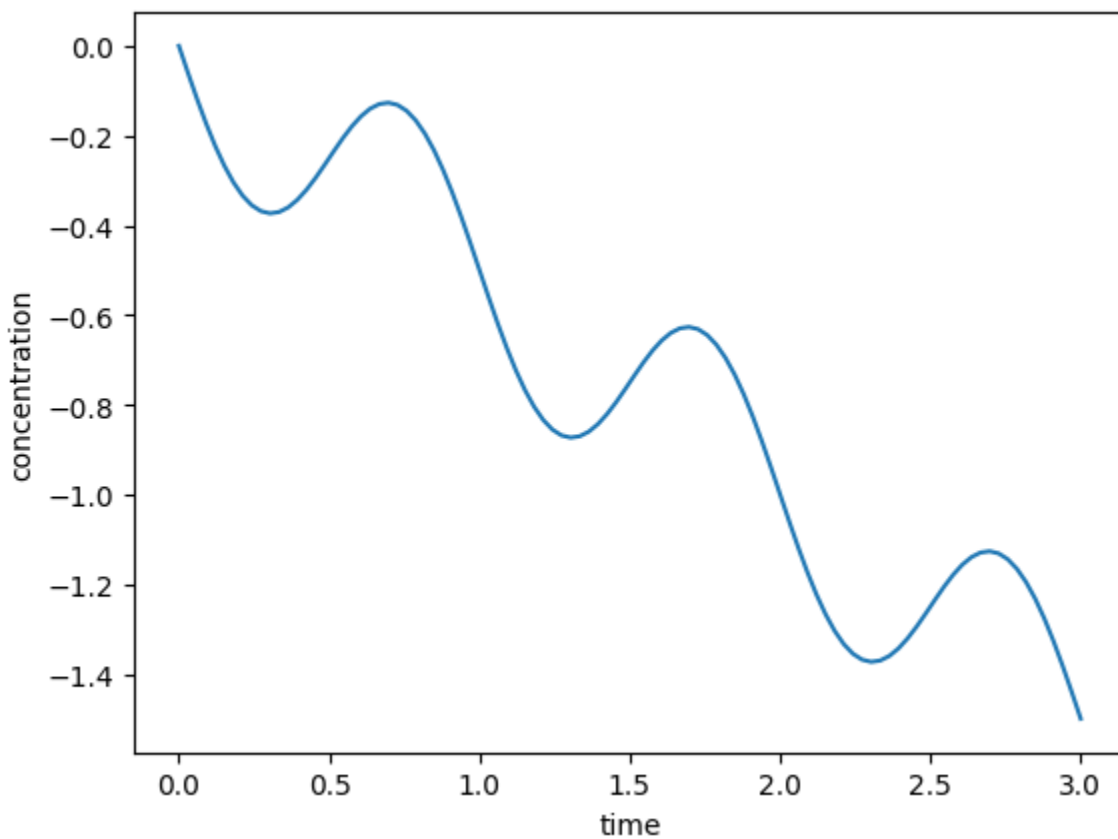
```
[22]: spline = amici.splines.CubicHermiteSpline(
        sbml_id="f",
        evaluate_at=amici.sbml_utils.amici_time_symbol,
        nodes=amici.splines.UniformGrid(0, 1, number_of_nodes=3),
        values_at_nodes=[-2, 1, -2], # first and last node must coincide
        extrapolate="periodic",
    )
```

```
[23]: spline.plot(xlabel="time", xlim=(0, 3));
```





```
[24]: sbml_doc = libsbml.SBMLReader().readSBML("example_splines.xml")
      sbml_model = sbml_doc.getModel()
      spline.add_to_sbml_model(sbml_model)
      simulate(sbml_model, T=3);
```



The spline annotation in this case is

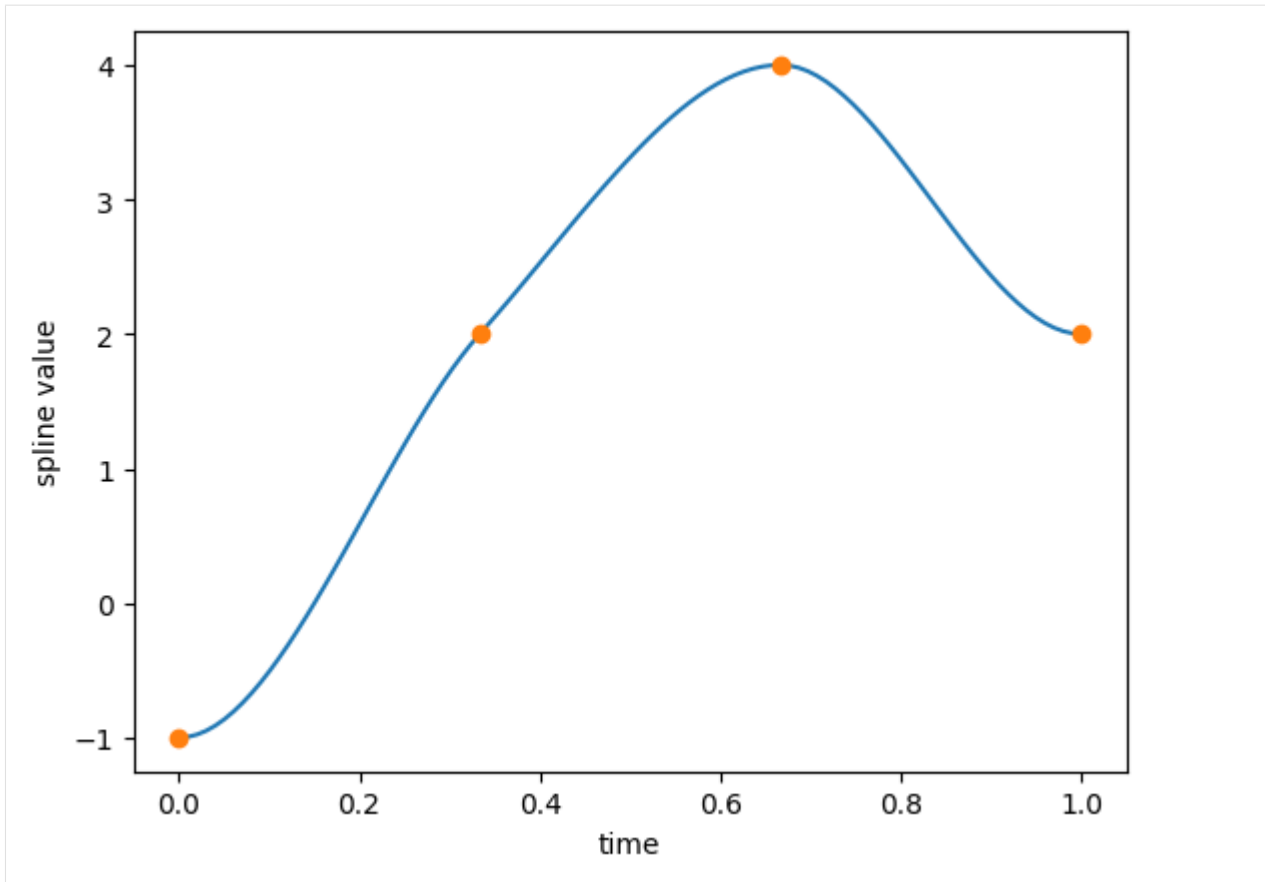
```
<amici:spline xmlns:amici="https://github.com/AMICI-dev/AMICI" amici:spline_method=
↪ "cubic_hermite" amici:spline_bc="periodic" amici:spline_extrapolate="periodic">
  <amici:spline_evaluation_point> ... </amici:spline_evaluation_point>
  <amici:spline_uniform_grid> ... </amici:spline_uniform_grid>
  <amici:spline_values> ... </amici:spline_values>
</amici:spline>
```

---

We can modify the spline's boundary conditions, for example requiring that the derivatives is zero.

```
[25]: spline = amici.splines.CubicHermiteSpline(
    sbml_id="f",
    evaluate_at=amici.sbml_utils.amici_time_symbol,
    nodes=amici.splines.UniformGrid(0, 1, number_of_nodes=4),
    values_at_nodes=[-1, 2, 4, 2],
    bc="zeroderivative",
)
```

```
[26]: spline.plot(xlabel="time");
```

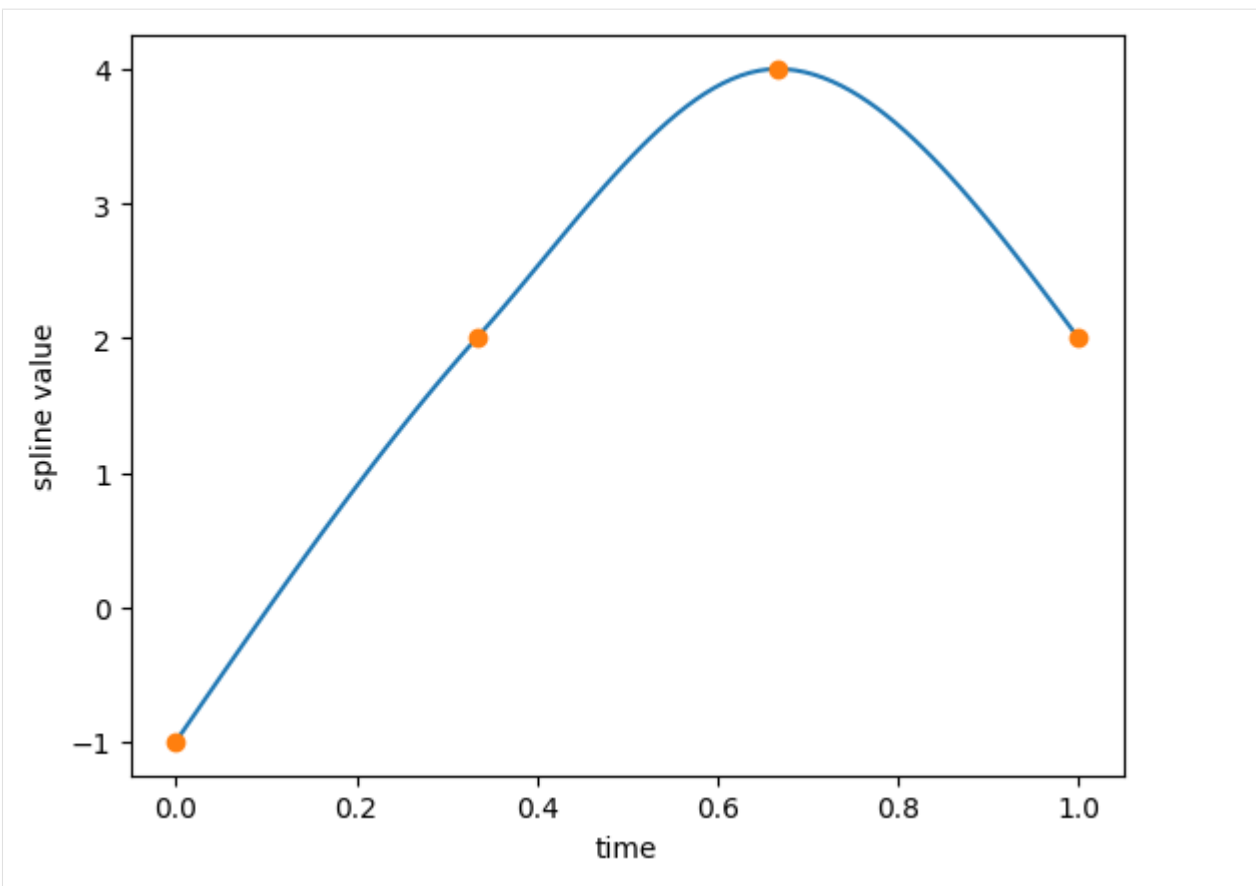


```
<amici:spline xmlns:amici="https://github.com/AMICI-dev/AMICI" amici:spline_method=
↪ "cubic_hermite" amici:spline_bc="zeroderivative">
  <amici:spline_evaluation_point> ... </amici:spline_evaluation_point>
  <amici:spline_uniform_grid> ... </amici:spline_uniform_grid>
  <amici:spline_values> ... </amici:spline_values>
</amici:spline>
```

Or we can impose natural boundary conditions.

```
[27]: spline = amici.splines.CubicHermiteSpline(
    sbml_id="f",
    evaluate_at=amici.sbml_utils.amici_time_symbol,
    nodes=amici.splines.UniformGrid(0, 1, number_of_nodes=4),
    values_at_nodes=[-1, 2, 4, 2],
    bc="natural",
)
```

```
[28]: spline.plot(xlabel="time");
```

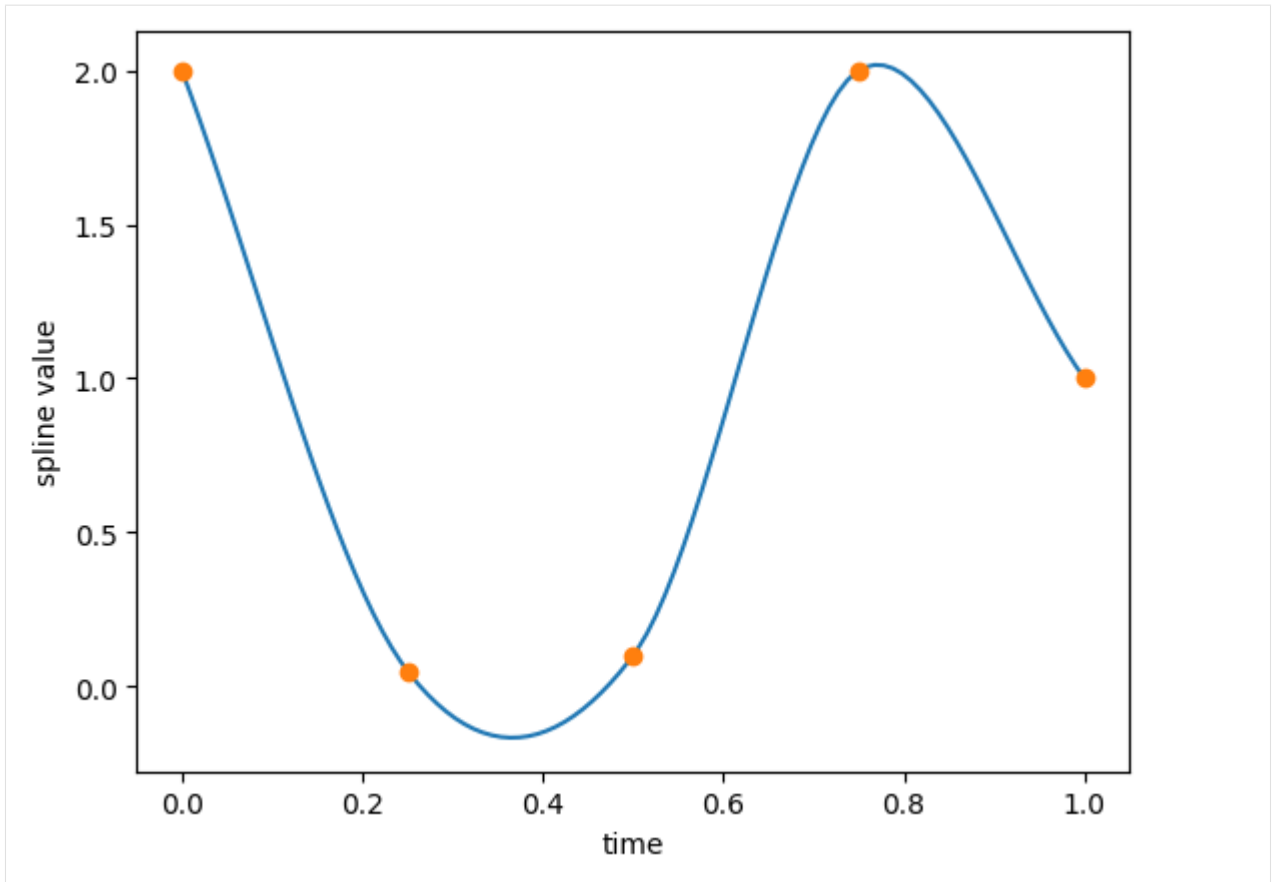


```
<amici:spline xmlns:amici="https://github.com/AMICI-dev/AMICI" amici:spline_method=
↪ "cubic_hermite" amici:spline_bc="natural">
  <amici:spline_evaluation_point> ... </amici:spline_evaluation_point>
  <amici:spline_uniform_grid> ... </amici:spline_uniform_grid>
  <amici:spline_values> ... </amici:spline_values>
</amici:spline>
```

Even if all node values are positive, due to under-shooting a cubic Hermite spline can assume negative values. In certain settings (e.g., when the spline represents a chemical reaction rate) this should be avoided. A possible solution is to carry out the interpolation in log-space (the resulting function is no longer a spline, but it is still a smooth interpolant).

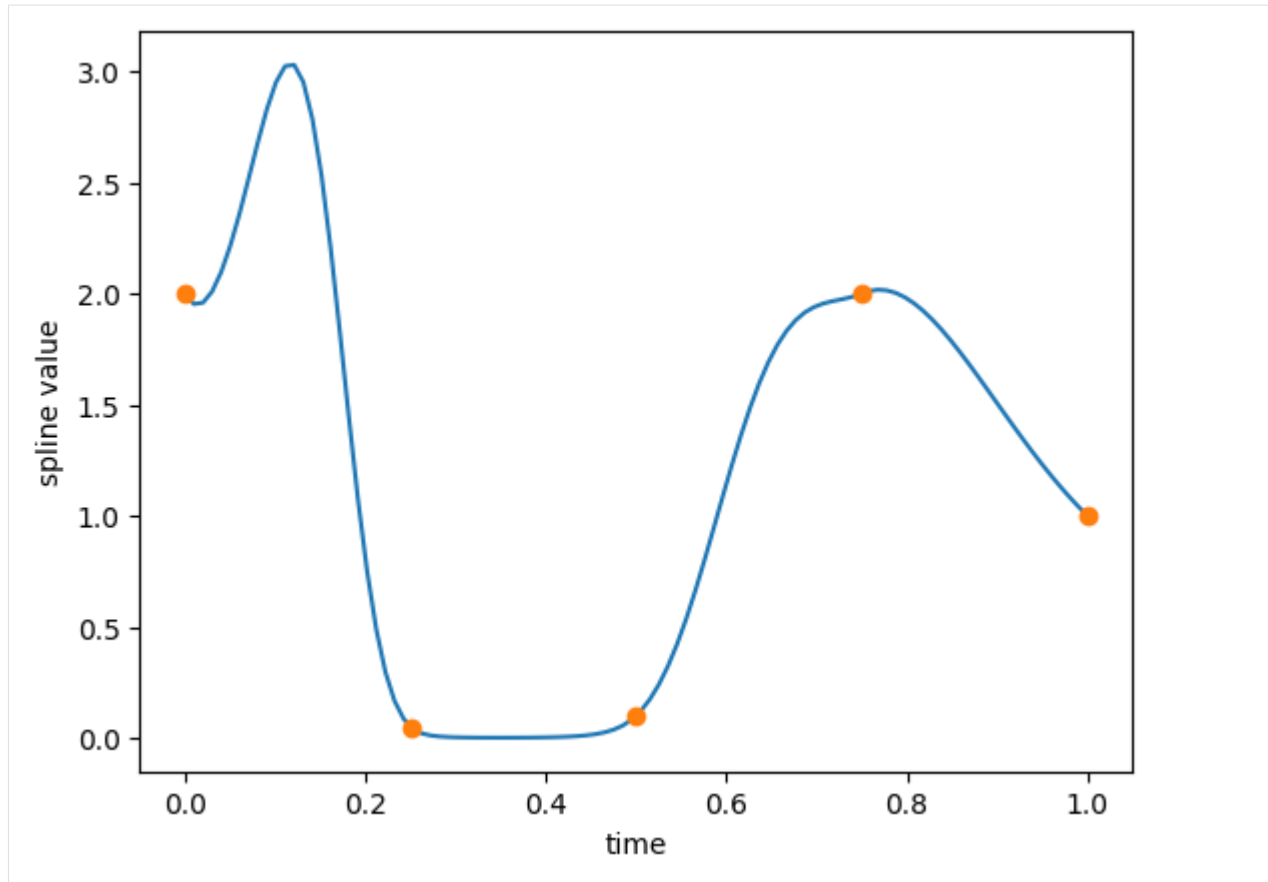
```
[29]: spline = amici.splines.CubicHermiteSpline(
    sbml_id="f",
    evaluate_at=amici.sbml_utils.amici_time_symbol,
    nodes=amici.splines.UniformGrid(0, 1, number_of_nodes=5),
    values_at_nodes=[2, 0.05, 0.1, 2, 1],
)
```

```
[30]: # This spline assumes negative values!
spline.plot(xlabel="time");
```



```
[31]: spline = amici.splines.CubicHermiteSpline(
    sbml_id="f",
    evaluate_at=amici.sbml_utils.amici_time_symbol,
    nodes=amici.splines.UniformGrid(0, 1, number_of_nodes=5),
    values_at_nodes=[2, 0.05, 0.1, 2, 1],
    logarithmic_parametrization=True,
)
```

```
[33]: # Instead of under-shooting we now have over-shooting,
      # but at least the "spline" is always positive
      spline.plot(xlabel="time");
```



The spline annotation in this case is

```
<amici:spline xmlns:amici="https://github.com/AMICI-dev/AMICI" amici:spline_method=
  ↪ "cubic_hermite" amici:spline_logarithmic_parametrization="true">
  <amici:spline_evaluation_point> ... </amici:spline_evaluation_point>
  <amici:spline_uniform_grid> ... </amici:spline_uniform_grid>
  <amici:spline_values> ... </amici:spline_values>
</amici:spline>
```

### Comparing model import time for the SBML-native piecewise implementation and the AMICI spline implementation

```
[33]: import pandas as pd
import seaborn as sns
import tempfile
import time
```

```
[34]: nruns = 6 # number of replicates
num_nodes = [
    5,
    10,
    15,
    20,
```

(continues on next page)

(continued from previous page)

```

    25,
    30,
    40,
] # benchmark model import for these node numbers
amici_only_nodes = [
    50,
    75,
    100,
    125,
    150,
    175,
    200,
    225,
    250,
] # for these node numbers, only benchmark the annotation-based implementation

```

```

[35]: # If running as a Github action, just do the minimal amount of work required to check_
      ↪ whether the code is working
if os.getenv("GITHUB_ACTIONS") is not None:
    nruns = 1
    num_nodes = [4]
    amici_only_nodes = [5]

```

```

[36]: df = None
for n in num_nodes + amici_only_nodes:
    # Create model
    spline = amici.splines.CubicHermiteSpline(
        sbml_id="f",
        evaluate_at=amici.sbml_utils.amici_time_symbol,
        nodes=amici.splines.UniformGrid(0, 1, number_of_nodes=n),
        values_at_nodes=np.random.rand(n),
    )
    sbml_doc = libsbml.SBMLReader().readSBML("example_splines.xml")
    sbml_model = sbml_doc.getModel()
    spline.add_to_sbml_model(sbml_model)
    # Benchmark model creation
    timings_amici = []
    timings_piecewise = []
    for _ in range(nruns):
        with tempfile.TemporaryDirectory() as tmpdir:
            t0 = time.perf_counter_ns()
            amici.SbmlImporter(sbml_model).sbml2amici("benchmark", tmpdir)
            dt = time.perf_counter_ns() - t0
            timings_amici.append(dt / 1e9)
    if n in num_nodes:
        with tempfile.TemporaryDirectory() as tmpdir:
            t0 = time.perf_counter_ns()
            amici.SbmlImporter(
                sbml_model, discard_annotations=True
            ).sbml2amici("benchmark", tmpdir)
            dt = time.perf_counter_ns() - t0

```

(continues on next page)

(continued from previous page)

```

        timings_pieewise.append(dt / 1e9)
# Append benchmark data to dataframe
df_amici = pd.DataFrame(
    dict(num_nodes=n, time=timings_amici, use_annotations=True)
)
df_pieewise = pd.DataFrame(
    dict(num_nodes=n, time=timings_pieewise, use_annotations=False)
)
if df is None:
    df = pd.concat(
        [df_amici, df_pieewise], ignore_index=True, verify_integrity=True
    )
else:
    df = pd.concat(
        [df, df_amici, df_pieewise],
        ignore_index=True,
        verify_integrity=True,
    )

```

```

[88]: kwargs = dict(markersize=7.5)
df_avg = df.groupby(["use_annotations", "num_nodes"]).mean().reset_index()
fig, ax = plt.subplots(1, 1, figsize=(6.5, 3.5))
ax.plot(
    df_avg[np.logical_not(df_avg["use_annotations"])]["num_nodes"],
    df_avg[np.logical_not(df_avg["use_annotations"])]["time"],
    ".",
    label="MathML pieewise",
    **kwargs,
)
ax.plot(
    df_avg[df_avg["use_annotations"]]["num_nodes"],
    df_avg[df_avg["use_annotations"]]["time"],
    ".",
    label="AMICI annotations",
    **kwargs,
)
ax.set_ylabel("model import time (s)")
ax.set_xlabel("number of spline nodes")
ax.set_yscale("log")
ax.yaxis.set_major_formatter(
    mpl.ticker.FuncFormatter(lambda x, pos: f"{x:.0f}")
)
ax.xaxis.set_ticks(
    [
        10,
        20,
        30,
        40,
        60,
        70,
        80,
        90,
    ]
)

```

(continues on next page)

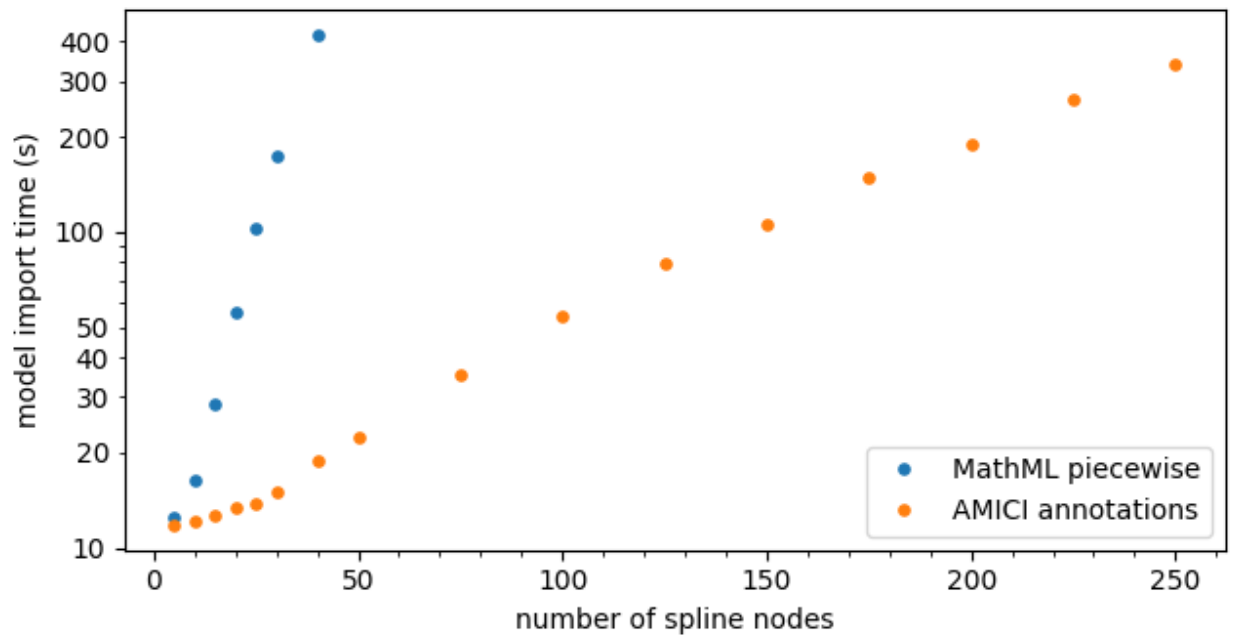


(continued from previous page)

```

110,
120,
130,
140,
160,
170,
180,
190,
210,
220,
230,
240,
260,
],
minor=True,
)
ax.yaxis.set_ticks(
    [20, 30, 40, 50, 60, 70, 80, 90, 200, 300, 400],
    ["20", "30", "40", "50", None, None, None, None, "200", "300", "400"],
    minor=True,
)
ax.legend()
ax.figure.tight_layout()
# ax.figure.savefig('benchmark_import.pdf')

```



### 10.2.12 Spline implementation of JAK2-STAT5 signaling pathway

In this notebook a practical example of the usage of AMICI spline functionalities is shown. The model under consideration is the JAK2-STAT5 signaling pathway (Swameye et al., 2003), in which the dynamics of the system depend on a measured input function (the quantity pEpoR in the model).

Following the approach of (Schelker et al., 2012), a continuous approximation of this input function is estimated together with the other parameters. As in the original paper, we will use a spline with logarithmic parameterization in order to enforce the positivity constraint.

The model of the signaling pathway will be implemented in SBML using AMICI's spline annotations, experimental data integrated using the PETab format and parameter estimation will be carried out using the `pyPESTO` library.

```
[ ]: %pip install pypesto
```

```
[ ]: %pip install fides
```

```
[1]: import os
import math
import logging
import contextlib
import multiprocessing
import copy

import numpy as np
import sympy as sp
import pandas as pd
from matplotlib import pyplot as plt

import libsbml
import amici
import petab
import pypesto
import pypesto.petab
```

```
[2]: # Number of multi-starts for MAP estimation
n_starts = 150
# n_starts = 0 # when loading results
```

```
[ ]: # Set default pypesto engine/optimizer
pypesto_optimizer = pypesto.optimize.FidesOptimizer(verbose=logging.WARNING)
pypesto_engine = pypesto.engine.MultiProcessEngine()
```

```
[4]: # If running as a Github action, just do the minimal amount of work required to check_
↳ whether the code is working
if os.getenv("GITHUB_ACTIONS") is not None:
    n_starts = 15
    pypesto_optimizer = pypesto.optimize.FidesOptimizer(
        verbose=logging.WARNING, options=dict(maxiter=10)
    )
    pypesto_engine = pypesto.engine.SingleCoreEngine()
```

```
[5]: # A dictionary to store different approaches for a final comparison
all_results = {}
```

### Spline approximation with few nodes, using finite differences for the derivatives

As a first attempt, we fix a small amount of nodes, create new parameters for the values of the splines at the nodes and let AMICI compute the derivative at the nodes by using finite differences.

### Creating the P<sub>E</sub>tab model

```
[6]: # Problem name
name = "Swameye_PNAS2003_5nodes_FD"
```

First, we create a spline to represent the input function pEpoR, parametrized by its values at the nodes. Since the value of the input function reaches its steady state by the end of the experiment, we extrapolate constantly after that (useful if we need to simulate the model after the last spline node).

```
[7]: # Create spline for pEpoR
nodes = [0, 5, 10, 20, 60]
values_at_nodes = [
    sp.Symbol(f"pEpoR_t{str(t).replace('.', '_dot')}") for t in nodes
] # new parameter symbols for spline values
spline = amici.splines.CubicHermiteSpline(
    sbml_id="pEpoR", # matches name of species in SBML model
    evaluate_at=amici.sbml_utils.amici_time_symbol, # the spline is evaluated at the
    ↪current time
    nodes=nodes,
    values_at_nodes=values_at_nodes, # values at the nodes (in linear scale)
    extrapolate=(None, "constant"), # because steady state is reached
    bc="auto", # automatically determined from extrapolate (bc at right end will be
    ↪'zero_derivative')
    logarithmic_parametrization=True,
)
```

We can then add the spline to a skeleton SBML model based on the d2d implementation by (Schelker et al., 2012). The skeleton SBML model defines a species pEpoR which interacts with the other species, but has no reactions or rate rules of its own. The code below creates an assignment rule for pEpoR using the spline formula, completing the model. The parameters pEpoR\_t\* are automatically added to the SBML model too (using nominal values of 0.1 and declaring them to be constant).

```
[8]: # Add spline formula to SBML model
sbml_doc = libsbml.SBMLReader().readSBML(
    os.path.join("Swameye_PNAS2003", "swameye2003_model.xml")
)
sbml_model = sbml_doc.getModel()
spline.add_to_sbml_model(
    sbml_model, auto_add=True, y_nominal=0.1, y_constant=True
)
```

A skeleton P<sub>E</sub>tab problem is provided, containing parameter bounds, observable definitions and experimental data. Of particular relevance is the noise model used for the measurements of pEpoR, normal additive noise with standard

deviation equal to  $0.0274 + 0.1 * p_{\text{EpoR}}$ ; this is the same choice used in (Schelker et al., 2012), where it was estimated from experimental replicates.

However, the parameters associated to the spline are to be added too. The code below defines parameter bounds for them according to the PETab format and then creates a full PETab problem integrating them together with the edited SBML file. The condition, measurement and observable PETab tables do not require additional modification and can be used as they are.

```
[9]: # Extra parameters associated to the spline
spline_parameters_df = pd.DataFrame(
    dict(
        parameterScale="log",
        lowerBound=0.001,
        upperBound=10,
        nominalValue=0.1,
        estimate=1,
    ),
    index=pd.Series(list(map(str, values_at_nodes)), name="parameterId"),
)
```

```
[10]: # Create PETab problem
petab_problem = petab.Problem(
    sbml_model,
    condition_df=petab.conditions.get_condition_df(
        os.path.join("Swameye_PNAS2003", "swameye2003_conditions.tsv")
    ),
    measurement_df=petab.measurements.get_measurement_df(
        os.path.join("Swameye_PNAS2003", "swameye2003_measurements.tsv")
    ),
    parameter_df=petab.core.concat_tables(
        [
            os.path.join("Swameye_PNAS2003", "swameye2003_parameters.tsv"),
            spline_parameters_df,
        ],
        petab.parameters.get_parameter_df,
    ),
    observable_df=petab.observables.get_observable_df(
        os.path.join("Swameye_PNAS2003", "swameye2003_observables.tsv")
    ),
)
```

The resulting PETab problem can be checked for errors and exported to disk if needed.

```
[11]: # Check whether PETab model is valid
assert not petab.lint_problem(petab_problem)
```

```
[12]: # Save PETab problem to disk
# import shutil
# shutil.rmtree(name, ignore_errors=True)
# os.mkdir(name)
# petab_problem.to_files_generic(prefix_path=name)
```

## Creating the pyPESTO problem

We can now create a pyPESTO problem directly from the PETab problem. Due to technical limitations in AMICI, currently the PETab problem has to be “flattened” before it can be simulated from, but such operation is merely syntactical and thus does not change the essence of the model.

```
[13]: # Problem must be "flattened" to be used with AMICI
petab.core.flatten_timepoint_specific_output_overrides(petab_problem)
```

```
[14]: # Check whether simulation from the PETab problem works
# import amici.petab_simulate
# simulator = amici.petab_simulate.PetabSimulator(petab_problem)
# simulator.simulate(noise=False)
```

```
[15]: # Import PETab problem into pyPESTO
pypesto_problem = pypesto.petab.PetabImporter(
    petab_problem, model_name=name
).create_problem()

# Increase maximum number of steps for AMICI
pypesto_problem.objective.amici_solver.setMaxSteps(10**5)
```

## Maximum Likelihood estimation

Using pyPESTO we can optimize for the parameter vector that maximizes the probability of observing the experimental data (maximum likelihood estimation).

A multistart method with local gradient-based optimization is used and the results of each multistart can be visualized in a waterfall plot.

```
[16]: # Load existing results if available
if os.path.exists(f"{name}.h5"):
    pypesto_result = pypesto.store.read_result(
        f"{name}.h5", problem=pypesto_problem
    )
else:
    pypesto_result = None
# Overwrite
# pypesto_result = None

[ ]: # Parallel multistart optimization with pyPESTO and FIDES
if n_starts > 0:
    if pypesto_result is None:
        new_ids = [str(i) for i in range(n_starts)]
    else:
        last_id = max(int(i) for i in pypesto_result.optimize_result.id)
        new_ids = [str(i) for i in range(last_id + 1, last_id + n_starts + 1)]
    pypesto_result = pypesto.optimize.minimize(
        pypesto_problem,
        n_starts=n_starts,
        ids=new_ids,
```

(continues on next page)

(continued from previous page)

```

    optimizer=pypesto_optimizer,
    engine=pypesto_engine,
    result=pypesto_result,
)
pypesto_result.optimize_result.sort()
if pypesto_result.optimize_result.x[0] is None:
    raise Exception(
        "All multistarts failed (n_starts is probably too small)! If this error
        occurred during CI, just run the workflow again."
    )

```

```

[18]: # Save results to disk
      # pypesto.store.write_result(pypesto_result, f'{name}.h5', overwrite=True)

```

```

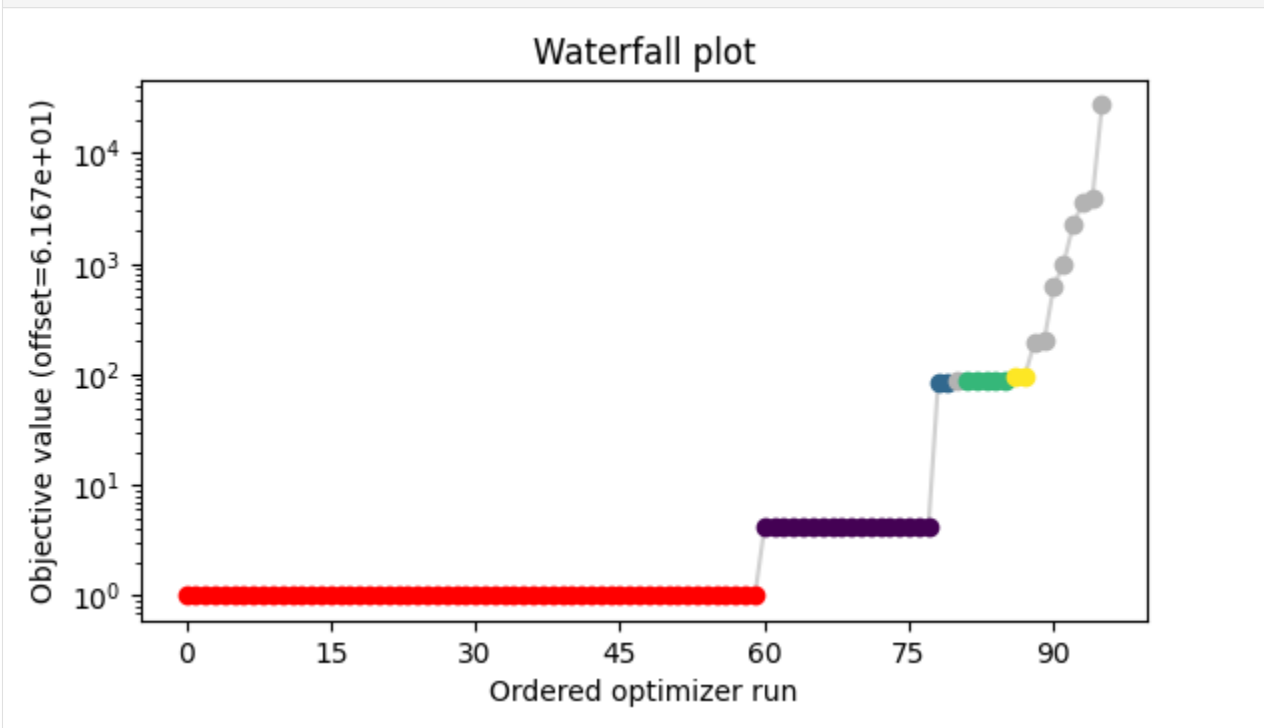
[19]: # Print result table
      # pypesto_result.optimize_result.as_dataframe()

```

```

[20]: # Visualize the results of the multistarts
      pypesto.visualize.waterfall(pypesto_result, size=[6.5, 3.5]);

```



Below the maximum likelihood estimates for pEpoR and the other observables are plotted, together with the experimental measurements.

To assess whether the noise model used in the observable is reasonable, we have also plotted 2-sigma error bands for pEpoR.

```

[21]: # Functions for simulating observables given a parameter vector
      def _simulate(x=None, *, problem=None, result=None, N=500, **kwargs):
          if result is None:

```

(continues on next page)

(continued from previous page)

```

        result = pypesto_result
    if problem is None:
        problem = pypesto_problem
    if x is None:
        x = result.optimize_result.x[0]
    if N is None:
        objective = problem.objective
    else:
        objective = problem.objective.set_custom_timepoints(
            timepoints_global=np.linspace(0, 60, N)
        )
    if len(x) != len(problem.x_free_indices):
        x = x[problem.x_free_indices]
    simresult = objective(x, return_dict=True, **kwargs)
    return problem, simresult["rdatas"][0]

def simulate_pEpoR(x=None, **kwargs):
    problem, rdata = _simulate(x, **kwargs)
    assert problem.objective.amici_model.getObservableIds()[0].startswith(
        "pEpoR"
    )
    return rdata["t"], rdata["y"][:, 0]

def simulate_pSTAT5(x=None, **kwargs):
    problem, rdata = _simulate(x, **kwargs)
    assert problem.objective.amici_model.getObservableIds()[1].startswith(
        "pSTAT5"
    )
    return rdata["t"], rdata["y"][:, 1]

def simulate_tSTAT5(x=None, **kwargs):
    problem, rdata = _simulate(x, **kwargs)
    assert problem.objective.amici_model.getObservableIds()[-1].startswith(
        "tSTAT5"
    )
    return rdata["t"], rdata["y"][:, -1]

# Experimental data
df_measurements = petab.measurements.get_measurement_df(
    os.path.join("Swameye_PNAS2003", "swameye2003_measurements.tsv")
)
df_pEpoR = df_measurements[
    df_measurements["observableId"].str.startswith("pEpoR")
]
df_pSTAT5 = df_measurements[
    df_measurements["observableId"].str.startswith("pSTAT5")
]
df_tSTAT5 = df_measurements[

```

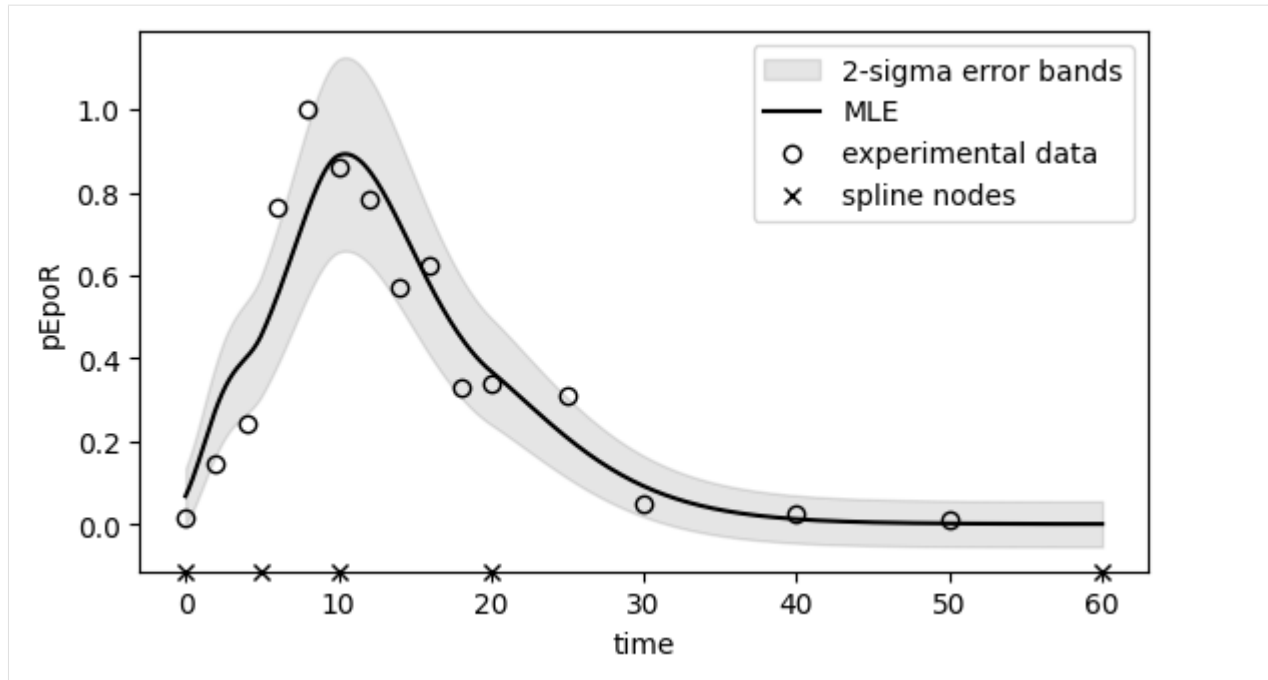
(continues on next page)

(continued from previous page)

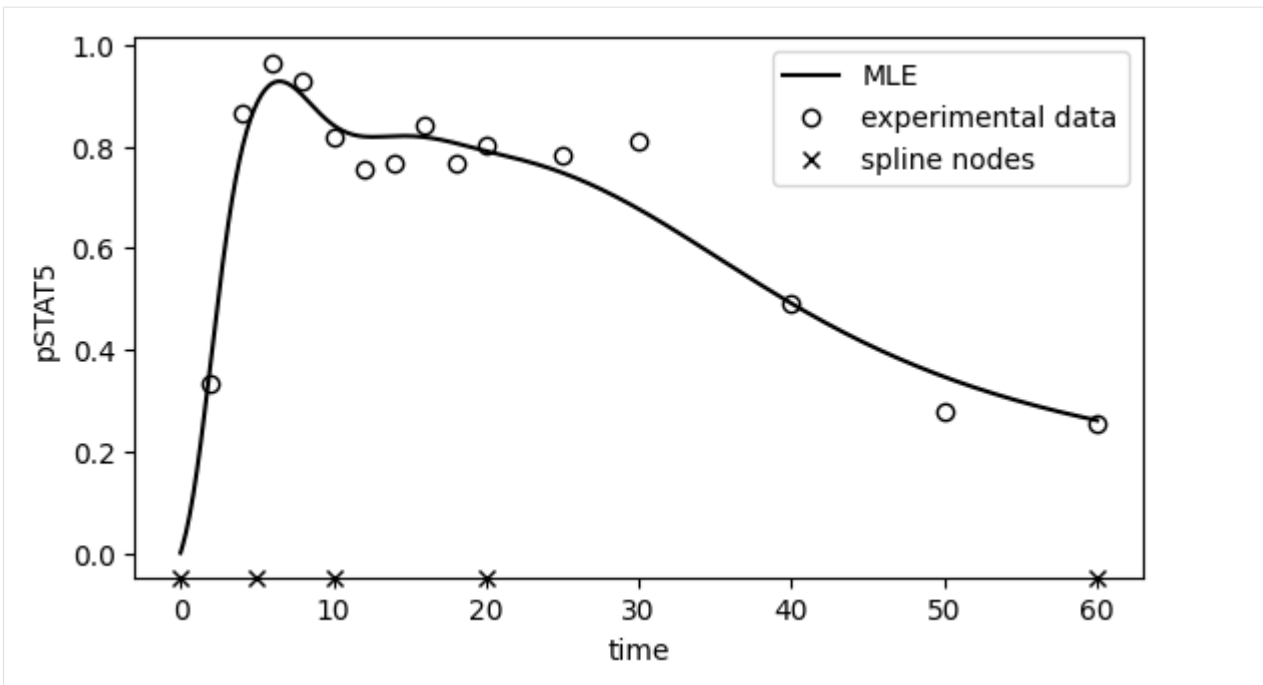
```
df_measurements["observableId"].str.startswith("tSTAT5")
]
```

```
[22]: # Plot ML fit for pEpoR
fig, ax = plt.subplots(figsize=(6.5, 3.5))
t, pEpoR = simulate_pEpoR()
sigma_pEpoR = 0.0274 + 0.1 * pEpoR
ax.fill_between(
    t,
    pEpoR - 2 * sigma_pEpoR,
    pEpoR + 2 * sigma_pEpoR,
    color="black",
    alpha=0.10,
    interpolate=True,
    label="2-sigma error bands",
)
ax.plot(t, pEpoR, color="black", label="MLE")
ax.plot(
    df_pEpoR["time"],
    df_pEpoR["measurement"],
    "o",
    color="black",
    markerfacecolor="none",
    label="experimental data",
)
ylim1 = ax.get_ylim()[0]
ax.plot(
    nodes,
    len(nodes) * [ylim1],
    "x",
    color="black",
    label="spline nodes",
    zorder=10,
    clip_on=False,
)
ax.set_ylim(ylim1, ax.get_ylim()[1])
ax.set_xlabel("time")
ax.set_ylabel("pEpoR")
ax.legend();
```

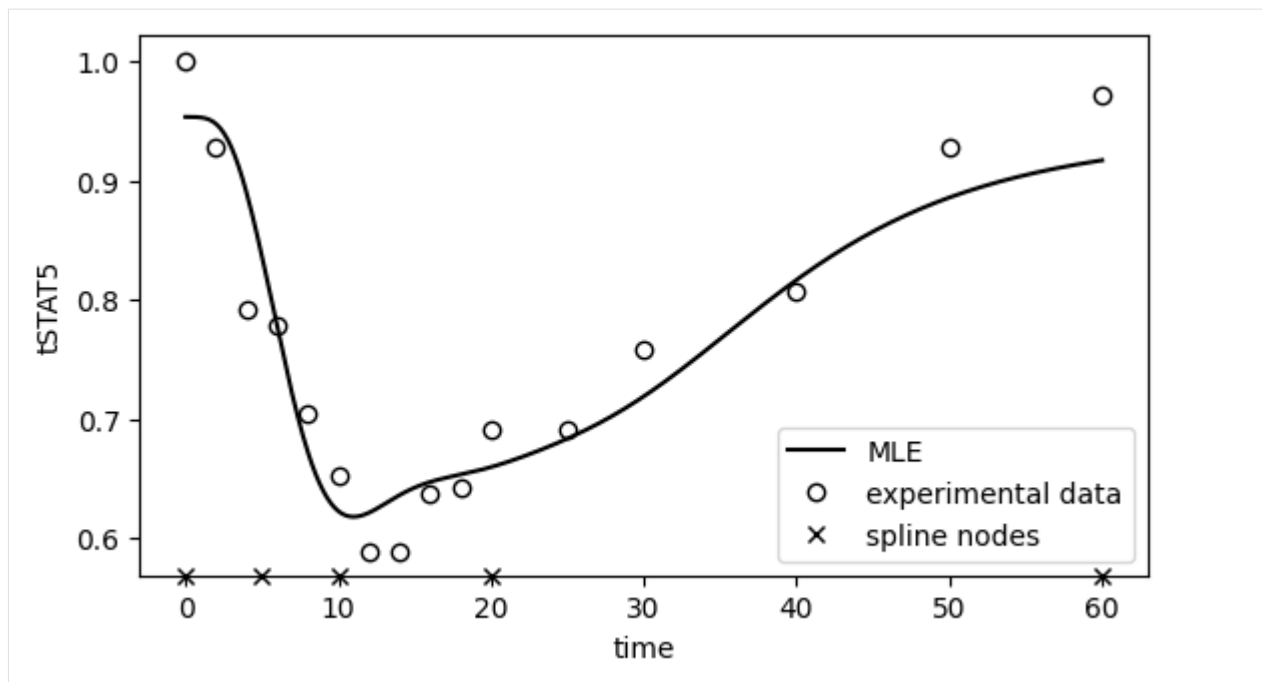




```
[23]: # Plot ML fit for pSTAT5
fig, ax = plt.subplots(figsize=(6.5, 3.5))
t, pSTAT5 = simulate_pSTAT5()
ax.plot(t, pSTAT5, color="black", label="MLE")
ax.plot(
    df_pSTAT5["time"],
    df_pSTAT5["measurement"],
    "o",
    color="black",
    markerfacecolor="none",
    label="experimental data",
)
ylim1 = ax.get_ylim()[0]
ax.plot(
    nodes,
    len(nodes) * [ylim1],
    "x",
    color="black",
    label="spline nodes",
    zorder=10,
    clip_on=False,
)
ax.set_ylim(ylim1, ax.get_ylim()[1])
ax.set_xlabel("time")
ax.set_ylabel("pSTAT5")
ax.legend();
```



```
[24]: # Plot ML fit for tSTAT5
fig, ax = plt.subplots(figsize=(6.5, 3.5))
t, tSTAT5 = simulate_tSTAT5()
ax.plot(t, tSTAT5, color="black", label="MLE")
ax.plot(
    df_tSTAT5["time"],
    df_tSTAT5["measurement"],
    "o",
    color="black",
    markerfacecolor="none",
    label="experimental data",
)
ylim1 = ax.get_ylim()[0]
ax.plot(
    nodes,
    len(nodes) * [ylim1],
    "x",
    color="black",
    label="spline nodes",
    zorder=10,
    clip_on=False,
)
ax.set_ylim(ylim1, ax.get_ylim()[1])
ax.set_xlabel("time")
ax.set_ylabel("tSTAT5")
ax.legend();
```



```
[25]: # Store results for later
all_results["5 nodes, FD"] = (pypesto_problem, pypesto_result)
```

### Spline approximation with many nodes, using finite differences for the derivatives

Five nodes is arguably not enough to represent all plausible input choices. Increasing the number of nodes would give the spline more freedom and it can be done with minimal changes to the example above. However, more degrees of freedom mean more chance of overfitting. Thus, following (Schelker et al., 2012), we will add a regularization term consisting in the squared L2 norm of the spline's curvature, which promotes smoother and less oscillating functions. The value for the regularization strength  $\lambda$  is chosen by comparing the sum of squared normalized residuals with its expected value, which can be computed by assuming it is roughly  $\chi^2$ -distributed.

### Creating the PEtab model

```
[26]: # Problem name
name = "Swameye_PNAS2003_15nodes_FD"

[27]: # Create spline for pEpoR
nodes = [0, 2.5, 5.0, 7.5, 10.0, 12.5, 15.0, 17.5, 20, 25, 30, 35, 40, 50, 60]
values_at_nodes = [
    sp.Symbol(f"pEpoR_t{str(t).replace('.', '_dot_')}") for t in nodes
]
spline = amici.splines.CubicHermiteSpline(
    sbml_id="pEpoR",
    evaluate_at=amici.sbml_utils.amici_time_symbol,
    nodes=nodes,
    values_at_nodes=values_at_nodes,
    extrapolate=(None, "constant"),
```

(continues on next page)

(continued from previous page)

```

    bc="auto",
    logarithmic_parametrization=True,
)

```

The regularization term can be easily computed by symbolic manipulation of the spline expression using AMICI and SymPy. Since it is very commonly used, we already provide a function for it in AMICI. Note: we regularize the curvature of the spline, which for positivity-enforcing spline is the logarithm of the function.

In order add the regularization term to the PEstab likelihood, a dummy observable has to be created.

```

[28]: # Compute L2 norm of the curvature of pEpoR
regularization = spline.squared_L2_norm_of_curvature()

```

```

[29]: # Add a parameter for regularization strength
reg_parameters_df = pd.DataFrame(
    dict(
        parameterScale="log10",
        lowerBound=1e-6,
        upperBound=1e6,
        nominalValue=1.0,
        estimate=0,
    ),
    index=pd.Series(["regularization_strength"], name="parameterId"),
)
# Encode regularization term as an additional observable
reg_observables_df = pd.DataFrame(
    dict(
        observableFormula=f"sqrt({regularization}).replace("{}", "^)",
        observableTransformation="lin",
        noiseFormula="1/sqrt(regularization_strength)",
        noiseDistribution="normal",
    ),
    index=pd.Series(["regularization"], name="observableId"),
)
# and corresponding measurement
reg_measurements_df = pd.DataFrame(
    dict(
        observableId="regularization",
        simulationConditionId="condition1",
        measurement=0,
        time=0,
        observableTransformation="lin",
    ),
    index=pd.Series([0]),
)

```

```

[30]: # Add spline formula to SBML model
sbml_doc = libsbml.SBMLReader().readSBML(
    os.path.join("Swameye_PNAS2003", "swameye2003_model.xml")
)
sbml_model = sbml_doc.getModel()
spline.add_to_sbml_model(

```

(continues on next page)

(continued from previous page)

```

    sbml_model, auto_add=True, y_nominal=0.1, y_constant=True
)

```

```

[31]: # Extra parameters associated to the spline
spline_parameters_df = pd.DataFrame(
    dict(
        parameterScale="log",
        lowerBound=0.001,
        upperBound=10,
        nominalValue=0.1,
        estimate=1,
    ),
    index=pd.Series(list(map(str, values_at_nodes)), name="parameterId"),
)

```

```

[32]: # Create PETab problem
petab_problem = petab.Problem(
    sbml_model,
    condition_df=petab.conditions.get_condition_df(
        os.path.join("Swameye_PNAS2003", "swameye2003_conditions.tsv")
    ),
    measurement_df=petab.core.concat_tables(
        [
            os.path.join("Swameye_PNAS2003", "swameye2003_measurements.tsv"),
            reg_measurements_df,
        ],
        petab.measurements.get_measurement_df,
    ).reset_index(drop=True),
    parameter_df=petab.core.concat_tables(
        [
            os.path.join("Swameye_PNAS2003", "swameye2003_parameters.tsv"),
            spline_parameters_df,
            reg_parameters_df,
        ],
        petab.parameters.get_parameter_df,
    ),
    observable_df=petab.core.concat_tables(
        [
            os.path.join("Swameye_PNAS2003", "swameye2003_observables.tsv"),
            reg_observables_df,
        ],
        petab.observables.get_observable_df,
    ),
)

```

```

[33]: # Check whether PETab model is valid
assert not petab.lint_problem(petab_problem)

```

```

[34]: # Save PETab problem to disk
# import shutil
# shutil.rmtree(name, ignore_errors=True)

```

(continues on next page)

(continued from previous page)

```
# os.mkdir(name)
# petab_problem.to_files_generic(prefix_path=name)
```

## Creating the pyPESTO problem

```
[35]: # Problem must be "flattened" to be used with AMICI
petab.core.flatten_timepoint_specific_output_overrides(petab_problem)
```

```
[36]: # Check whether simulation from the PEtab problem works
# import amici.petab_simulate
# simulator = amici.petab_simulate.PetabSimulator(petab_problem)
# simulator.simulate(noise=False)
```

```
[37]: # Import PEtab problem into pyPESTO
pypesto_problem = pypesto.petab.PetabImporter(
    petab_problem, model_name=name
).create_problem()
```

## Maximum Likelihood estimation

We will optimize the problem for different values of the regularization strength  $\lambda$ , then compute the sum of squared normalized residuals for each of the resulting parameter vectors. The one for which such a value is nearest to its expected value of 15 (the number of observations from the input function) will be chosen as the final estimate.

```
[ ]: # Try different regularization strengths
regstrengths = np.asarray([1, 10, 40, 75, 150, 500])
if os.getenv("GITHUB_ACTIONS") is not None:
    regstrengths = np.asarray([75])
regproblems = {}
regresults = {}

for regstrength in regstrengths:
    # Fix parameter in pypesto problem
    name = f"Swameye_PNAS2003_15nodes_FD_reg{regstrength}"
    pypesto_problem.fix_parameters(
        pypesto_problem.x_names.index("regularization_strength"),
        np.log10(
            regstrength
        ), # parameter is specified as log10 scale in PEtab
    )
    regproblem = copy.deepcopy(pypesto_problem)

    # Load existing results if available
    if os.path.exists(f"{name}.h5"):
        regresult = pypesto.store.read_result(f"{name}.h5", problem=regproblem)
    else:
        regresult = None
    # Overwrite
```

(continues on next page)

(continued from previous page)

```

# regresult = None

# Parallel multistart optimization with pyPESTO and FIDES
if n_starts > 0:
    if regresult is None:
        new_ids = [str(i) for i in range(n_starts)]
    else:
        last_id = max(int(i) for i in regresult.optimize_result.id)
        new_ids = [
            str(i) for i in range(last_id + 1, last_id + n_starts + 1)
        ]
    regresult = pypesto.optimize.minimize(
        regproblem,
        n_starts=n_starts,
        ids=new_ids,
        optimizer=pypesto_optimizer,
        engine=pypesto_engine,
        result=regresult,
    )
    regresult.optimize_result.sort()
    if regresult.optimize_result.x[0] is None:
        raise Exception(
            "All multistarts failed (n_starts is probably too small)! If this error_
↳ occurred during CI, just run the workflow again."
        )

# Save results to disk
# pypesto.store.write_result(regresult, f'{name}.h5', overwrite=True)

# Store result
regproblems[regstrength] = regproblem
regresults[regstrength] = regresult

```

```

[39]: # Compute sum of squared normalized residuals
print(f"Target value is {len(df_pEpoR['time'])}")
regstrengths = sorted(regproblems.keys())
stats = []
for regstrength in regstrengths:
    t, pEpoR = simulate_pEpoR(
        N=None,
        problem=regproblems[regstrength],
        result=regresults[regstrength],
    )
    assert np.array_equal(df_pEpoR["time"], t[:-1])
    pEpoR = pEpoR[:-1]
    sigma_pEpoR = 0.0274 + 0.1 * pEpoR
    stat = np.sum(((pEpoR - df_pEpoR["measurement"]) / sigma_pEpoR) ** 2)
    print(f"Regularization strength: {regstrength}. Statistic is {stat}")
    stats.append(stat)
# Select best regularization strength
chosen_regstrength = regstrengths[
    np.abs(np.asarray(stats) - len(df_pEpoR["time"])).argmin()
]

```

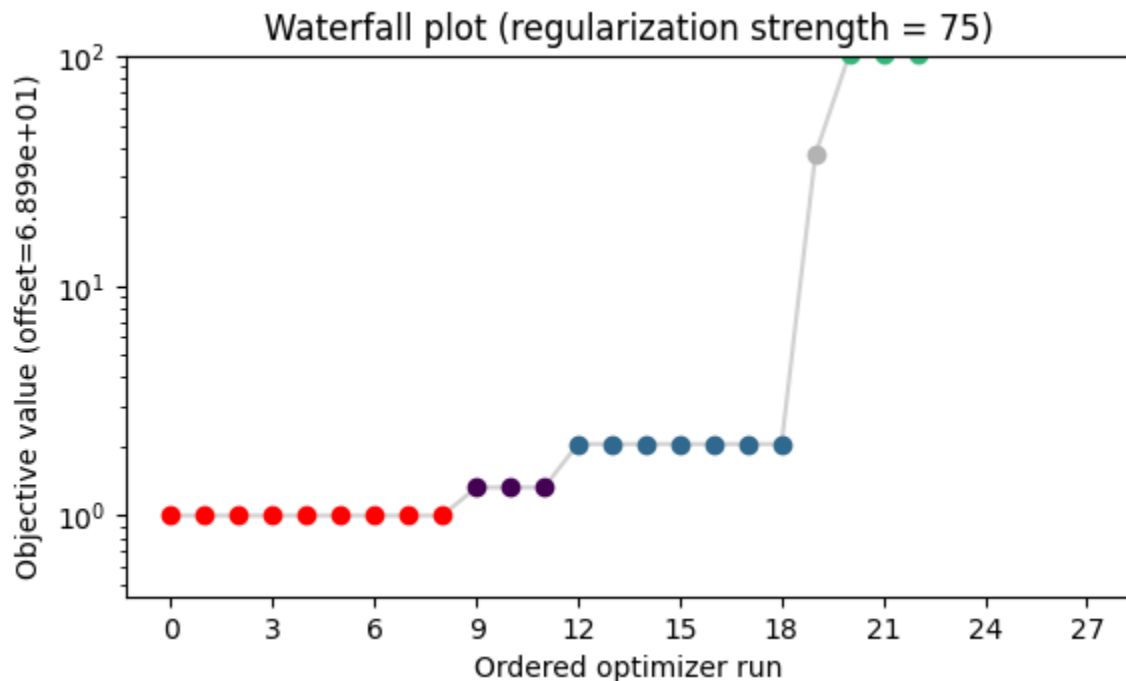
(continues on next page)

(continued from previous page)

]

Target value is 15  
 Regularization strength: 1. Statistic is 6.794369874307712  
 Regularization strength: 10. Statistic is 8.435094498146606  
 Regularization strength: 40. Statistic is 11.83872830962955  
 Regularization strength: 75. Statistic is 15.030926511510327  
 Regularization strength: 150. Statistic is 19.971139477161476  
 Regularization strength: 500. Statistic is 32.44623424533765

```
[40]: # Visualize the results of the multistarts for a chosen regularization strength
ax = pypesto.visualize.waterfall(
    regresults[chosen_regstrength], size=[6.5, 3.5]
)
ax.set_title(
    f"Waterfall plot (regularization strength = {chosen_regstrength})"
)
ax.set_ylim(ax.get_ylim()[0], 100);
```



```
[46]: # Plot ML fit for pEpoR (all regularization strengths)
fig, ax = plt.subplots(figsize=(6.5, 3.5))
for regstrength in sorted(regproblems.keys()):
    t, pEpoR = simulate_pEpoR(
        problem=regproblems[regstrength], result=regresults[regstrength]
    )
    if regstrength == chosen_regstrength:
        kwargs = dict(
            color="black",
            label=f"$\\mathbf{{\\lambda}} = {regstrength}$",
```

(continues on next page)

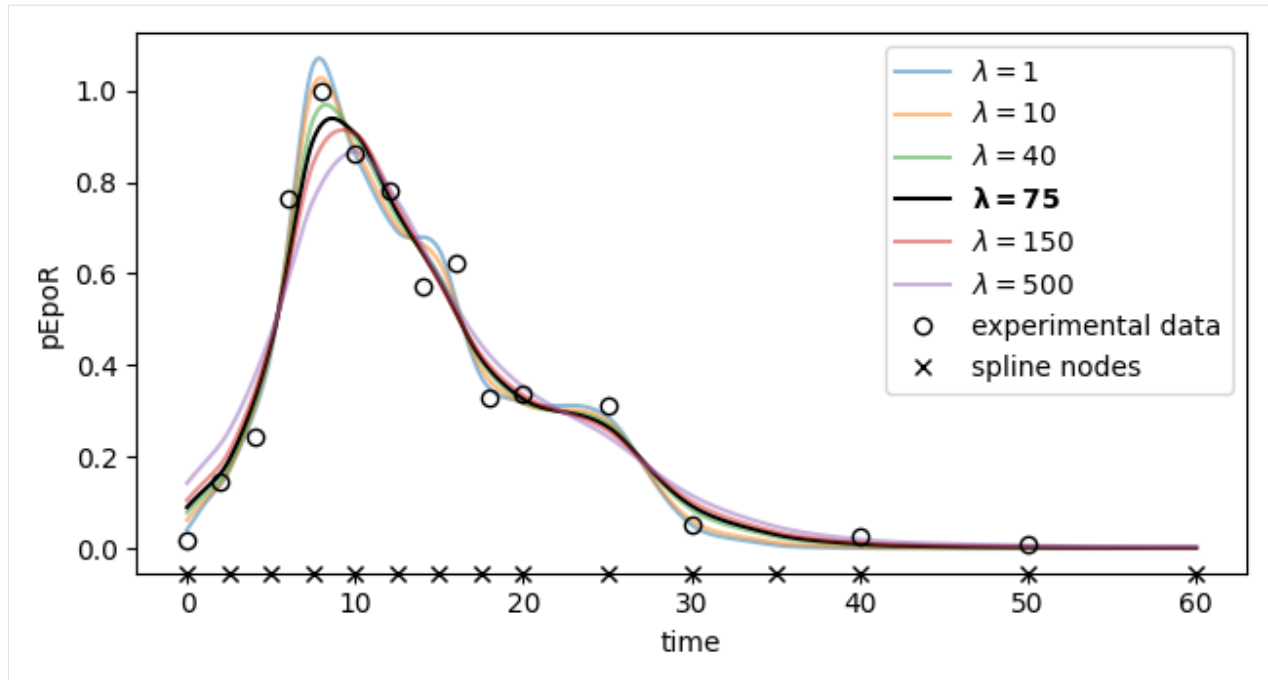


(continued from previous page)

```

        zorder=2,
    )
    else:
        kwargs = dict(label=f"$\\lambda = {regstrength}$", alpha=0.5)
    ax.plot(t, pEpoR, **kwargs)
ax.plot(
    df_pEpoR["time"],
    df_pEpoR["measurement"],
    "o",
    color="black",
    markerfacecolor="none",
    label="experimental data",
)
ylim1 = ax.get_ylim()[0]
ax.plot(
    nodes,
    len(nodes) * [ylim1],
    "x",
    color="black",
    label="spline nodes",
    zorder=10,
    clip_on=False,
)
ax.set_ylim(ylim1, ax.get_ylim()[1])
ax.set_xlabel("time")
ax.set_ylabel("pEpoR")
ax.set_xlim(-3.0, 63.0)
ax.set_ylim(-0.05299052022388704, 1.126290214024833)
ax.legend()
ax.figure.tight_layout()
# ax.set_ylabel("input function")
# print(f"xlim = {ax.get_xlim()}, ylim = {ax.get_ylim()}")
# ax.figure.savefig('fit_15nodes_lambdas.pdf')

```



```
[47]: # Plot ML fit for pSTAT5 (all regularization strengths)
fig, ax = plt.subplots(figsize=(6.5, 3.5))
for regstrength in sorted(regproblems.keys()):
    t, pSTAT5 = simulate_pSTAT5(
        problem=regproblems[regstrength], result=regresults[regstrength]
    )
    if regstrength == chosen_regstrength:
        kwargs = dict(
            color="black",
            label=f"$\\mathbf{{{\\lambda} = {regstrength}}}$",
            zorder=2,
        )
    else:
        kwargs = dict(label=f"$\\mathbf{{{\\lambda} = {regstrength}}}$", alpha=0.5)
    ax.plot(t, pSTAT5, **kwargs)
ax.plot(
    df_pSTAT5["time"],
    df_pSTAT5["measurement"],
    "o",
    color="black",
    markerfacecolor="none",
    label="experimental data",
)
ylim1 = ax.get_ylim()[0]
ax.plot(
    nodes,
    len(nodes) * [ylim1],
    "x",
    color="black",
    label="spline nodes",
)
```

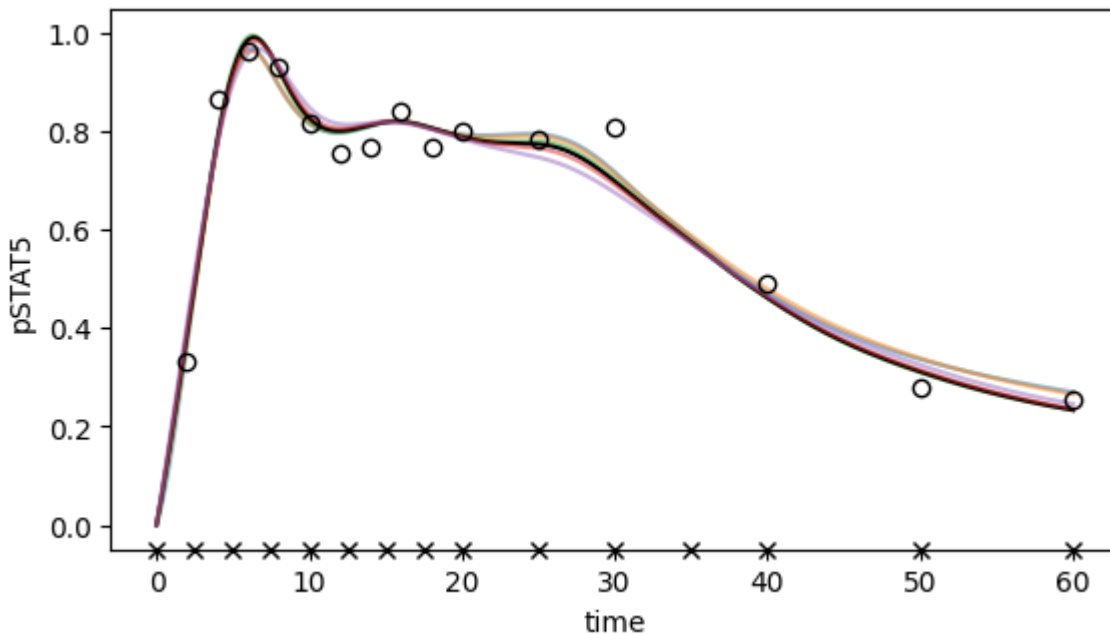
(continues on next page)

(continued from previous page)

```

    zorder=10,
    clip_on=False,
)
ax.set_ylim(ylim1, ax.get_ylim()[1])
ax.set_xlabel("time")
ax.set_ylabel("pSTAT5");
# ax.legend();

```



```

[48]: # Plot ML fit for tSTAT5 (all regularization strengths)
fig, ax = plt.subplots(figsize=(6.5, 3.5))
for regstrength in sorted(regproblems.keys()):
    t, tSTAT5 = simulate_tSTAT5(
        problem=regproblems[regstrength], result=regresults[regstrength]
    )
    if regstrength == chosen_regstrength:
        kwargs = dict(
            color="black",
            label=f"$\\mathbf{{\\lambda}} = {regstrength}$",
            zorder=2,
        )
    else:
        kwargs = dict(label=f"$\\lambda = {regstrength}$", alpha=0.5)
    ax.plot(t, tSTAT5, **kwargs)
ax.plot(
    df_tSTAT5["time"],
    df_tSTAT5["measurement"],
    "o",
    color="black",
    markerfacecolor="none",
    label="experimental data",

```

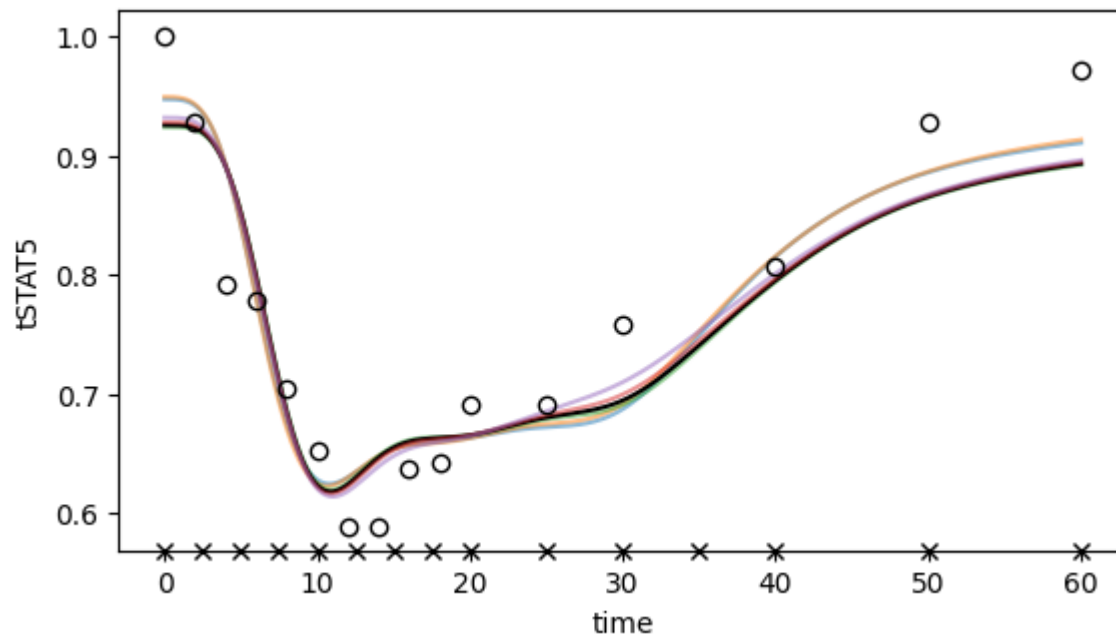
(continues on next page)

(continued from previous page)

```

)
ylim1 = ax.get_ylim()[0]
ax.plot(
    nodes,
    len(nodes) * [ylim1],
    "x",
    color="black",
    label="spline nodes",
    zorder=10,
    clip_on=False,
)
ax.set_ylim(ylim1, ax.get_ylim()[1])
ax.set_xlabel("time")
ax.set_ylabel("tSTAT5");
# ax.legend();

```



```

[49]: # Plot ML fit for pEpoR (single regularization strength with noise model)
fig, ax = plt.subplots(figsize=(6.5, 3.5))
t, pEpoR = simulate_pEpoR(
    problem=regproblems[chosen_regstrength],
    result=regresults[chosen_regstrength],
)
sigma_pEpoR = 0.0274 + 0.1 * pEpoR
ax.fill_between(
    t,
    pEpoR - 2 * sigma_pEpoR,
    pEpoR + 2 * sigma_pEpoR,
    color="black",
    alpha=0.10,
    interpolate=True,
)

```

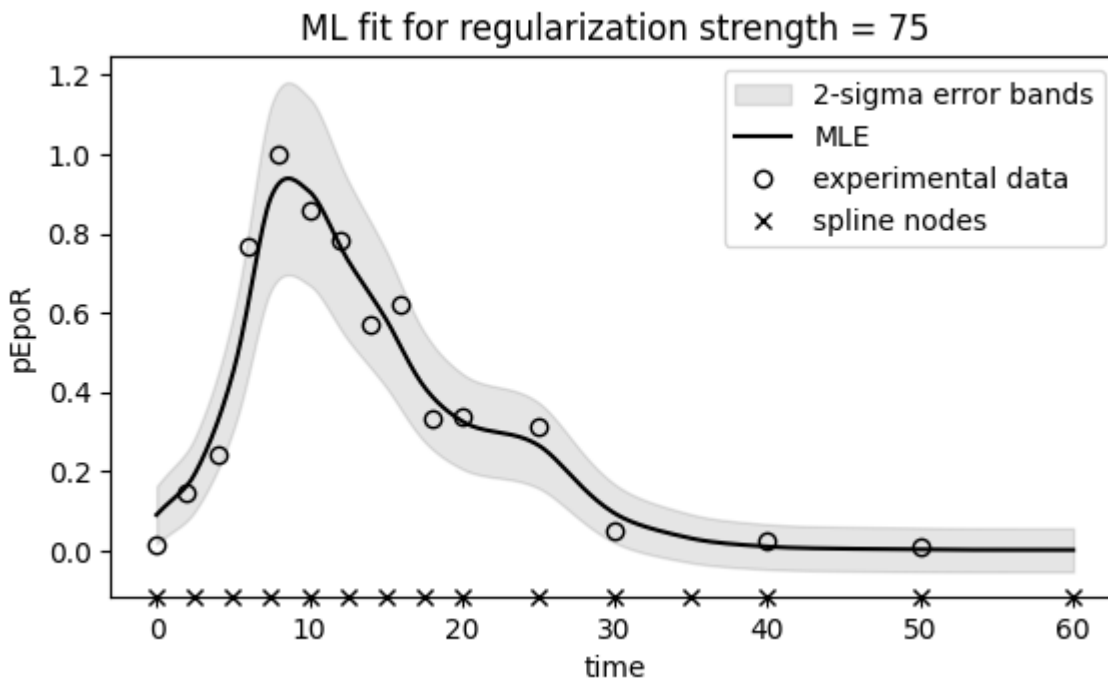
(continues on next page)

(continued from previous page)

```

    label="2-sigma error bands",
)
ax.plot(t, pEpoR, color="black", label="MLE")
ax.plot(
    df_pEpoR["time"],
    df_pEpoR["measurement"],
    "o",
    color="black",
    markerfacecolor="none",
    label="experimental data",
)
ylim1 = ax.get_ylim()[0]
ax.plot(
    nodes,
    len(nodes) * [ylim1],
    "x",
    color="black",
    label="spline nodes",
    zorder=10,
    clip_on=False,
)
ax.set_ylim(ylim1, ax.get_ylim()[1])
ax.set_xlabel("time")
ax.set_ylabel("pEpoR")
ax.set_title(f"ML fit for regularization strength = {chosen_regstrength}")
ax.legend();

```



```

[50]: # Store results for later
all_results["15 nodes, FD"] = (

```

(continues on next page)

(continued from previous page)

```

    regproblems[chosen_regstrength],
    regresults[chosen_regstrength],
)

```

### Spline approximation with few nodes, optimizing derivatives explicitly

An alternative way to achieve higher expressivity, while not increasing the number of nodes, is to optimize the derivatives of the spline at the nodes instead of computing them by finite differencing. The risk of overfitting is still present, so we will include regularization as in the above example.

### Creating the PETab model

```

[51]: # Problem name
name = "Swameye_PNAS2003_5nodes"

```

We now need to create additional parameters for the spline derivatives too.

```

[52]: # Create spline for pEpoR
nodes = [0, 5, 10, 20, 60]
values_at_nodes = [
    sp.Symbol(f"pEpoR_t{str(t).replace('.', '_dot_')}") for t in nodes
]
derivatives_at_nodes = [
    sp.Symbol(f"derivative_pEpoR_t{str(t).replace('.', '_dot_')}")
    for t in nodes[:-1]
]
spline = amici.splines.CubicHermiteSpline(
    sbml_id="pEpoR",
    evaluate_at=amici.sbml_utils.amici_time_symbol,
    nodes=nodes,
    values_at_nodes=values_at_nodes,
    derivatives_at_nodes=derivatives_at_nodes
    + [0], # last value is zero because steady state is reached
    extrapolate=(None, "constant"),
    bc="auto",
    logarithmic_parametrization=True,
)

```

```

[53]: # Compute L2 norm of the curvature of pEpoR
regularization = spline.squared_L2_norm_of_curvature()

```

```

[54]: # Add a parameter for regularization strength
reg_parameters_df = pd.DataFrame(
    dict(
        parameterScale="log10",
        lowerBound=1e-6,
        upperBound=1e6,
        nominalValue=1.0,
        estimate=0,
    )
)

```

(continues on next page)

(continued from previous page)

```

    ),
    index=pd.Series(["regularization_strength"], name="parameterId"),
)
# Encode regularization term as an additional observable
reg_observables_df = pd.DataFrame(
    dict(
        observableFormula=f"sqrt({regularization}).replace("**", "^)",
        observableTransformation="lin",
        noiseFormula="1/sqrt(regularization_strength)",
        noiseDistribution="normal",
    ),
    index=pd.Series(["regularization"], name="observableId"),
)
# and corresponding measurement
reg_measurements_df = pd.DataFrame(
    dict(
        observableId="regularization",
        simulationConditionId="condition1",
        measurement=0,
        time=0,
        observableTransformation="lin",
    ),
    index=pd.Series([0]),
)

```

```

[55]: # Add spline formula to SBML model
sbml_doc = libsbml.SBMLReader().readSBML(
    os.path.join("Swameye_PNAS2003", "swameye2003_model.xml")
)
sbml_model = sbml_doc.getModel()
spline.add_to_sbml_model(
    sbml_model, auto_add=True, y_nominal=0.1, y_constant=True
)

```

```

[56]: # Derivative parameters must be added separately
for p in derivatives_at_nodes:
    amici.sbml_utils.add_parameter(sbml_model, p, value=0.0, constant=True)

```

```

[57]: # Extra parameters associated to the spline
spline_parameters_df1 = pd.DataFrame(
    dict(
        parameterScale="log",
        lowerBound=0.001,
        upperBound=10,
        nominalValue=0.1,
        estimate=1,
    ),
    index=pd.Series(list(map(str, values_at_nodes)), name="parameterId"),
)
spline_parameters_df2 = pd.DataFrame(
    dict(

```

(continues on next page)

(continued from previous page)

```

        parameterScale="lin",
        lowerBound=-0.666,
        upperBound=0.666,
        nominalValue=0.0,
        estimate=1,
    ),
    index=pd.Series(list(map(str, derivatives_at_nodes)), name="parameterId"),
)

```

```

[58]: # Create PEtab problem
petab_problem = petab.Problem(
    sbml_model,
    condition_df=petab.conditions.get_condition_df(
        os.path.join("Swameye_PNAS2003", "swameye2003_conditions.tsv")
    ),
    measurement_df=petab.core.concat_tables(
        [
            os.path.join("Swameye_PNAS2003", "swameye2003_measurements.tsv"),
            reg_measurements_df,
        ],
        petab.measurements.get_measurement_df,
    ).reset_index(drop=True),
    parameter_df=petab.core.concat_tables(
        [
            os.path.join("Swameye_PNAS2003", "swameye2003_parameters.tsv"),
            spline_parameters_df1,
            spline_parameters_df2,
            reg_parameters_df,
        ],
        petab.parameters.get_parameter_df,
    ),
    observable_df=petab.core.concat_tables(
        [
            os.path.join("Swameye_PNAS2003", "swameye2003_observables.tsv"),
            reg_observables_df,
        ],
        petab.observables.get_observable_df,
    ),
)

```

```

[59]: # Check whether PEtab model is valid
assert not petab.lint_problem(petab_problem)

```

```

[60]: # Save PEtab problem to disk
# import shutil
# shutil.rmtree(name, ignore_errors=True)
# os.mkdir(name)
# petab_problem.to_files_generic(prefix_path=name)

```



## Creating the pyPESTO problem

```
[61]: # Problem must be "flattened" to be used with AMICI
petab.core.flatten_timepoint_specific_output_overrides(petab_problem)

[62]: # Check whether simulation from the PEtab problem works
# import amici.petab_simulate
# simulator = amici.petab_simulate.PetabSimulator(petab_problem)
# simulator.simulate(noise=False)

[63]: # Import PEtab problem into pyPESTO
pypesto_problem = pypesto.petab.PetabImporter(
    petab_problem, model_name=name
).create_problem()
```

## Maximum Likelihood estimation

```
[ ]: # Try different regularization strengths
regstrengths = np.asarray([1, 175, 500, 1000])
if os.getenv("GITHUB_ACTIONS") is not None:
    regstrengths = np.asarray([175])
regproblems = {}
regresults = {}

for regstrength in regstrengths:
    # Fix parameter in pypesto problem
    name = f"Swameye_PNAS2003_5nodes_reg{regstrength}"
    pypesto_problem.fix_parameters(
        pypesto_problem.x_names.index("regularization_strength"),
        np.log10(
            regstrength
        ), # parameter is specified as log10 scale in PEtab
    )
    regproblem = copy.deepcopy(pypesto_problem)

    # Load existing results if available
    if os.path.exists(f"{name}.h5"):
        regresult = pypesto.store.read_result(f"{name}.h5", problem=regproblem)
    else:
        regresult = None
    # Overwrite
    # regresult = None

    # Parallel multistart optimization with pyPESTO and FIDES
    if n_starts > 0:
        if regresult is None:
            new_ids = [str(i) for i in range(n_starts)]
        else:
            last_id = max(int(i) for i in regresult.optimize_result.id)
            new_ids = [
```

(continues on next page)

(continued from previous page)

```

        str(i) for i in range(last_id + 1, last_id + n_starts + 1)
    ]
    regresult = pypesto.optimize.minimize(
        regproblem,
        n_starts=n_starts,
        ids=new_ids,
        optimizer=pypesto_optimizer,
        engine=pypesto_engine,
        result=regresult,
    )
    regresult.optimize_result.sort()
    if regresult.optimize_result.x[0] is None:
        raise Exception(
            "All multistarts failed (n_starts is probably too small)! If this error
            occurred during CI, just run the workflow again."
        )

    # Save results to disk
    # pypesto.store.write_result(regresult, f'{name}.h5', overwrite=True)

    # Store result
    regproblems[regstrength] = regproblem
    regresults[regstrength] = regresult

```

```

[65]: # Compute sum of squared normalized residuals
print(f"Target value is {len(df_pEpoR['time'])}")
regstrengths = sorted(regproblems.keys())
stats = []
for regstrength in regstrengths:
    t, pEpoR = simulate_pEpoR(
        N=None,
        problem=regproblems[regstrength],
        result=regresults[regstrength],
    )
    assert np.array_equal(df_pEpoR["time"], t[:-1])
    pEpoR = pEpoR[:-1]
    sigma_pEpoR = 0.0274 + 0.1 * pEpoR
    stat = np.sum(((pEpoR - df_pEpoR["measurement"]) / sigma_pEpoR) ** 2)
    print(f"Regularization strength: {regstrength}. Statistic is {stat}")
    stats.append(stat)
# Select best regularization strength
chosen_regstrength = regstrengths[
    np.abs(np.asarray(stats) - len(df_pEpoR["time"])).argmin()
]

```

```

Target value is 15
Regularization strength: 1. Statistic is 9.638207938045252
Regularization strength: 175. Statistic is 15.115255701660317
Regularization strength: 500. Statistic is 19.156287450444093
Regularization strength: 1000. Statistic is 25.09224919998158

```

```

[66]: # Visualize the results of the multistarts for a chosen regularization strength

```

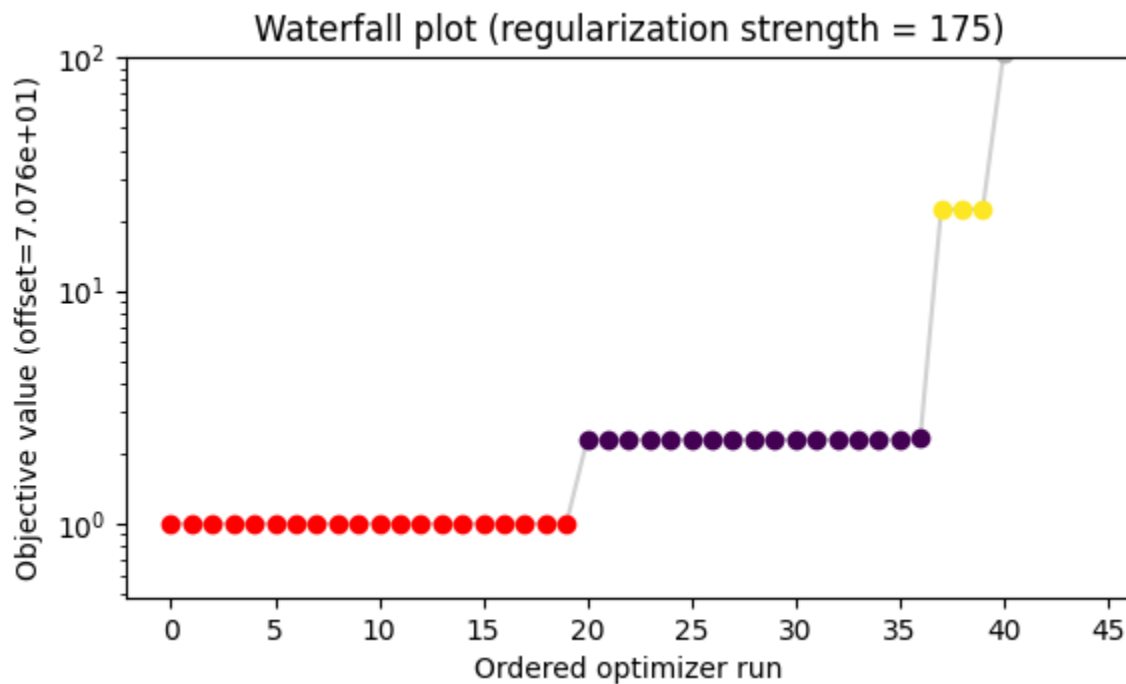
(continues on next page)

(continued from previous page)

```

ax = pypesto.visualize.waterfall(
    regresults[chosen_regstrength], size=[6.5, 3.5]
)
ax.set_title(
    f"Waterfall plot (regularization strength = {chosen_regstrength})"
)
ax.set_ylim(ax.get_ylim()[0], 100);

```



```

[76]: # Plot ML fit for pEpoR (all regularization strengths)
fig, ax = plt.subplots(figsize=(6.5, 3.5))
for regstrength in sorted(regproblems.keys()):
    t, pEpoR = simulate_pEpoR(
        problem=regproblems[regstrength], result=regresults[regstrength]
    )
    if regstrength == chosen_regstrength:
        kwargs = dict(
            color="black",
            label=f"$\\mathbf{{\\lambda}} = {regstrength}$",
            zorder=2,
        )
    else:
        kwargs = dict(label=f"$\\lambda = {regstrength}$", alpha=0.5)
    ax.plot(t, pEpoR, **kwargs)
ax.plot(
    df_pEpoR["time"],
    df_pEpoR["measurement"],
    "o",
    color="black",
    markerfacecolor="none",

```

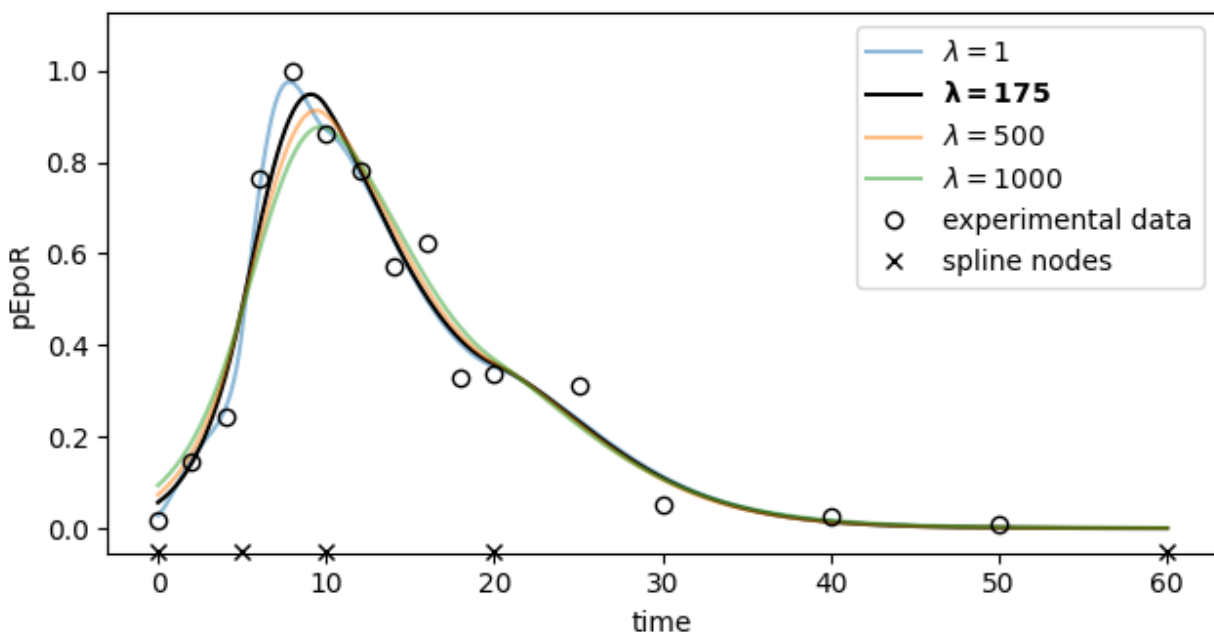
(continues on next page)

(continued from previous page)

```

    label="experimental data",
)
ylim1 = ax.get_ylim()[0]
ax.plot(
    nodes,
    len(nodes) * [ylim1],
    "x",
    color="black",
    label="spline nodes",
    zorder=10,
    clip_on=False,
)
ax.set_ylim(ylim1, ax.get_ylim()[1])
ax.set_xlabel("time")
ax.set_ylabel("pEpoR")
ax.set_xlim(-3.0, 63.0)
ax.set_ylim(-0.05299052022388704, 1.126290214024833)
ax.legend()
ax.figure.tight_layout()
# ax.set_ylabel("input function")
# ax.figure.savefig('fit_5nodes_lambdas.pdf')

```



```

[68]: # Plot ML fit for pSTAT5 (all regularization strengths)
fig, ax = plt.subplots(figsize=(6.5, 3.5))
for regstrength in sorted(regproblems.keys()):
    t, pSTAT5 = simulate_pSTAT5(
        problem=regproblems[regstrength], result=regresults[regstrength]
    )
    if regstrength == chosen_regstrength:
        kwargs = dict(

```

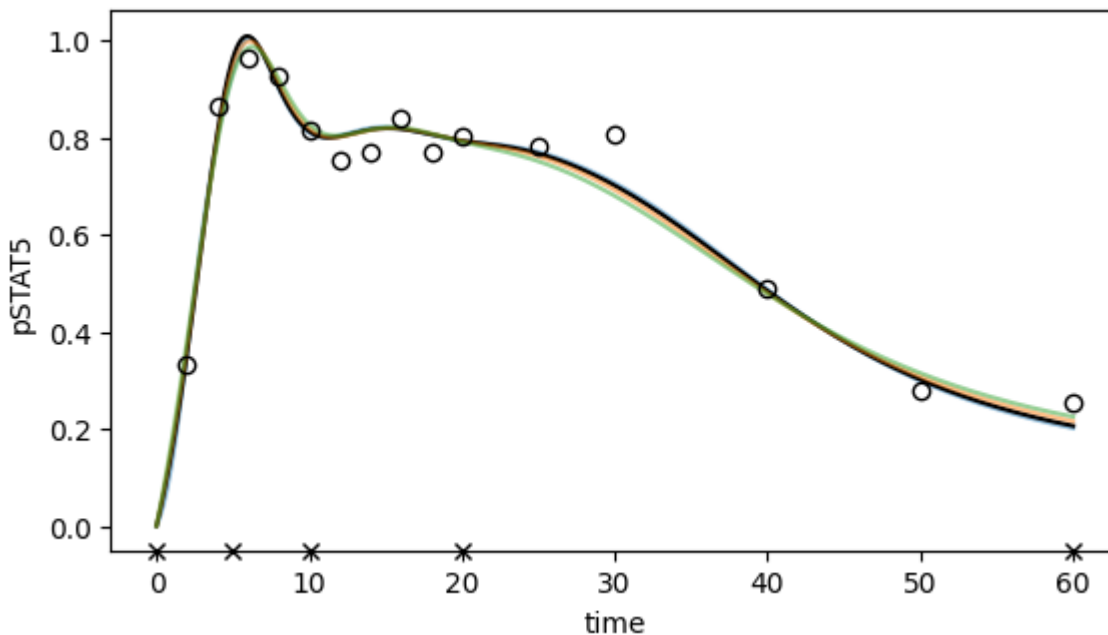
(continues on next page)

(continued from previous page)

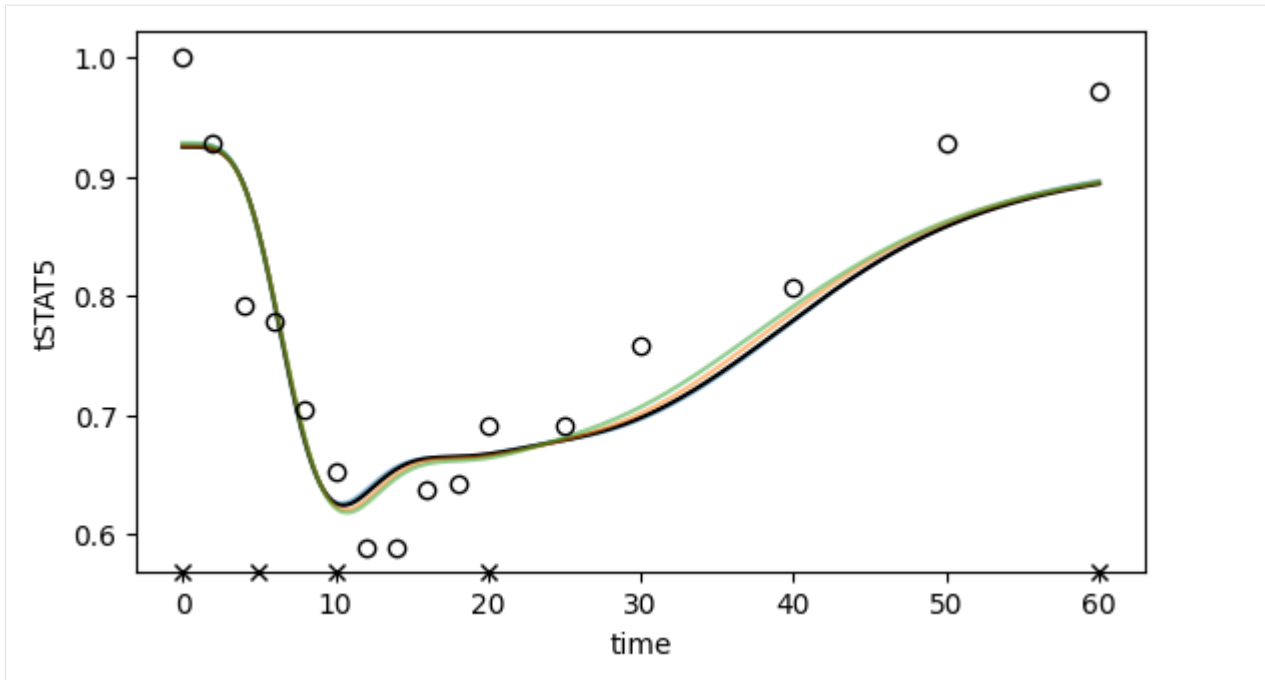
```

        color="black",
        label=f"$\\mathbf{{\\lambda}} = {regstrength}$",
        zorder=2,
    )
    else:
        kwargs = dict(label=f"$\\mathbf{{\\lambda}} = {regstrength}$", alpha=0.5)
    ax.plot(t, pSTAT5, **kwargs)
ax.plot(
    df_pSTAT5["time"],
    df_pSTAT5["measurement"],
    "o",
    color="black",
    markerfacecolor="none",
    label="experimental data",
)
ylim1 = ax.get_ylim()[0]
ax.plot(
    nodes,
    len(nodes) * [ylim1],
    "x",
    color="black",
    label="spline nodes",
    zorder=10,
    clip_on=False,
)
ax.set_ylim(ylim1, ax.get_ylim()[1])
ax.set_xlabel("time")
ax.set_ylabel("pSTAT5");
# ax.legend();

```



```
[69]: # Plot ML fit for tSTAT5 (all regularization strengths)
fig, ax = plt.subplots(figsize=(6.5, 3.5))
for regstrength in sorted(regproblems.keys()):
    t, tSTAT5 = simulate_tSTAT5(
        problem=regproblems[regstrength], result=regresults[regstrength]
    )
    if regstrength == chosen_regstrength:
        kwargs = dict(
            color="black",
            label=f"$\\mathbf{{{\\lambda} = {regstrength}}}$",
            zorder=2,
        )
    else:
        kwargs = dict(label=f"$\\lambda = {regstrength}$", alpha=0.5)
    ax.plot(t, tSTAT5, **kwargs)
ax.plot(
    df_tSTAT5["time"],
    df_tSTAT5["measurement"],
    "o",
    color="black",
    markerfacecolor="none",
    label="experimental data",
)
ylim1 = ax.get_ylim()[0]
ax.plot(
    nodes,
    len(nodes) * [ylim1],
    "x",
    color="black",
    label="spline nodes",
    zorder=10,
    clip_on=False,
)
ax.set_ylim(ylim1, ax.get_ylim()[1])
ax.set_xlabel("time")
ax.set_ylabel("tSTAT5");
# ax.legend();
```



```
[70]: # Plot ML fit for pEpoR (single regularization strength with noise model)
fig, ax = plt.subplots(figsize=(6.5, 3.5))
t, pEpoR = simulate_pEpoR(
    problem=regproblems[chosen_regstrength],
    result=regresults[chosen_regstrength],
)
sigma_pEpoR = 0.0274 + 0.1 * pEpoR
ax.fill_between(
    t,
    pEpoR - 2 * sigma_pEpoR,
    pEpoR + 2 * sigma_pEpoR,
    color="black",
    alpha=0.10,
    interpolate=True,
    label="2-sigma error bands",
)
ax.plot(t, pEpoR, color="black", label="MLE")
ax.plot(
    df_pEpoR["time"],
    df_pEpoR["measurement"],
    "o",
    color="black",
    markerfacecolor="none",
    label="experimental data",
)
ylim1 = ax.get_ylim()[0]
ax.plot(
    nodes,
    len(nodes) * [ylim1],
    "x",
```

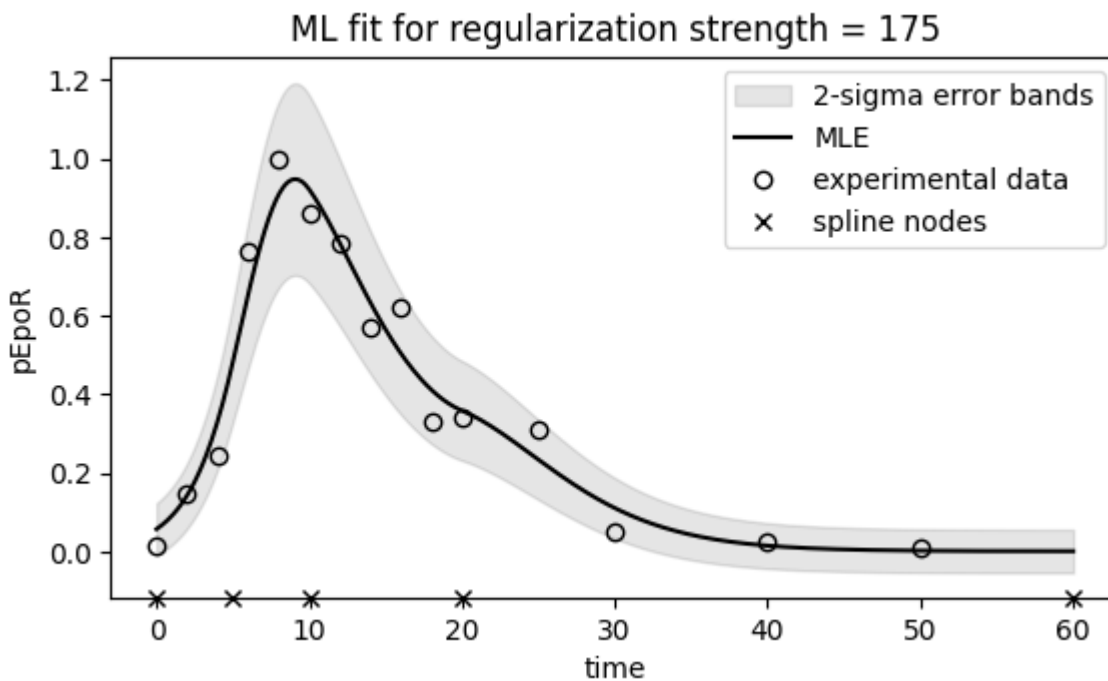
(continues on next page)

(continued from previous page)

```

    color="black",
    label="spline nodes",
    zorder=10,
    clip_on=False,
)
ax.set_ylim(ylim1, ax.get_ylim()[1])
ax.set_xlabel("time")
ax.set_ylabel("pEpoR")
ax.set_title(f"ML fit for regularization strength = {chosen_regstrength}")
ax.legend();

```



```

[71]: # Store results for later
all_results["5 nodes"] = (
    regproblems[chosen_regstrength],
    regresults[chosen_regstrength],
)

```

### Comparing the three approaches

```

[72]: # Plot ML fit for pEpoR
fig, ax = plt.subplots(figsize=(6.5, 3.5))
for label, (problem, result) in all_results.items():
    t, pEpoR = simulate_pEpoR(problem=problem, result=result)
    ax.plot(t, pEpoR, label=label)
ax.plot(
    df_pEpoR["time"],
    df_pEpoR["measurement"],

```

(continues on next page)

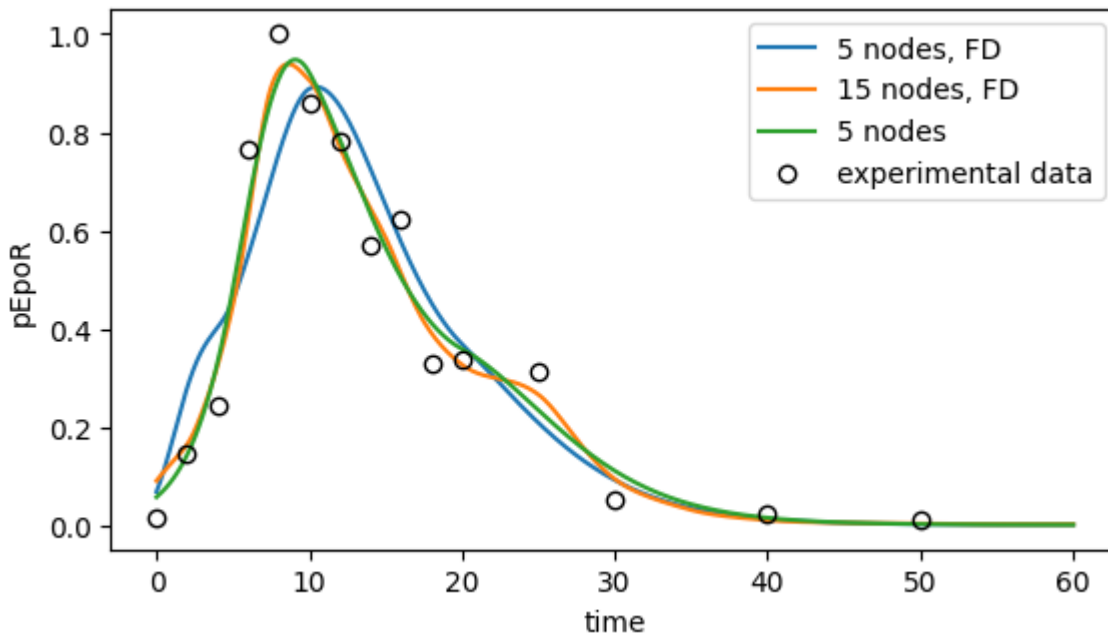


(continued from previous page)

```

    "o",
    color="black",
    markerfacecolor="none",
    label="experimental data",
)
ax.set_xlabel("time")
ax.set_ylabel("pEpoR")
ax.legend();

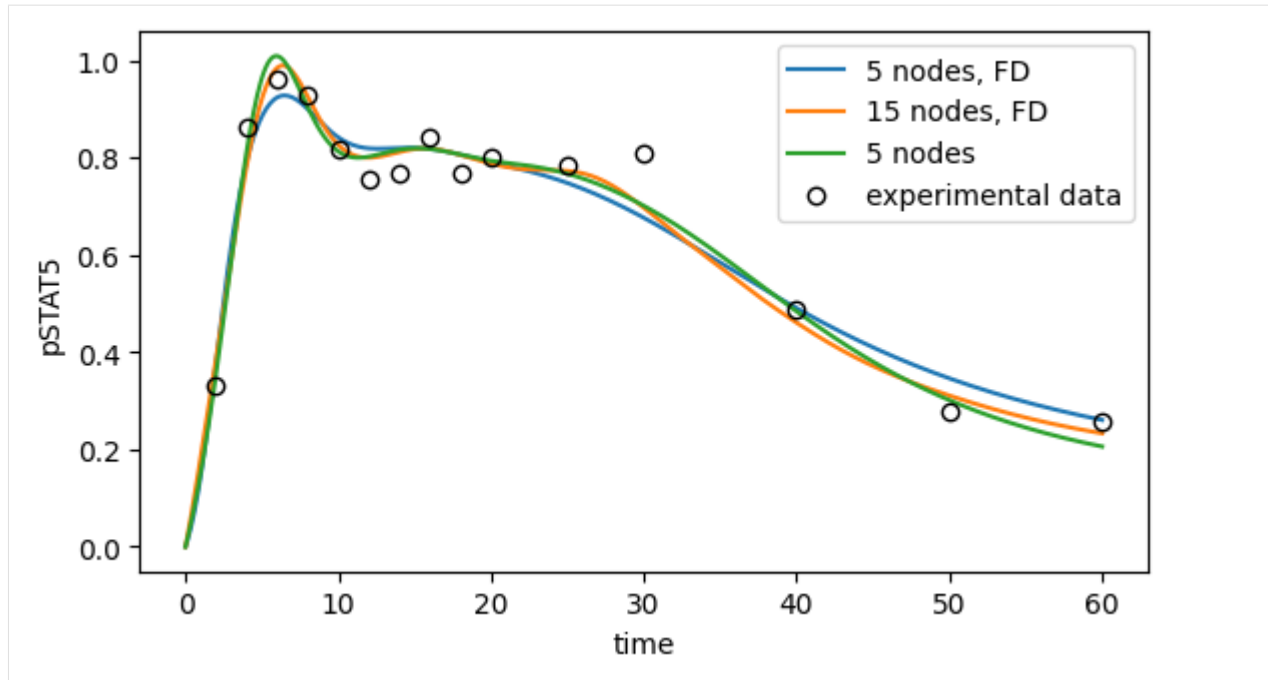
```



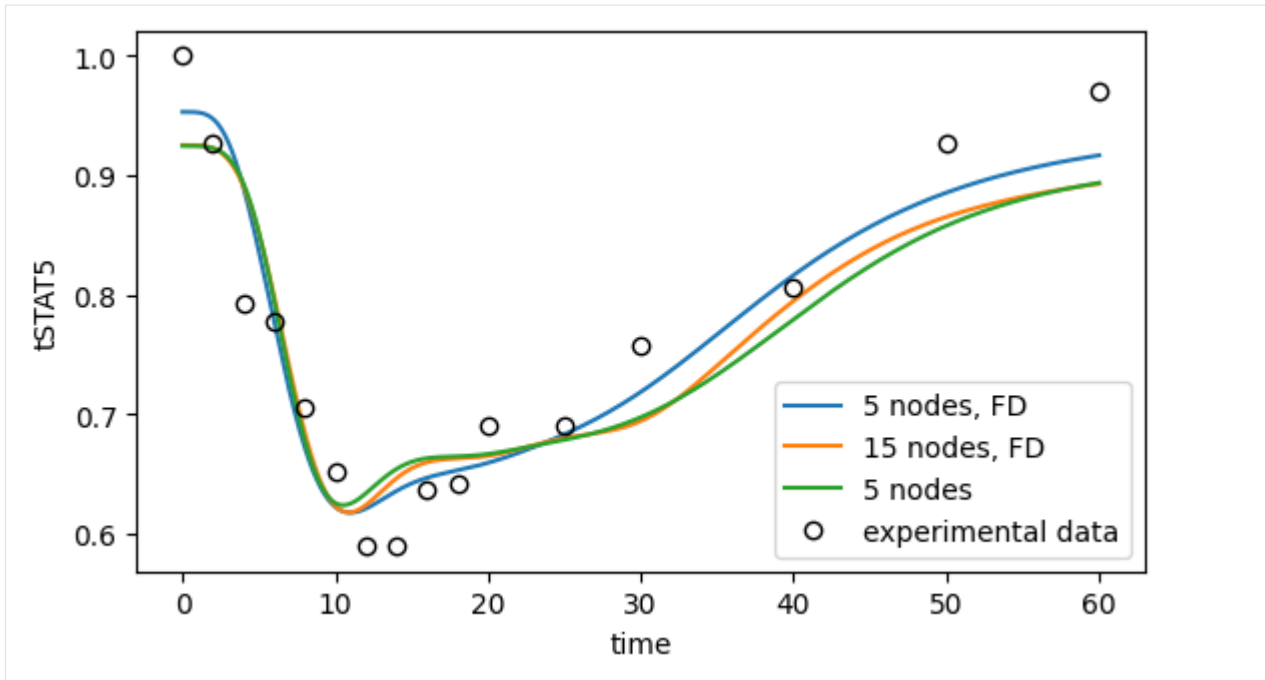
```

[73]: # Plot ML fit for pSTAT5
fig, ax = plt.subplots(figsize=(6.5, 3.5))
for label, (problem, result) in all_results.items():
    t, pSTAT5 = simulate_pSTAT5(problem=problem, result=result)
    ax.plot(t, pSTAT5, label=label)
ax.plot(
    df_pSTAT5["time"],
    df_pSTAT5["measurement"],
    "o",
    color="black",
    markerfacecolor="none",
    label="experimental data",
)
ax.set_xlabel("time")
ax.set_ylabel("pSTAT5")
ax.legend();

```



```
[74]: # Plot ML fit for tSTAT5
fig, ax = plt.subplots(figsize=(6.5, 3.5))
for label, (problem, result) in all_results.items():
    t, tSTAT5 = simulate_tSTAT5(problem=problem, result=result)
    ax.plot(t, tSTAT5, label=label)
ax.plot(
    df_tSTAT5["time"],
    df_tSTAT5["measurement"],
    "o",
    color="black",
    markerfacecolor="none",
    label="experimental data",
)
ax.set_xlabel("time")
ax.set_ylabel("tSTAT5")
ax.legend();
```



```
[75]: # Compare parameter values
for label, (problem, result) in all_results.items():
    print(f"\n### {label}")
    x = result.optimize_result.x[0]
    if len(x) == len(problem.x_free_indices):
        names = problem.x_names[problem.x_free_indices]
    else:
        names = problem.x_names
    for name, value in zip(names, x):
        print(f"{name} = {value}")
```

```
### 5 nodes, FD
k1 = -0.012344171128634264
k2 = -1.11975626735931
k3 = 5.999999816644789
k4 = 0.22576351403212522
scale_tSTAT5 = -0.020792663448966672
scale_pSTAT5 = 0.1422550065768319
sigma_pEpoR_abs = -1.562249437179612
sigma_pEpoR_rel = -1.0
pEpoR_t0 = -2.6870875267006804
pEpoR_t5 = -0.7797622417853871
pEpoR_t10 = -0.11820562755751975
pEpoR_t20 = -0.9974218537437654
pEpoR_t60 = -6.90775527898212
```

```
### 15 nodes, FD
k1 = 0.1543170078851364
k2 = -1.0042579083153138
k3 = -0.17925294344845363
```

(continues on next page)

(continued from previous page)

```
k4 = 0.31486258696137254
scale_tSTAT5 = -0.03364700359730668
scale_pSTAT5 = 0.11013784140762342
sigma_pEpoR_abs = -1.562249437179612
sigma_pEpoR_rel = -1.0
pEpoR_t0 = -2.40456191981547
pEpoR_t2_dot_5 = -1.6438670641678346
pEpoR_t5_dot_0 = -0.80437214623219
pEpoR_t7_dot_5 = -0.1144993579909219
pEpoR_t10_dot_0 = -0.09849380649928209
pEpoR_t12_dot_5 = -0.30861764847405077
pEpoR_t15_dot_0 = -0.535565172217061
pEpoR_t17_dot_5 = -0.8808659628360864
pEpoR_t20 = -1.1184724117332843
pEpoR_t25 = -1.3245209689075161
pEpoR_t30 = -2.3651756746835053
pEpoR_t35 = -3.4734027477458524
pEpoR_t40 = -4.578132101040909
pEpoR_t50 = -5.7968417139258435
pEpoR_t60 = -6.268875988124801
regularization_strength = 1.8750612633917

### 5 nodes
k1 = 0.2486924371230916
k2 = -0.9010429810987043
k3 = -0.3408591074551208
k4 = 0.3594353532480489
scale_tSTAT5 = -0.03395751814386045
scale_pSTAT5 = 0.1008121903144357
sigma_pEpoR_abs = -1.562249437179612
sigma_pEpoR_rel = -1.0
pEpoR_t0 = -2.8601663957890175
pEpoR_t5 = -0.7275787612811422
pEpoR_t10 = -0.08172482568007049
pEpoR_t20 = -1.02532663950965
pEpoR_t60 = -6.907755278982137
derivative_pEpoR_t0 = 0.026587630472163528
derivative_pEpoR_t5 = 0.17154606724507934
derivative_pEpoR_t10 = -0.05503215878900286
derivative_pEpoR_t20 = -0.016352876798592663
regularization_strength = 2.2430380486862944
```

## Bibliography

Schelker, M. et al. (2012). “Comprehensive estimation of input signals and dynamics in biochemical reaction networks”. In: Bioinformatics 28.18, pp. i529–i534. doi: [10.1093/bioinformatics/bts393](https://doi.org/10.1093/bioinformatics/bts393).

Swameye, I. et al. (2003). “Identification of nucleocytoplasmic cycling as a remote sensor in cellular signaling by databased modeling”. In: Proceedings of the National Academy of Sciences 100.3, pp. 1028–1033. doi: [10.1073/pnas.0237333100](https://doi.org/10.1073/pnas.0237333100).

## 10.3 Using AMICI’s Python interface

In the following we will give a detailed overview how to specify models in Python and how to call the generated simulation files.

### 10.3.1 Model definition

This document provides an overview of different interfaces to import models in AMICI. Further examples are available in the AMICI repository in the [python/examples](#) directory.

#### SBML import

AMICI can import *SBML* models via the `amici.sbml_import.SbmlImporter()` class.

#### Status of SBML support in Python-AMICI

Python-AMICI currently **passes 1247 out of the 1821 (~68%) test cases** from the semantic [SBML Test Suite](#) (current status).

The following SBML test suite tags are currently supported (i.e., at least one test case with the respective test passes; [tag descriptions](#)):

##### Component tags:

- AlgebraicRule
- AssignmentRule
- comp
- Compartment
- CSymbolAvogadro
- CSymbolRateOf
- CSymbolTime
- Deletion
- EventNoDelay
- ExternalModelDefinition
- FunctionDefinition
- InitialAssignment
- ModelDefinition

- Parameter
- Port
- RateRule
- Reaction
- ReplacedBy
- ReplacedElement
- SBaseRef
- Species
- Submodel

**Test tags:**

- 0D-Compartment
- Amount
- AssignedConstantStoichiometry
- AssignedVariableStoichiometry
- BoolNumericSwap
- BoundaryCondition
- comp
- Concentration
- ConstantSpecies
- ConversionFactor
- ConversionFactors
- DefaultValue
- EventT0Firing
- ExtentConversionFactor
- HasOnlySubstanceUnits
- InitialValueReassigned
- L3v2MathML
- LocalParameters
- MultiCompartment
- NoMathML
- NonConstantCompartment
- NonConstantParameter
- NonUnityCompartment
- NonUnityStoichiometry
- ReversibleReaction
- SpeciesReferenceInMath

- SubmodelOutput
- TimeConversionFactor
- UncommonMathML
- VolumeConcentrationRates

Additional support may be added in the future. However, the following features are unlikely to be supported:

- *factorial()*, *ceil()*, *floor()*, due to incompatibility with symbolic sensitivity computations
- *delay()* due to missing *SUNDIALS* solver support
- events with delays, events with non-persistent triggers

## Tutorials

A basic tutorial on how to import and simulate SBML models is available in the *Getting Started notebook*, while a more detailed example including customized import and sensitivity computation is available in the *Example Steadystate notebook*.

## PySB import

AMICI can import *PySB* models via `amici.pysb_import.pysb2amici()`.

## BNGL import

AMICI can import *BNGL* models via `amici.bngl_import.bngl2amici()`.

## PETab import

AMICI can import *PETab*-based model definitions and run simulations for the specified simulations conditions. For usage, see *python/examples/example\_petab/petab.ipynb*.

## Importing plain ODEs

The AMICI Python interface does not currently support direct import of ODEs. However, it is straightforward to encode them as RateRules in an SBML model. The most convenient options to do that are maybe *Antimony* and *yaml2sbml*.

An example using Antimony to specify the Lotka-Volterra equations is shown below:

```
ant_model = """
model lotka_volterra
  # see https://en.wikipedia.org/wiki/Lotka%E2%80%93Volterra_equations

  # initial conditions
  prey_density = 10;
  predator_density = 10;

  # parameters
  prey_growth_rate = 1.1;
  predator_effect_on_pre = 0.4;
```

(continues on next page)

(continued from previous page)

```

    predator_death_rate = 0.4;
    prey_effect_on_predator = 0.1;

    # dx/dt
    prey_density' = prey_growth_rate * prey_density - predator_effect_on_prey * prey_
    ↪density * predator_density;
    predator_density' = prey_effect_on_predator * prey_density * predator_density -
    ↪predator_death_rate * predator_density;
end
"""
module_name = "test_antimony_example_lv"
from amici.antimony_import import Antimony2Amici
Antimony2Amici(
    ant_model,
    model_name=module_name,
    output_dir=module_name,
)
model_module = amici.import_model_module(
    module_name=module_name, module_path=outdir
)
amici_model = model_module.getModel()
amici_model.setTimepoints(np.linspace(0, 100, 200))
amici_solver = amici_model.getSolver()
rdata = amici.runAmiciSimulation(amici_model, amici_solver)

from amici.plotting import plot_state_trajectories
plot_state_trajectories(rdata, model=amici_model)

```

The `yaml2sbml` package creates SBML models from a YAML-based specification of an ODE model. Various examples are [provided](#). Besides the SBML model, `yaml2sbml` can also create [PEtab](#) files.

## SED-ML import

We also plan to implement support for the [Simulation Experiment Description Markup Language \(SED-ML\)](#).

### 10.3.2 Environment variables affecting model import

In addition to the environment variables listed [here](#), the following environment variables control various behaviours during model import and compilation:



Table 1: Environment variables affecting model import

Variable	Purpose	Example
AMICI_EXTRACT_CSE	Extract common subexpressions. May significantly reduce file size and compile time for large models, but makes the generated code less readable. Disabled by default.	AMICI_EXTRACT_CSE=1
AMICI_IMPORT_NPROCS	Number of processes to be used for model import. Defaults to 1. Speeds up import of large models. Will slow down import of small models, benchmarking recommended.	AMICI_IMPORT_NPROCS=4
AMICI_EXPERIMENTAL_SBML_IMPORT_NONCONSTANTS	Non-constant SBML compartment and species laws for non-constant species. SBML-import only. See <code>amici.sbml_import.SbmlImporter.sbml2amici()</code> .	

### 10.3.3 Miscellaneous

#### OpenMP support for parallelized simulation for multiple experimental conditions

AMICI can be built with OpenMP support, which allows to parallelize model simulations for multiple experimental conditions.

On Linux and OSX this is enabled by default. This can be verified using:

```
import amici
amici.compileWithOpenMP()
```

If not already enabled by default, you can enable OpenMP support by setting the environment variables `AMICI_CXXFLAGS` and `AMICI_LDFLAGS` to the correct OpenMP flags of your compiler and linker, respectively. This has to be done for both AMICI package installation *and* model compilation. When using gcc on Linux, this would be:

```
# on your shell:
AMICI_CXXFLAGS=-fopenmp AMICI_LDFLAGS=-fopenmp pip3 install amici
```

```
# in python, before model compilation:
import os
os.environ['AMICI_CXXFLAGS'] = '-fopenmp'
os.environ['AMICI_LDFLAGS'] = '-fopenmp'
```

## 10.4 FAQ

**Q:** I am trying to install the AMICI Python package, but installation fails with something like

```
amici/src/cblas.cpp:16:13: fatal error: cblas.h: No such file or directory
#include <cblas.h>
          ^~~~~~
compilation terminated.
error: command 'x86_64-linux-gnu-gcc' failed with exit status 1
```

**A:** You will have to install a CBLAS-compatible BLAS library and/or set `BLAS_CFLAGS` as described in the [installation guide](#).

---

**Q:** Importing my model fails with something like `ImportError: _someModelName.cpython-37m-x86_64-linux-gnu.so: undefined symbol: omp_get_thread_num`.

**A:** You probably installed the AMICI package with OpenMP support, but did not have the relevant compiler/linker flags set when importing/building the model. See [here](#).

## 10.5 AMICI Python API

## Modules

<code>amici</code>	AMICI
<code>amici.amici</code>	Core C++ bindings This module encompasses the complete public C++ API of AMICI, which was exposed via swig. All functions listed here are directly accessible in the main amici package, i.e., <code>amici.amici.ExpData</code> is available as <code>amici.ExpData</code> . Usage of functions and classes from the base amici package is generally recommended as they often include convenience wrappers that avoid common pitfalls when accessing C++ types from python and implement some nonstandard type conversions.
<code>amici.sbml_import</code>	SBML Import This module provides all necessary functionality to import a model specified in the Systems Biology Markup Language (SBML).
<code>amici.pysb_import</code>	PySB Import This module provides all necessary functionality to import a model specified in the <code>pysb.core.Model</code> format.
<code>amici.bngl_import</code>	BNGL Import This module provides all necessary functionality to import a model specified in the BNGL format.
<code>amici.petab_import</code>	PETab Import Import a model in the petab ( <a href="https://github.com/PETab-dev/PETab">https://github.com/PETab-dev/PETab</a> ) format into AMICI.
<code>amici.petab_import_pysb</code>	PySB-PETab Import Import a model in the PySB-adapted petab ( <a href="https://github.com/PETab-dev/PETab">https://github.com/PETab-dev/PETab</a> ) format into AMICI.
<code>amici.petab_objective</code>	PETab Objective Functionality related to running simulations or evaluating the objective function as defined by a PETab problem
<code>amici.petab_simulate</code>	PETab Simulate Functionality related to the use of AMICI for simulation with PETab's Simulator class.
<code>amici.import_utils</code>	Miscellaneous functions related to model import, independent of any specific model format
<code>amici.de_export</code>	C++ Export This module provides all necessary functionality specify an DE model and generate executable C++ simulation code. The user generally won't have to directly call any function from this module as this will be done by <code>amici.pysb_import.pysb2amici()</code> , <code>amici.sbml_import.SbmlImporter.sbml2amici()</code> and <code>amici.petab_import.import_model()</code> .
<code>amici.de_model</code>	Objects for AMICI's internal differential equation model representation
<code>amici.plotting</code>	Plotting Plotting related functions
<code>amici.pandas</code>	Pandas Wrappers This module contains convenience wrappers that allow for easy interconversion between C++ objects from <code>amici.amici</code> and pandas DataFrames
<code>amici.logging</code>	Logging This module provides custom logging functionality for other amici modules
<b>10.5. AMICI Python API</b>	<b>263</b>
<code>amici.gradient_check</code>	Finite Difference Check This module provides functions to automatically check correctness of amici computed sensitivities using finite

## 10.5.1 amici

### AMICI

The AMICI Python module provides functionality for importing SBML or PySB models and turning them into C++ Python extensions.

#### Module Attributes

<i>amici_path</i>	absolute root path of the amici repository or Python package
<i>amiciSwigPath</i>	absolute path of the amici swig directory
<i>amiciSrcPath</i>	absolute path of the amici source directory
<i>amiciModulePath</i>	absolute root path of the amici module
<i>hdf5_enabled</i>	boolean indicating if amici was compiled with hdf5 support

#### Functions

<i>import_model_module</i> (module_name[, module_path])	mod-	Import Python module of an AMICI model
---	------	--

#### Classes

<i>ModelModule</i> (*args, **kwargs)	Type of AMICI-generated model modules.
<i>add_path</i> (path)	Context manager for temporarily changing PYTHON-PATH

#### Exceptions

<i>AmiciVersionError</i>	Error thrown if an AMICI model is loaded that is incompatible with the installed AMICI base package
--------------------------	---

#### exception amici.AmiciVersionError

Error thrown if an AMICI model is loaded that is incompatible with the installed AMICI base package

**\_\_init\_\_**(\*args, \*\*kwargs)

**args**

**with\_traceback()**

Exception.with\_traceback(tb) – set self.\_\_traceback\_\_ to tb and return self.

#### class amici.ModelModule(\*args, \*\*kwargs)

Type of AMICI-generated model modules.

To enable static type checking.

```

__init__(*args, **kwargs)

getModel()
    Create a model instance.

    Return type amici.amici.Model

get_model()
    Create a model instance.

    Return type amici.amici.Model

class amici.add_path(path)
    Context manager for temporarily changing PYTHONPATH

    __init__(path)

amici.amiciModulePath = '/home/docs/checkouts/readthedocs.org/user_builds/amici/
checkouts/v0.20.0/python/sdist/amici'
    absolute root path of the amici module

amici.amiciSrcPath = '/home/docs/checkouts/readthedocs.org/user_builds/amici/checkouts/
v0.20.0/python/sdist/amici/src'
    absolute path of the amici source directory

amici.amiciSwigPath = '/home/docs/checkouts/readthedocs.org/user_builds/amici/checkouts/
v0.20.0/python/sdist/amici/swig'
    absolute path of the amici swig directory

amici.amici_path = '/home/docs/checkouts/readthedocs.org/user_builds/amici/checkouts/v0.
20.0/python/sdist/amici'
    absolute root path of the amici repository or Python package

amici.hdf5_enabled: bool = False
    boolean indicating if amici was compiled with hdf5 support

amici.import_model_module(module_name, module_path=None)
    Import Python module of an AMICI model

    Parameters
    • module_name (str) – Name of the python package of the model
    • module_path (typing.Union[pathlib.Path, str, None]) – Absolute or relative path of
      the package directory

    Return type amici.ModelModule

    Returns The model module

```

## 10.5.2 amici.amici

### Core C++ bindings

This module encompasses the complete public C++ API of AMICI, which was exposed via swig. All functions listed here are directly accessible in the main amici package, i.e., `amici.amici.ExpData` is available as `amici.ExpData`. Usage of functions and classes from the base amici package is generally recommended as they often include convenience wrappers that avoid common pitfalls when accessing C++ types from python and implement some nonstandard type conversions.

### Module Attributes

<code>SensitivityOrder_none</code>	Don't compute sensitivities.
<code>SensitivityOrder_first</code>	First-order sensitivities.
<code>SensitivityOrder_second</code>	Second-order sensitivities.
<code>SensitivityMethod_none</code>	Don't compute sensitivities.
<code>SensitivityMethod_forward</code>	Forward sensitivity analysis.
<code>SensitivityMethod_adjoint</code>	Adjoint sensitivity analysis.
<code>NonlinearSolverIteration_fixedpoint</code>	deprecated

### Functions

<code>compiledWithOpenMP()</code>	AMICI extension was compiled with OpenMP?
<code>enum(prefix)</code>	
<code>getScaledParameter(unscaledParameter, scaling)</code>	Apply parameter scaling according to <i>scaling</i>
<code>getUnscaledParameter(scaledParameter, scaling)</code>	Remove parameter scaling according to <i>scaling</i>
<code>parameterScalingFromIntVector(intVec)</code>	Swig-Generated class, which, in contrast to other Vector classes, does not allow for simple interoperability with common Python types, but must be created using <code>amici.amici.parameterScalingFromIntVector()</code>
<code>runAmiciSimulation(solver, edata, model[, ...])</code>	Core integration routine.
<code>runAmiciSimulations(solver, edatas, model, ...)</code>	Same as runAmiciSimulation, but for multiple ExpData instances.
<code>scaleParameters(bufferUnscaled, pscale, ...)</code>	Apply parameter scaling according to <i>scaling</i>
<code>simulation_status_to_str(status)</code>	Get the string representation of the given simulation status code (see ReturnData::status).
<code>unscaleParameters(bufferScaled, pscale, ...)</code>	Remove parameter scaling according to the parameter scaling in pscale

## Classes

<i>BoolVector</i> (*args)	Swig-Generated class templating common python types including <code>Iterable</code> [ <code>bool</code> ] and <code>numpy.array</code> [ <code>bool</code> ] to facilitate interfacing with C++ bindings.
<i>CpuTimer</i> ()	Tracks elapsed CPU time using <code>std::clock</code> .
<i>DoubleVector</i> (*args)	Swig-Generated class templating common python types including <code>Iterable</code> [ <code>float</code> ] and <code>numpy.array</code> [ <code>float</code> ] to facilitate interfacing with C++ bindings.
<i>ExpData</i> (*args)	<code>ExpData</code> carries all information about experimental or condition-specific data
<i>ExpDataPtr</i> (*args)	Swig-Generated class that implements smart pointers to <code>ExpData</code> as objects.
<i>ExpDataPtrVector</i> (*args)	Swig-Generated class templating common python types including <code>Iterable</code> [ <code>amici.amici.ExpData</code> ] and <code>numpy.array</code> [ <code>amici.amici.ExpData</code> ] to facilitate interfacing with C++ bindings.
<i>FixedParameterContext</i> (value)	An enumeration.
<i>IntVector</i> (*args)	Swig-Generated class templating common python types including <code>Iterable</code> [ <code>int</code> ] and <code>numpy.array</code> [ <code>int</code> ] to facilitate interfacing with C++ bindings.
<i>InternalSensitivityMethod</i> (value)	An enumeration.
<i>InterpolationType</i> (value)	An enumeration.
<i>LinearMultistepMethod</i> (value)	An enumeration.
<i>LinearSolver</i> (value)	An enumeration.
<i>LogItem</i> (*args)	A log item.
<i>LogItemVector</i> (*args)	
<i>Model</i> (*args, **kwargs)	The <code>Model</code> class represents an AMICI ODE/DAE model.
<i>ModelDimensions</i> (*args)	Container for model dimensions.
<i>ModelPtr</i> (*args)	Swig-Generated class that implements smart pointers to <code>Model</code> as objects.
<i>NewtonDampingFactorMode</i> (value)	An enumeration.
<i>NonlinearSolverIteration</i> (value)	An enumeration.
<i>ObservableScaling</i> (value)	An enumeration.
<i>ParameterScaling</i> (value)	An enumeration.
<i>ParameterScalingVector</i> (*args)	
<i>RDataReporting</i> (value)	An enumeration.
<i>ReturnData</i> (*args)	Stores all data to be returned by <code>amici.amici.runAmiciSimulation()</code> .
<i>ReturnDataPtr</i> (*args)	Swig-Generated class that implements smart pointers to <code>ReturnData</code> as objects.
<i>SecondOrderMode</i> (value)	An enumeration.
<i>SensitivityMethod</i> (value)	An enumeration.
<i>SensitivityOrder</i> (value)	An enumeration.
<i>SimulationParameters</i> (*args)	Container for various simulation parameters.
<i>Solver</i> (*args, **kwargs)	The <code>Solver</code> class provides a generic interface to CVODES and IDAS solvers, individual realizations are realized in the <code>CVodeSolver</code> and the <code>IDASolver</code> class.

continues on next page

Table 2 – continued from previous page

<code>SolverPtr(*args)</code>	Swig-Generated class that implements smart pointers to Solver as objects.
<code>SteadyStateComputationMode(value)</code>	An enumeration.
<code>SteadyStateSensitivityMode(value)</code>	An enumeration.
<code>SteadyStateStatus(value)</code>	An enumeration.
<code>SteadyStateStatusVector(*args)</code>	
<code>StringDoubleMap(*args)</code>	Swig-Generated class templating Dict [ <code>str</code> , <code>float</code> ] to facilitate interfacing with C++ bindings.
<code>StringVector(*args)</code>	Swig-Generated class templating common python types including Iterable [ <code>str</code> ] and <code>numpy.array</code> [ <code>str</code> ] to facilitate interfacing with C++ bindings.

**class** amici.amici.**BoolVector**(\*args)

Swig-Generated class templating common python types including Iterable [`bool`] and `numpy.array` [`bool`] to facilitate interfacing with C++ bindings.

**class** amici.amici.**CpuTimer**

Tracks elapsed CPU time using `std::clock`.

**\_\_init\_\_**()

Constructor

**elapsed\_milliseconds**() → `float`

Get elapsed CPU time in milliseconds since initialization or last reset

**Return type** `float`

**Returns** CPU time in milliseconds

**elapsed\_seconds**() → `float`

Get elapsed CPU time in seconds since initialization or last reset

**Return type** `float`

**Returns** CPU time in seconds

**reset**()

Reset the timer

**uses\_thread\_clock** = `False`

**class** amici.amici.**DoubleVector**(\*args)

Swig-Generated class templating common python types including Iterable [`float`] and `numpy.array` [`float`] to facilitate interfacing with C++ bindings.

**class** amici.amici.**ExpData**(\*args)

ExpData carries all information about experimental or condition-specific data

**\_\_init\_\_**(\*args)

Overload 1:

default constructor



*Overload 2:*

Copy constructor, needs to be declared to be generated in swig

*Overload 3:*

constructor that only initializes dimensions

**Parameters**

- **nytrue** (*int*) – Number of observables
- **nztrue** (*int*) – Number of event outputs
- **nmaxevent** (*int*) – Maximal number of events to track

*Overload 4:*

constructor that initializes timepoints from vectors

**Parameters**

- **nytrue** (*int*) – Number of observables
- **nztrue** (*int*) – Number of event outputs
- **nmaxevent** (*int*) – Maximal number of events to track
- **ts** (*DoubleVector*) – Timepoints (dimension: nt)

*Overload 5:*

constructor that initializes timepoints and fixed parameters from vectors

**Parameters**

- **nytrue** (*int*) – Number of observables
- **nztrue** (*int*) – Number of event outputs
- **nmaxevent** (*int*) – Maximal number of events to track
- **ts** (*DoubleVector*) – Timepoints (dimension: nt)
- **fixedParameters** (*DoubleVector*) – Model constants (dimension: nk)

*Overload 6:*

constructor that initializes timepoints and data from vectors

**Parameters**

- **nytrue** (*int*) – Number of observables
- **nztrue** (*int*) – Number of event outputs
- **nmaxevent** (*int*) – Maximal number of events to track
- **ts** (*DoubleVector*) – Timepoints (dimension: nt)
- **observedData** (*DoubleVector*) – observed data (dimension: nt x nytrue, row-major)
- **observedDataStdDev** (*DoubleVector*) – standard deviation of observed data (dimension: nt x nytrue, row-major)
- **observedEvents** (*DoubleVector*) – observed events (dimension: nmaxevents x nztrue, row-major)
- **observedEventsStdDev** (*DoubleVector*) – standard deviation of observed events/roots (dimension: nmaxevents x nztrue, row-major)

*Overload 7:*

constructor that initializes with Model

**Parameters** **model** (*Model*) – pointer to model specification object

*Overload 8:*

constructor that initializes with returnData, adds noise according to specified sigmas

**Parameters**

- **rdata** (*ReturnData*) – return data pointer with stored simulation results
- **sigma\_y** (*float*) – scalar standard deviations for all observables
- **sigma\_z** (*float*) – scalar standard deviations for all event observables

*Overload 9:*

constructor that initializes with returnData, adds noise according to specified sigmas

**Parameters**

- **rdata** (*ReturnData*) – return data pointer with stored simulation results

- **sigma\_y** (*DoubleVector*) – vector of standard deviations for observables (dimension: nytrue or nt x nytrue, row-major)
- **sigma\_z** (*DoubleVector*) – vector of standard deviations for event observables (dimension: nztrue or nmaxevent x nztrue, row-major)

**property fixedParameters**

Model constants

Vector of size Model::nk() or empty

**property fixedParametersPreequilibration**

Model constants for pre-equilibration

Vector of size Model::nk() or empty.

**property fixedParametersPresimulation**

Model constants for pre-simulation

Vector of size Model::nk() or empty.

**getObservedData()** → *Tuple*[float]

Get all measurements.

**Return type** *DoubleVector*

**Returns** observed data (dimension: nt x nytrue, row-major)

**getObservedDataPtr(it: int)** → amici::realtype const \*

Get measurements for a given timepoint index.

**Parameters** **it** (int) – timepoint index

**Return type** float

**Returns** pointer to observed data at index (dimension: nytrue)

**getObservedDataStdDev()** → *Tuple*[float]

Get measurement standard deviations.

**Return type** *DoubleVector*

**Returns** standard deviation of observed data

**getObservedDataStdDevPtr(it: int)** → amici::realtype const \*

Get pointer to measurement standard deviations.

**Parameters** **it** (int) – timepoint index

**Return type** float

**Returns** pointer to standard deviation of observed data at index

**getObservedEvents()** → *Tuple*[float]

get function that copies data from ExpData::mz to output

**Return type** *DoubleVector*

**Returns** observed event data

**getObservedEventsPtr(ie: int)** → amici::realtype const \*

get function that returns a pointer to observed data at ieth occurrence

**Parameters** **ie** (int) – event occurrence

**Return type** `float`

**Returns** pointer to observed event data at ieth occurrence

**getObservedEventsStdDev()**  $\rightarrow$  `Tuple[float]`

get function that copies data from `ExpData::observedEventsStdDev` to output

**Return type** `DoubleVector`

**Returns** standard deviation of observed event data

**getObservedEventsStdDevPtr**(*ie*: `int`)  $\rightarrow$  `amici::realtype const *`

get function that returns a pointer to standard deviation of observed event data at ie-th occurrence

**Parameters** **ie** (`int`) – event occurrence

**Return type** `float`

**Returns** pointer to standard deviation of observed event data at ie-th occurrence

**getTimepoint**(*it*: `int`)  $\rightarrow$  `float`

Get timepoint for the given index

**Parameters** **it** (`int`) – timepoint index

**Return type** `float`

**Returns** timepoint timepoint at index

**getTimepoints()**  $\rightarrow$  `Tuple[float]`

Get output timepoints.

**Return type** `DoubleVector`

**Returns** `ExpData::ts`

**property id**

Arbitrary (not necessarily unique) identifier.

**isSetObservedData**(*it*: `int`, *iy*: `int`)  $\rightarrow$  `bool`

Whether there is a measurement for the given time- and observable- index.

**Parameters**

- **it** (`int`) – time index
- **iy** (`int`) – observable index

**Return type** `boolean`

**Returns** `boolean` specifying if data was set

**isSetObservedDataStdDev**(*it*: `int`, *iy*: `int`)  $\rightarrow$  `bool`

Whether standard deviation for a measurement at specified timepoint- and observable index has been set.

**Parameters**

- **it** (`int`) – time index
- **iy** (`int`) – observable index

**Return type** `boolean`

**Returns** `boolean` specifying if standard deviation of data was set

**isSetObservedEvents**(*ie*: *int*, *iz*: *int*) → *bool*

get function that checks whether event data at specified indices has been set

**Parameters**

- **ie** (*int*) – event index
- **iz** (*int*) – event observable index

**Return type** *boolean*

**Returns** *boolean* specifying if data was set

**isSetObservedEventsStdDev**(*ie*: *int*, *iz*: *int*) → *bool*

get function that checks whether standard deviation of even data at specified indices has been set

**Parameters**

- **ie** (*int*) – event index
- **iz** (*int*) – event observable index

**Return type** *boolean*

**Returns** *boolean* specifying if standard deviation of event data was set

**nmaxevent**() → *int*

maximal number of events to track

**Return type** *int*

**Returns** maximal number of events to track

**nt**() → *int*

number of timepoints

**Return type** *int*

**Returns** number of timepoints

**nytrue**() → *int*

number of observables of the non-augmented model

**Return type** *int*

**Returns** number of observables of the non-augmented model

**nztrue**() → *int*

number of event observables of the non-augmented model

**Return type** *int*

**Returns** number of event observables of the non-augmented model

**property parameters**

Model parameters

Vector of size `Model::np()` or empty with parameter scaled according to `SimulationParameter::pscale`.

**property plist**

Parameter indices w.r.t. which to compute sensitivities

**property pscale**

Parameter scales

Vector of parameter scale of size `Model::np()`, indicating how/if each parameter is to be scaled.

**property reinitialization\_state\_idxes\_presim**

Indices of states to be reinitialized based on provided presimulation constants / fixed parameters.

**property reinitialization\_state\_idxes\_sim**

Indices of states to be reinitialized based on provided constants / fixed parameters.

**reinitializeAllFixedParameterDependentInitialStates**(*nx\_rdata*: *int*)

Set reinitialization of all states based on model constants for all simulation phases.

Convenience function to populate *reinitialization\_state\_idxes\_presim* and *reinitialization\_state\_idxes\_sim*

**Parameters** *nx\_rdata* (*int*) – Number of states (`Model::nx_rdata`)

**reinitializeAllFixedParameterDependentInitialStatesForPresimulation**(*nx\_rdata*: *int*)

Set reinitialization of all states based on model constants for presimulation (only meaningful if preequilibration is performed).

Convenience function to populate *reinitialization\_state\_idxes\_presim* and *reinitialization\_state\_idxes\_sim*

**Parameters** *nx\_rdata* (*int*) – Number of states (`Model::nx_rdata`)

**reinitializeAllFixedParameterDependentInitialStatesForSimulation**(*nx\_rdata*: *int*)

Set reinitialization of all states based on model constants for the ‘main’ simulation (only meaningful if presimulation or preequilibration is performed).

Convenience function to populate *reinitialization\_state\_idxes\_presim* and *reinitialization\_state\_idxes\_sim*

**Parameters** *nx\_rdata* (*int*) – Number of states (`Model::nx_rdata`)

**property reinitializeFixedParameterInitialStates**

Flag indicating whether reinitialization of states depending on fixed parameters is activated

**setObservedData**(\*args)

*Overload 1:*

Set all measurements.

**Parameters** *observedData* (*DoubleVector*) – observed data (dimension: *nt* x *nytrue*, row-major)

*Overload 2:*

Set measurements for a given observable index

**Parameters**

- *observedData* (*DoubleVector*) – observed data (dimension: *nt*)
- *iy* (*int*) – observed data index

**setObservedDataStdDev(\*args)***Overload 1:*

Set standard deviations for measurements.

**Parameters** **observedDataStdDev** (**DoubleVector**) – standard deviation of observed data (dimension: nt x nytrue, row-major)

*Overload 2:*

Set identical standard deviation for all measurements.

**Parameters** **stdDev** (*float*) – standard deviation (dimension: scalar)

*Overload 3:*

Set standard deviations of observed data for a specific observable index.

**Parameters**

- **observedDataStdDev** (**DoubleVector**) – standard deviation of observed data (dimension: nt)
- **iy** (*int*) – observed data index

*Overload 4:*

Set all standard deviation for a given observable index to the input value.

**Parameters**

- **stdDev** (*float*) – standard deviation (dimension: scalar)
- **iy** (*int*) – observed data index

**setObservedEvents(\*args)***Overload 1:*

set function that copies observed event data from input to ExpData::observedEvents

**Parameters** **observedEvents** (**DoubleVector**) – observed data (dimension: nmaxevent x nztrue, row-major)

*Overload 2:*

set function that copies observed event data for specific event observable

**Parameters**

- **observedEvents** (`DoubleVector`) – observed data (dimension: nmaxevent)
- **iz** (`int`) – observed event data index

**setObservedEventsStdDev**(\*args)*Overload 1:*

set function that copies data from input to ExpData::observedEventsStdDev

**Parameters** **observedEventsStdDev** (`DoubleVector`) – standard deviation of observed event data*Overload 2:*

set function that sets all ExpData::observedDataStdDev to the input value

**Parameters** **stdDev** (`float`) – standard deviation (dimension: scalar)*Overload 3:*

set function that copies standard deviation of observed data for specific observable

**Parameters**

- **observedEventsStdDev** (`DoubleVector`) – standard deviation of observed data (dimension: nmaxevent)
- **iz** (`int`) – observed data index

*Overload 4:*

set function that sets all standard deviation of a specific observable to the input value

**Parameters**

- **stdDev** (`float`) – standard deviation (dimension: scalar)
- **iz** (`int`) – observed data index

**setTimepoints**(ts: `Sequence[float]`)

Set output timepoints.

**Parameters** **ts** (`typing.Sequence[float]`) – timepoints**property** **sx0**

Initial state sensitivities

Dimensions: Model::nx() \* Model::nplist(), Model::nx() \* ExpData::plist.size(), if ExpData::plist is not empty, or empty



**property t\_presim**

Duration of pre-simulation.

If this is > 0, presimulation will be performed from (model->t0 - t\_presim) to model->t0 using the fixed-Parameters in fixedParametersPresimulation

**property ts\_**

Timepoints for which model state/outputs/... are requested

Vector of timepoints.

**property tstart\_**

starting time

**property x0**

Initial state

Vector of size Model::nx() or empty

**class amici.amici.ExpDataPtr(\*args)**

Swig-Generated class that implements smart pointers to ExpData as objects.

**property fixedParameters**

Model constants

Vector of size Model::nk() or empty

**property fixedParametersPreequilibration**

Model constants for pre-equilibration

Vector of size Model::nk() or empty.

**property fixedParametersPresimulation**

Model constants for pre-simulation

Vector of size Model::nk() or empty.

**property id**

Arbitrary (not necessarily unique) identifier.

**property parameters**

Model parameters

Vector of size Model::np() or empty with parameter scaled according to SimulationParameter::pscale.

**property plist**

Parameter indices w.r.t. which to compute sensitivities

**property pscale**

Parameter scales

Vector of parameter scale of size Model::np(), indicating how/if each parameter is to be scaled.

**property reinitialization\_state\_idxsim**

Indices of states to be reinitialized based on provided presimulation constants / fixed parameters.

**property reinitialization\_state\_idxsim**

Indices of states to be reinitialized based on provided constants / fixed parameters.

**property reinitializeFixedParameterInitialStates**

Flag indicating whether reinitialization of states depending on fixed parameters is activated

**property `sx0`**

Initial state sensitivities

Dimensions: `Model::nx() * Model::nplist()`, `Model::nx() * ExpData::plist.size()`, if `ExpData::plist` is not empty, or empty

**property `t_presim`**

Duration of pre-simulation.

If this is  $> 0$ , presimulation will be performed from `(model->t0 - t_presim)` to `model->t0` using the fixed-Parameters in `fixedParametersPresimulation`

**property `ts_`**

Timepoints for which model state/outputs/... are requested

Vector of timepoints.

**property `tstart_`**

starting time

**property `x0`**

Initial state

Vector of size `Model::nx()` or empty

**class `amici.amici.ExpDataPtrVector(*args)`**

Swig-Generated class templating common python types including `Iterable` [[`amici.amici.ExpData`](#)] and `numpy.array` [[`amici.amici.ExpData`](#)] to facilitate interfacing with C++ bindings.

**class `amici.amici.FixedParameterContext(value)`**

An enumeration.

**`preequilibration = 1`**

**`presimulation = 2`**

**`simulation = 0`**

**class `amici.amici.IntVector(*args)`**

Swig-Generated class templating common python types including `Iterable` [`int`] and `numpy.array` [`int`] to facilitate interfacing with C++ bindings.

**class `amici.amici.InternalSensitivityMethod(value)`**

An enumeration.

**`simultaneous = 1`**

**`staggered = 2`**

**`staggered1 = 3`**

**class `amici.amici.InterpolationType(value)`**

An enumeration.

**`hermite = 1`**

**`polynomial = 2`**

**class `amici.amici.LinearMultistepMethod(value)`**

An enumeration.

```

BDF = 2

adams = 1

class amici.amici.LinearSolver(value)
    An enumeration.

    KLU = 9

    LAPACKBand = 4

    LAPACKDense = 3

    SPBCG = 7

    SPGMR = 6

    SPTFQMR = 8

    SuperLUMT = 10

    band = 2

    dense = 1

    diag = 5

class amici.amici.LogItem(*args)
    A log item.

    __init__(*args)
        Overload 1:
        Default ctor.

        Overload 2:
        Construct a LogItem

        Parameters
        • severity (int) –
        • identifier (str) –
        • message (str) –

    property identifier
        Short identifier for the logged event

    property message
        A more detailed and readable message

    property severity
        Severity level

class amici.amici.LogItemVector(*args)

```

```
class amici.amici.Model(*args, **kwargs)
```

The Model class represents an AMICI ODE/DAE model.

The model can compute various model related quantities based on symbolically generated code.

```
__init__(*args, **kwargs)
```

*Overload 1:* Default ctor

*Overload 2:*

Constructor with model dimensions

#### Parameters

- **nx\_rdata** (*int*) – Number of state variables
- **nxtrue\_rdata** (*int*) – Number of state variables of the non-augmented model
- **nx\_solver** (*int*) – Number of state variables with conservation laws applied
- **nxtrue\_solver** (*int*) – Number of state variables of the non-augmented model with conservation laws applied
- **nx\_solver\_reinit** (*int*) – Number of state variables with conservation laws subject to reinitialization
- **np** (*int*) – Number of parameters
- **nk** (*int*) – Number of constants
- **ny** (*int*) – Number of observables
- **nytrue** (*int*) – Number of observables of the non-augmented model
- **nz** (*int*) – Number of event observables
- **nztrue** (*int*) – Number of event observables of the non-augmented model
- **ne** (*int*) – Number of events
- **nspl** (*int*) – Number of splines
- **nJ** (*int*) – Number of objective functions
- **nw** (*int*) – Number of repeating elements
- **ndwdx** (*int*) – Number of nonzero elements in the  $x$  derivative of the repeating elements
- **ndwdp** (*int*) – Number of nonzero elements in the  $p$  derivative of the repeating elements
- **ndwdw** (*int*) – Number of nonzero elements in the  $w$  derivative of the repeating elements
- **ndxdotdw** (*int*) – Number of nonzero elements in the  $w$  derivative of  $x_{dot}$
- **ndJydy** (*IntVector*) – Number of nonzero elements in the  $y$  derivative of  $dJy$  (shape *nytrue*)
- **ndxrdatadxsolver** (*int*) – Number of nonzero elements in the  $x$  derivative of  $x_rdata$
- **ndxrdatadtcl** (*int*) – Number of nonzero elements in the  $tcl$  derivative of  $x_rdata$

- **ndtotal\_cldx\_rdata** (*int*) – Number of nonzero elements in the  $x_r$  data derivative of  $total_{cl}$
- **nnz** (*int*) – Number of nonzero elements in Jacobian
- **ubw** (*int*) – Upper matrix bandwidth in the Jacobian
- **lbw** (*int*) – Lower matrix bandwidth in the Jacobian

**clone()** → *amici.amici.Model*

Clone this instance.

**Return type** *Model*

**Returns** The clone

**fdsigmaydy**(*dsigmaydy: Iterable[float], t: float const, p: Iterable[float], k: Iterable[float], y: Iterable[float]*)

Model-specific implementation of fsigmay

**Parameters**

- **dsigmaydy** (*typing.Iterable[float]*) – partial derivative of standard deviation of measurements w.r.t. model outputs
- **t** (*float*) – current time
- **p** (*typing.Iterable[float]*) – parameter vector
- **k** (*typing.Iterable[float]*) – constant vector
- **y** (*typing.Iterable[float]*) – model output at timepoint t

**fdspline\_slopesdp**(*dspline\_slopesdp: Iterable[float], p: Iterable[float], k: Iterable[float], ip: int const*)

Model-specific implementation the parametric derivatives of slopevalues at spline nodes

**Parameters**

- **dspline\_slopesdp** (*typing.Iterable[float]*) – vector to which derivatives will be written
- **p** (*typing.Iterable[float]*) – parameter vector
- **k** (*typing.Iterable[float]*) – constants vector
- **ip** (*int*) – Sensitivity index

**fdspline\_valuesdp**(*dspline\_valuesdp: Iterable[float], p: Iterable[float], k: Iterable[float], ip: int const*)

Model-specific implementation the parametric derivatives of spline node values

**Parameters**

- **dspline\_valuesdp** (*typing.Iterable[float]*) – vector to which derivatives will be written
- **p** (*typing.Iterable[float]*) – parameter vector
- **k** (*typing.Iterable[float]*) – constants vector
- **ip** (*int*) – Sensitivity index

**fdtotal\_cldp**(*dtotal\_cldp: Iterable[float], x\_rdata: Iterable[float], p: Iterable[float], k: Iterable[float], ip: int const*)

Compute dtotal\_cl / dp

**Parameters**

- **dtotal\_cldp** (*typing.Iterable[float]*) – dtotal\_cl / dp

- **x\_rdata** (`typing.Iterable[float]`) – State variables with conservation laws applied
- **p** (`typing.Iterable[float]`) – parameter vector
- **k** (`typing.Iterable[float]`) – constant vector
- **ip** (`int`) – Sensitivity index

**fdtotal\_cldx\_rdata**(*dtotal\_cldx\_rdata: Iterable[float], x\_rdata: Iterable[float], p: Iterable[float], k: Iterable[float], tcl: Iterable[float]*)

Compute  $dtotal\_cl / dx\_rdata$

#### Parameters

- **dtotal\_cldx\_rdata** (`typing.Iterable[float]`) –  $dtotal\_cl / dx\_rdata$
- **x\_rdata** (`typing.Iterable[float]`) – State variables with conservation laws applied
- **p** (`typing.Iterable[float]`) – parameter vector
- **k** (`typing.Iterable[float]`) – constant vector
- **tcl** (`typing.Iterable[float]`) – Total abundances for conservation laws

**fdx\_rdatadp**(*dx\_rdatadp: Iterable[float], x: Iterable[float], tcl: Iterable[float], p: Iterable[float], k: Iterable[float], ip: int const*)

Compute  $dx\_rdata / dp$

#### Parameters

- **dx\_rdatadp** (`typing.Iterable[float]`) –  $dx\_rdata / dp$
- **p** (`typing.Iterable[float]`) – parameter vector
- **k** (`typing.Iterable[float]`) – constant vector
- **x** (`typing.Iterable[float]`) – State variables with conservation laws applied
- **tcl** (`typing.Iterable[float]`) – Total abundances for conservation laws
- **ip** (`int`) – Sensitivity index

**fdx\_rdatadtcl**(*dx\_rdatadtcl: Iterable[float], x: Iterable[float], tcl: Iterable[float], p: Iterable[float], k: Iterable[float]*)

Compute  $dx\_rdata / dtcl$

#### Parameters

- **dx\_rdatadtcl** (`typing.Iterable[float]`) –  $dx\_rdata / dtcl$
- **p** (`typing.Iterable[float]`) – parameter vector
- **k** (`typing.Iterable[float]`) – constant vector
- **x** (`typing.Iterable[float]`) – State variables with conservation laws applied
- **tcl** (`typing.Iterable[float]`) – Total abundances for conservation laws

**fdx\_rdatadx\_solver**(*dx\_rdatadx\_solver: Iterable[float], x: Iterable[float], tcl: Iterable[float], p: Iterable[float], k: Iterable[float]*)

Compute  $dx\_rdata / dx\_solver$

#### Parameters

- **dx\_rdatadx\_solver** (`typing.Iterable[float]`) –  $dx\_rdata / dx\_solver$
- **p** (`typing.Iterable[float]`) – parameter vector

- **k** (`typing.Iterable[float]`) – constant vector
- **x** (`typing.Iterable[float]`) – State variables with conservation laws applied
- **tcl** (`typing.Iterable[float]`) – Total abundances for conservation laws

**getAddSigmaResiduals()** → `bool`

Checks whether residuals should be added to account for parameter dependent sigma.

**Return type** `boolean`

**Returns** `sigma_res`

**getAlwaysCheckFinite()** → `bool`

Get setting of whether the result of every call to *Model::f\** should be checked for finiteness.

**Return type** `boolean`

**Returns** `that`

**getAmiciCommit()** → `str`

Returns the AMICI commit that was used to generate the model

**Return type** `str`

**Returns** AMICI commit string

**getAmiciVersion()** → `str`

Returns the AMICI version that was used to generate the model

**Return type** `str`

**Returns** AMICI version string

**getExpressionIds()** → `Tuple[str]`

Get IDs of the expression.

**Return type** `StringVector`

**Returns** Expression IDs

**getExpressionNames()** → `Tuple[str]`

Get names of the expressions.

**Return type** `StringVector`

**Returns** Expression names

**getFixedParameterById(par\_id: str)** → `float`

Get value of fixed parameter with the specified ID.

**Parameters** **par\_id** (`str`) – Parameter ID

**Return type** `float`

**Returns** Parameter value

**getFixedParameterByName(par\_name: str)** → `float`

Get value of fixed parameter with the specified name.

If multiple parameters have the same name, the first parameter with matching name is returned.

**Parameters** **par\_name** (`str`) – Parameter name

**Return type** `float`

**Returns** Parameter value

**getFixedParameterIds()** → `Tuple[str]`

Get IDs of the fixed model parameters.

**Return type** *StringVector*

**Returns** Fixed parameter IDs

**getFixedParameterNames()** → `Tuple[str]`

Get names of the fixed model parameters.

**Return type** *StringVector*

**Returns** Fixed parameter names

**getFixedParameters()** → `Tuple[float]`

Get values of fixed parameters.

**Return type** *DoubleVector*

**Returns** Vector of fixed parameters with same ordering as in `Model::getFixedParameterIds`

**getInitialStateSensitivities()** → `Tuple[float]`

Get the initial states sensitivities.

**Return type** *DoubleVector*

**Returns** vector of initial state sensitivities

**getInitialStates()** → `Tuple[float]`

Get the initial states.

**Return type** *DoubleVector*

**Returns** Initial state vector

**getMinimumSigmaResiduals()** → `float`

Gets the specified estimated lower boundary for `sigma_y`.

**Return type** `float`

**Returns** lower boundary

**getName()** → `str`

Get the model name.

**Return type** `str`

**Returns** Model name

**getObservableIds()** → `Tuple[str]`

Get IDs of the observables.

**Return type** *StringVector*

**Returns** Observable IDs

**getObservableNames()** → `Tuple[str]`

Get names of the observables.

**Return type** *StringVector*

**Returns** Observable names



**getObservableScaling**(*iy*: *int*) → *amici.amici.ObservableScaling*

Get scaling type for observable

**Parameters** *iy* (*int*) – observable index

**Return type** *int*

**Returns** scaling type

**getParameterById**(*par\_id*: *str*) → *float*

Get value of first model parameter with the specified ID.

**Parameters** *par\_id* (*str*) – Parameter ID

**Return type** *float*

**Returns** Parameter value

**getParameterByName**(*par\_name*: *str*) → *float*

Get value of first model parameter with the specified name.

**Parameters** *par\_name* (*str*) – Parameter name

**Return type** *float*

**Returns** Parameter value

**getParameterIds**() → *Tuple[str]*

Get IDs of the model parameters.

**Return type** *StringVector*

**Returns** Parameter IDs

**getParameterList**() → *Tuple[int]*

Get the list of parameters for which sensitivities are computed.

**Return type** *IntVector*

**Returns** List of parameter indices

**getParameterNames**() → *Tuple[str]*

Get names of the model parameters.

**Return type** *StringVector*

**Returns** The parameter names

**getParameterScale**() → *amici.amici.ParameterScalingVector*

Get parameter scale for each parameter.

**Return type** *ParameterScalingVector*

**Returns** Vector of parameter scales

**getParameters**() → *Tuple[float]*

Get parameter vector.

**Return type** *DoubleVector*

**Returns** The user-set parameters (see also *Model::getUnscaledParameters*)

**getReinitializationStateIdxs()** → `Tuple[int]`

Return indices of states to be reinitialized based on provided constants / fixed parameters

**Return type** *IntVector*

**Returns** Those indices.

**getReinitializeFixedParameterInitialStates()** → `bool`

Get whether initial states depending on fixedParameters are to be reinitialized after preequilibration and presimulation.

**Return type** `boolean`

**Returns** flag *true* / *false*

**getSolver()** → *amici.amici.Solver*

Retrieves the solver object

**Return type** *Solver*

**Returns** The Solver instance

**getStateIds()** → `Tuple[str]`

Get IDs of the model states.

**Return type** *StringVector*

**Returns** State IDs

**getStateIdsSolver()** → `Tuple[str]`

Get IDs of the solver states.

**Return type** *StringVector*

**Returns** State IDs

**getStateIsNonNegative()** → `Tuple[bool]`

Get flags indicating whether states should be treated as non-negative.

**Return type** *BoolVector*

**Returns** Vector of flags

**getStateNames()** → `Tuple[str]`

Get names of the model states.

**Return type** *StringVector*

**Returns** State names

**getStateNamesSolver()** → `Tuple[str]`

Get names of the solver states.

**Return type** *StringVector*

**Returns** State names

**getSteadyStateComputationMode()** → `amici::SteadyStateComputationMode`

Gets the mode how steady state is computed in the steadystate simulation.

**Return type** `int`

**Returns** Mode

**getSteadyStateSensitivityMode()** → *amici.amici.SteadyStateSensitivityMode*

Gets the mode how sensitivities are computed in the steadystate simulation.

**Return type** *SteadyStateSensitivityMode*

**Returns** Mode

**getTimepoint(it: int)** → float

Get simulation timepoint for time index *it*.

**Parameters** *it* (int) – Time index

**Return type** float

**Returns** Timepoint

**getTimepoints()** → Tuple[float]

Get the timepoint vector.

**Return type** *DoubleVector*

**Returns** Timepoint vector

**getUnscaledParameters()** → Tuple[float]

Get parameters with transformation according to parameter scale applied.

**Return type** *DoubleVector*

**Returns** Unscaled parameters

**hasCustomInitialStateSensitivities()** → bool

Return whether custom initial state sensitivities have been set.

**Return type** boolean

**Returns** *true* if has custom initial state sensitivities, otherwise *false*.

**hasCustomInitialStates()** → bool

Return whether custom initial states have been set.

**Return type** boolean

**Returns** *true* if has custom initial states, otherwise *false*

**hasExpressionIds()** → bool

Report whether the model has expression IDs set.

**Return type** boolean

**Returns** Boolean indicating whether expression ids were set. Also returns *true* if the number of corresponding variables is just zero.

**hasExpressionNames()** → bool

Report whether the model has expression names set.

**Return type** boolean

**Returns** Boolean indicating whether expression names were set. Also returns *true* if the number of corresponding variables is just zero.

**hasFixedParameterIds()** → bool

Report whether the model has fixed parameter IDs set.

**Return type** boolean

**Returns** Boolean indicating whether fixed parameter IDs were set. Also returns *true* if the number of corresponding variables is just zero.

**hasFixedParameterNames()** → *bool*

Report whether the model has fixed parameter names set.

**Return type** boolean

**Returns** Boolean indicating whether fixed parameter names were set. Also returns *true* if the number of corresponding variables is just zero.

**hasObservableIds()** → *bool*

Report whether the model has observable IDs set.

**Return type** boolean

**Returns** Boolean indicating whether observable ids were set. Also returns *true* if the number of corresponding variables is just zero.

**hasObservableNames()** → *bool*

Report whether the model has observable names set.

**Return type** boolean

**Returns** Boolean indicating whether observable names were set. Also returns *true* if the number of corresponding variables is just zero.

**hasParameterIds()** → *bool*

Report whether the model has parameter IDs set.

**Return type** boolean

**Returns** Boolean indicating whether parameter IDs were set. Also returns *true* if the number of corresponding variables is just zero.

**hasParameterNames()** → *bool*

Report whether the model has parameter names set.

**Return type** boolean

**Returns** Boolean indicating whether parameter names were set. Also returns *true* if the number of corresponding variables is just zero.

**hasQuadraticLLH()** → *bool*

Checks whether the defined noise model is gaussian, i.e., the nllh is quadratic

**Return type** boolean

**Returns** boolean flag

**hasStateIds()** → *bool*

Report whether the model has state IDs set.

**Return type** boolean

**Returns** Boolean indicating whether state IDs were set. Also returns *true* if the number of corresponding variables is just zero.

**hasStateNames()** → *bool*

Report whether the model has state names set.

**Return type** boolean

**Returns** Boolean indicating whether state names were set. Also returns *true* if the number of corresponding variables is just zero.

**property idlist**

Flag array for DAE equations

**initializeSplineSensitivities()**

Initialization of spline sensitivity functions

**initializeSplines()**

Initialization of spline functions

**isFixedParameterStateReinitializationAllowed()** → bool

Function indicating whether reinitialization of states depending on fixed parameters is permissible

**Return type** boolean

**Returns** flag indicating whether reinitialization of states depending on fixed parameters is permissible

**k()** → double const \*

Get fixed parameters.

**Return type** float

**Returns** Pointer to constants array

**property lbw**

Lower bandwidth of the Jacobian

**property logger**

Logger

**property nJ**

Dimension of the augmented objective function for 2nd order ASA

**nMaxEvent()** → int

Get maximum number of events that may occur for each type.

**Return type** int

**Returns** Maximum number of events that may occur for each type

**ncl()** → int

Get number of conservation laws.

**Return type** int

**Returns** Number of conservation laws (i.e., difference between *nx\_rdata* and *nx\_solver*).

**property ndJydy**

Number of nonzero elements in the *y* derivative of *dJy* (dimension *nytrue*)

**property ndtotal\_cldx\_rdata**

Number of nonzero elements in the *x<sub>r</sub>* derivative of *total<sub>c</sub>*

**property ndwdp**

Number of nonzero elements in the *p* derivative of the repeating elements

**property ndwdw**

Number of nonzero elements in the *w* derivative of the repeating elements

**property ndwdx**

Number of nonzero elements in the  $x$  derivative of the repeating elements

**property ndxdotdw**

Number of nonzero elements in the  $w$  derivative of  $\dot{x}$

**property ndxrdatadtcl**

Number of nonzero elements in the  $tcl$  derivative of  $x_rdata$

**property ndxrdatadxsolver**

Number of nonzero elements in the  $x$  derivative of  $x_rdata$

**property ne**

Number of events

**nk()** → int

Get number of constants

**Return type** int

**Returns** Length of constant vector

**property nnz**

Number of nonzero entries in Jacobian

**np()** → int

Get total number of model parameters.

**Return type** int

**Returns** Length of parameter vector

**nplist()** → int

Get number of parameters wrt to which sensitivities are computed.

**Return type** int

**Returns** Length of sensitivity index vector

**property nspl**

number of spline functions in the model

**nt()** → int

Get number of timepoints.

**Return type** int

**Returns** Number of timepoints

**property nw**

Number of common expressions

**property nx\_rdata**

Number of states

**nx\_reinit()** → int

Get number of solver states subject to reinitialization.

**Return type** int

**Returns** Model member `nx_solver_reinit`

**property nx\_solver**

Number of states with conservation laws applied

**property nx\_solver\_reinit**

Number of solver states subject to reinitialization

**property nxtrue\_rdata**

Number of states in the unaugmented system

**property nxtrue\_solver**

Number of states in the unaugmented system with conservation laws applied

**property ny**

Number of observables

**property nytrue**

Number of observables in the unaugmented system

**property nz**

Number of event outputs

**property nztrue**

Number of event outputs in the unaugmented system

**property o2mode**

Flag indicating whether for `amici::Solver::sensi_ == amici::SensitivityOrder::second` directional or full second order derivative will be computed

**plist**(*pos*: *int*) → *int*

Get entry in parameter list by index.

**Parameters** *pos* (*int*) – Index in sensitivity parameter list

**Return type** *int*

**Returns** Index in parameter list

**property pythonGenerated**

Flag indicating Matlab- or Python-based model generation

**requireSensitivitiesForAllParameters()**

Require computation of sensitivities for all parameters *p* [0..np[ in natural order.

NOTE: Resets initial state sensitivities.

**setAddSigmaResiduals**(*sigma\_res*: *bool*)

Specifies whether residuals should be added to account for parameter dependent sigma.

If set to true, additional residuals of the form  $\sqrt{\log(\sigma) + C}$  will be added. This enables least-squares optimization for variables with Gaussian noise assumption and parameter dependent standard deviation sigma. The constant *C* can be set via `setMinimumSigmaResiduals()`.

**Parameters** *sigma\_res* (*bool*) – if true, additional residuals are added

**setAllStatesNonNegative()**

Set flags indicating that all states should be treated as non-negative.

**setAlwaysCheckFinite**(*alwaysCheck*: *bool*)

Set whether the result of every call to `Model::f*` should be checked for finiteness.

**Parameters** *alwaysCheck* (*bool*) –

**setFixedParameterById**(*par\_id*: *str*, *value*: *float*)

Set value of first fixed parameter with the specified ID.

**Parameters**

- **par\_id** (*str*) – Fixed parameter id
- **value** (*float*) – Fixed parameter value

**setFixedParameterByName**(*par\_name*: *str*, *value*: *float*)

Set value of first fixed parameter with the specified name.

**Parameters**

- **par\_name** (*str*) – Fixed parameter ID
- **value** (*float*) – Fixed parameter value

**setFixedParameters**(*k*: *Sequence*[*float*])

Set values for constants.

**Parameters** **k** (*typing.Sequence*[*float*]) – Vector of fixed parameters

**setFixedParametersByIdRegex**(*par\_id\_regex*: *str*, *value*: *float*) → *int*

Set values of all fixed parameters with the ID matching the specified regex.

**Parameters**

- **par\_id\_regex** (*str*) – Fixed parameter name regex
- **value** (*float*) – Fixed parameter value

**Return type** *int*

**Returns** Number of fixed parameter IDs that matched the regex

**setFixedParametersByNameRegex**(*par\_name\_regex*: *str*, *value*: *float*) → *int*

Set value of all fixed parameters with name matching the specified regex.

**Parameters**

- **par\_name\_regex** (*str*) – Fixed parameter name regex
- **value** (*float*) – Fixed parameter value

**Return type** *int*

**Returns** Number of fixed parameter names that matched the regex

**setInitialStateSensitivities**(*sx0*: *Sequence*[*float*])

Set the initial state sensitivities.

**Parameters** **sx0** (*typing.Sequence*[*float*]) – vector of initial state sensitivities with chainrule applied. This could be a slice of `ReturnData::sx` or `ReturnData::sx0`

**setInitialStates**(*x0*: *Sequence*[*float*])

Set the initial states.

**Parameters** **x0** (*typing.Sequence*[*float*]) – Initial state vector

**setMinimumSigmaResiduals**(*min\_sigma*: *float*)

Sets the estimated lower boundary for `sigma_y`. When `setAddSigmaResiduals()` is activated, this lower boundary must ensure that  $\log(\text{sigma}) + \text{min\_sigma} > 0$ .

**Parameters** **min\_sigma** (*float*) – lower boundary



**setMaxEvent**(*nmaxevent*: *int*)

Set maximum number of events that may occur for each type.

**Parameters** **nmaxevent** (*int*) – Maximum number of events that may occur for each type

**setParameterById**(\*args)

*Overload 1:*

Set model parameters according to the parameter IDs and mapped values.

**Parameters**

- **p** (*StringDoubleMap*) – Map of parameters IDs and values
- **ignoreErrors** (*boolean*, *optional*) – Ignore errors such as parameter IDs in p which are not model parameters

*Overload 2:*

Set value of first model parameter with the specified ID.

**Parameters**

- **par\_id** (*str*) – Parameter ID
- **value** (*float*) – Parameter value

**setParameterByName**(\*args)

*Overload 1:*

Set value of first model parameter with the specified name.

**Parameters**

- **par\_name** (*str*) – Parameter name
- **value** (*float*) – Parameter value

*Overload 2:*

Set model parameters according to the parameter name and mapped values.

**Parameters**

- **p** (*StringDoubleMap*) – Map of parameters names and values
- **ignoreErrors** (*boolean*, *optional*) – Ignore errors such as parameter names in p which are not model parameters

*Overload 3:*

Set model parameters according to the parameter name and mapped values.

**Parameters**

- **p** (`StringDoubleMap`) – Map of parameters names and values
- **ignoreErrors** – Ignore errors such as parameter names in p which are not model parameters

**setParameterList** (*plist*: `Sequence[int]`)

Set the list of parameters for which sensitivities are to be computed.

NOTE: Resets initial state sensitivities.

**Parameters** **plist** (`typing.Sequence[int]`) – List of parameter indices

**setParameterScale** (\*args)

**setParameters** (p: `Sequence[float]`)

Set the parameter vector.

**Parameters** **p** (`typing.Sequence[float]`) – Vector of parameters

**setParametersByIdRegex** (par\_id\_regex: `str`, value: `float`) → `int`

Set all values of model parameters with IDs matching the specified regular expression.

**Parameters**

- **par\_id\_regex** (`str`) – Parameter ID regex
- **value** (`float`) – Parameter value

**Return type** `int`

**Returns** Number of parameter IDs that matched the regex

**setParametersByNameRegex** (par\_name\_regex: `str`, value: `float`) → `int`

Set all values of all model parameters with names matching the specified regex.

**Parameters**

- **par\_name\_regex** (`str`) – Parameter name regex
- **value** (`float`) – Parameter value

**Return type** `int`

**Returns** Number of fixed parameter names that matched the regex

**setReinitializationStateIdxs** (idxs: `Sequence[int]`)

Set indices of states to be reinitialized based on provided constants / fixed parameters

**Parameters** **idxs** (`typing.Sequence[int]`) – Array of state indices

**setReinitializeFixedParameterInitialStates** (flag: `bool`)

Set whether initial states depending on fixed parameters are to be reinitialized after preequilibration and presimulation.

**Parameters** **flag** (`bool`) – Fixed parameters reinitialized?

**setStateIsNonNegative** (stateIsNonNegative: `amici.amici.BoolVector`)

Set flags indicating whether states should be treated as non-negative.

**Parameters** **stateIsNonNegative** (`amici.amici.BoolVector`) – Vector of flags

**setSteadyStateComputationMode**(*mode*: `SteadyStateComputationMode`)

Set the mode how steady state is computed in the steadystate simulation.

**Parameters** *mode* (`amici.amici.SteadyStateComputationMode`) – Steadystate computation mode

**setSteadyStateSensitivityMode**(*mode*: `amici.amici.SteadyStateSensitivityMode`)

Set the mode how sensitivities are computed in the steadystate simulation.

**Parameters** *mode* (`amici.amici.SteadyStateSensitivityMode`) – Steadystate sensitivity mode

**setT0**(*t0*: `float`)

Set simulation start time.

**Parameters** *t0* (`float`) – Simulation start time

**setTimepoints**(*ts*: `Sequence[float]`)

Set the timepoint vector.

**Parameters** *ts* (`typing.Sequence[float]`) – New timepoint vector

**setUnscaledInitialStateSensitivities**(*sx0*: `Sequence[float]`)

Set the initial state sensitivities.

**Parameters** *sx0* (`typing.Sequence[float]`) – Vector of initial state sensitivities without chainrule applied. This could be the readin from a `model.sx0data` saved to HDF5.

**t0**() → `float`

Get simulation start time.

**Return type** `float`

**Returns** Simulation start time

**property** *ubw*

Upper bandwidth of the Jacobian

**class** `amici.amici.ModelDimensions(*args)`

Container for model dimensions.

Holds number of states, observables, etc.

**\_\_init\_\_**(\*args)

**Overload 1:** Default ctor

*Overload 2:*

Constructor with model dimensions

**Parameters**

- **nx\_rdata** (`int`) – Number of state variables
- **nxtrue\_rdata** (`int`) – Number of state variables of the non-augmented model
- **nx\_solver** (`int`) – Number of state variables with conservation laws applied

- **nxtrue\_solver** (*int*) – Number of state variables of the non-augmented model with conservation laws applied
- **nx\_solver\_reinit** (*int*) – Number of state variables with conservation laws subject to reinitialization
- **np** (*int*) – Number of parameters
- **nk** (*int*) – Number of constants
- **ny** (*int*) – Number of observables
- **nytrue** (*int*) – Number of observables of the non-augmented model
- **nz** (*int*) – Number of event observables
- **nztrue** (*int*) – Number of event observables of the non-augmented model
- **ne** (*int*) – Number of events
- **nspl** (*int*) – Number of splines
- **nJ** (*int*) – Number of objective functions
- **nw** (*int*) – Number of repeating elements
- **ndwdx** (*int*) – Number of nonzero elements in the  $x$  derivative of the repeating elements
- **ndwdp** (*int*) – Number of nonzero elements in the  $p$  derivative of the repeating elements
- **ndwdw** (*int*) – Number of nonzero elements in the  $w$  derivative of the repeating elements
- **ndxdotdw** (*int*) – Number of nonzero elements in the  $w$  derivative of  $x_{dot}$
- **ndJydy** (*IntVector*) – Number of nonzero elements in the  $y$  derivative of  $dJy$  (shape *nytrue*)
- **ndxrdatadxsolver** (*int*) – Number of nonzero elements in the  $x$  derivative of  $x_rdata$
- **ndxrdatadtcl** (*int*) – Number of nonzero elements in the  $tcl$  derivative of  $x_rdata$
- **ndtotal\_cldx\_rdata** (*int*) – Number of nonzero elements in the  $x_rdata$  derivative of  $total_{cl}$
- **nnz** (*int*) – Number of nonzero elements in Jacobian
- **ubw** (*int*) – Upper matrix bandwidth in the Jacobian
- **lbw** (*int*) – Lower matrix bandwidth in the Jacobian

**property lbw**

Lower bandwidth of the Jacobian

**property nJ**

Dimension of the augmented objective function for 2nd order ASA

**property ndJydy**

Number of nonzero elements in the  $y$  derivative of  $dJy$  (dimension *nytrue*)

**property ndtotal\_cldx\_rdata**

Number of nonzero elements in the  $x_rdata$  derivative of  $total_{cl}$

**property ndwdp**

Number of nonzero elements in the  $p$  derivative of the repeating elements

**property ndwdw**

Number of nonzero elements in the  $w$  derivative of the repeating elements

**property ndwdx**

Number of nonzero elements in the  $x$  derivative of the repeating elements

**property ndxdotdw**

Number of nonzero elements in the  $w$  derivative of  $\dot{x}$

**property ndxrdatadtcl**

Number of nonzero elements in the  $tcl$  derivative of  $x_rdata$

**property ndxrdatadxsolver**

Number of nonzero elements in the  $x$  derivative of  $x_rdata$

**property ne**

Number of events

**property nk**

Number of constants

**property nnz**

Number of nonzero entries in Jacobian

**property np**

Number of parameters

**property nspl**

number of spline functions in the model

**property nw**

Number of common expressions

**property nx\_rdata**

Number of states

**property nx\_solver**

Number of states with conservation laws applied

**property nx\_solver\_reinit**

Number of solver states subject to reinitialization

**property nxtrue\_rdata**

Number of states in the unaugmented system

**property nxtrue\_solver**

Number of states in the unaugmented system with conservation laws applied

**property ny**

Number of observables

**property nytrue**

Number of observables in the unaugmented system

**property nz**

Number of event outputs

**property nztrue**

Number of event outputs in the unaugmented system

**property ubw**

Upper bandwidth of the Jacobian

**class** amici.amici.**ModelPtr**(\*args)

Swig-Generated class that implements smart pointers to Model as objects.

**property idlist**

Flag array for DAE equations

**property lbw**

Lower bandwidth of the Jacobian

**property logger**

Logger

**property nJ**

Dimension of the augmented objective function for 2nd order ASA

**property ndJydy**

Number of nonzero elements in the  $y$  derivative of  $dJy$  (dimension  $nytrue$ )

**property ndtotal\_cldx\_rdata**

Number of nonzero elements in the  $x_rdata$  derivative of  $total_{cl}$

**property ndwdp**

Number of nonzero elements in the  $p$  derivative of the repeating elements

**property ndwdw**

Number of nonzero elements in the  $w$  derivative of the repeating elements

**property ndwdx**

Number of nonzero elements in the  $x$  derivative of the repeating elements

**property ndxdotdw**

Number of nonzero elements in the  $w$  derivative of  $xdot$

**property ndxrdatadtcl**

Number of nonzero elements in the  $tcl$  derivative of  $x_rdata$

**property ndxrdatadxsolver**

Number of nonzero elements in the  $x$  derivative of  $x_rdata$

**property ne**

Number of events

**property nnz**

Number of nonzero entries in Jacobian

**property nspl**

number of spline functions in the model

**property nw**

Number of common expressions

**property nx\_rdata**

Number of states

**property nx\_solver**

Number of states with conservation laws applied

**property nx\_solver\_reinit**

Number of solver states subject to reinitialization

**property nxtrue\_rdata**

Number of states in the unaugmented system

**property nxtrue\_solver**

Number of states in the unaugmented system with conservation laws applied

**property ny**

Number of observables

**property nytrue**

Number of observables in the unaugmented system

**property nz**

Number of event outputs

**property nztrue**

Number of event outputs in the unaugmented system

**property o2mode**

Flag indicating whether for *amici::Solver::sensi\_ == amici::SensitivityOrder::second* directional or full second order derivative will be computed

**property pythonGenerated**

Flag indicating Matlab- or Python-based model generation

**property ubw**

Upper bandwidth of the Jacobian

```
class amici.amici.NewtonDampingFactorMode(value)
```

An enumeration.

```
off = 0
```

```
on = 1
```

```
class amici.amici.NonlinearSolverIteration(value)
```

An enumeration.

```
fixedpoint = 1
```

```
functional = 1
```

```
newton = 2
```

```
amici.amici.NonlinearSolverIteration_fixedpoint = 1
```

deprecated

```
class amici.amici.ObservableScaling(value)
```

An enumeration.

```
lin = 0
```

```
log = 1
```

```
log10 = 2
```

```
class amici.amici.ParameterScaling(value)
```

An enumeration.

```
ln = 1
```

```
log10 = 2
```

```
none = 0
```

```
class amici.amici.ParameterScalingVector(*args)
```

```
class amici.amici.RDataReporting(value)
```

An enumeration.

```
full = 0
```

```
likelihood = 2
```

```
residuals = 1
```

```
class amici.amici.ReturnData(*args)
```

Stores all data to be returned by `amici.amici.runAmiciSimulation()`.

NOTE: multi-dimensional arrays are stored in row-major order (C-style)

**property FIM**

fisher information matrix (shape *nplist* x *nplist*, row-major)

**property J**

Jacobian of differential equation right hand side (shape *nx* x *nx*, row-major)

```
__init__(*args)
```

*Overload 1:*

Default constructor

*Overload 2:*

Constructor

#### Parameters

- **ts** ([DoubleVector](#)) – see `amici::SimulationParameters::ts`
- **model\_dimensions** ([ModelDimensions](#)) – Model dimensions
- **nplist** (*int*) – see `amici::ModelDimensions::nplist`
- **nmaxevent** (*int*) – see `amici::ModelDimensions::nmaxevent`
- **nt** (*int*) – see `amici::ModelDimensions::nt`
- **newton\_maxsteps** (*int*) – see `amici::Solver::newton_maxsteps`
- **pscale** ([ParameterScalingVector](#)) – see `amici::SimulationParameters::pscale`
- **o2mode** (*int*) – see `amici::SimulationParameters::o2mode`



- **sensi** (*SensitivityOrder*) – see `amici::Solver::sensi`
- **sensi\_meth** (*SensitivityMethod*) – see `amici::Solver::sensi_meth`
- **rdrn** (*RDataReporting*) – see `amici::Solver::rdata_reporting`
- **quadratic\_llh** (*boolean*) – whether model defines a quadratic nllh and computing res, sres and FIM makes sense
- **sigma\_res** (*boolean*) – indicates whether additional residuals are to be added for each sigma
- **sigma\_offset** (*float*) – offset to ensure real-valuedness of sigma residuals

*Overload 3:*

constructor that uses information from model and solver to appropriately initialize fields

#### Parameters

- **solver** (*Solver*) – solver instance
- **model** (*Model*) – model instance

#### property **chi2**

$\chi^2$  value

#### property **cpu\_time**

computation time of forward solve [ms]

**Warning:** If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

#### property **cpu\_timeB**

computation time of backward solve [ms]

**Warning:** If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

#### property **cpu\_time\_total**

total CPU time from entering `runAmiciSimulation` until exiting [ms]

**Warning:** If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

#### property **id**

Arbitrary (not necessarily unique) identifier.

#### property **lbw**

Lower bandwidth of the Jacobian

**property llh**

log-likelihood value

**property messages**

log messages

**property nJ**

Dimension of the augmented objective function for 2nd order ASA

**property ndJydy**

Number of nonzero elements in the  $y$  derivative of  $dJy$  (dimension  $nytrue$ )

**property ndtotal\_cldx\_rdata**

Number of nonzero elements in the  $x_rdata$  derivative of  $total_{cl}$

**property ndwdp**

Number of nonzero elements in the  $p$  derivative of the repeating elements

**property ndwdw**

Number of nonzero elements in the  $w$  derivative of the repeating elements

**property ndwdx**

Number of nonzero elements in the  $x$  derivative of the repeating elements

**property ndxdotdw**

Number of nonzero elements in the  $w$  derivative of  $xdot$

**property ndxrdatadtcl**

Number of nonzero elements in the  $tcl$  derivative of  $x_rdata$

**property ndxrdatadxsolver**

Number of nonzero elements in the  $x$  derivative of  $x_rdata$

**property ne**

Number of events

**property newton\_maxsteps**

maximal number of newton iterations for steady state calculation

**property nk**

Number of constants

**property nmaxevent**

maximal number of occurring events (for every event type)

**property nnz**

Number of nonzero entries in Jacobian

**property np**

Number of parameters

**property nplist**

number of parameter for which sensitivities were requested

**property nspl**

numer of spline functions in the model

**property nt**  
 number of considered timepoints

**property numerrtestfails**  
 number of error test failures forward problem (shape *nt*)

**property numerrtestfailsB**  
 number of error test failures backward problem (shape *nt*)

**property numnonlinsolvconvfails**  
 number of linear solver convergence failures forward problem (shape *nt*)

**property numnonlinsolvconvfailsB**  
 number of linear solver convergence failures backward problem (shape *nt*)

**property numrhsevals**  
 number of right hand side evaluations forward problem (shape *nt*)

**property numrhsevalsB**  
 number of right hand side evaluations backward problem (shape *nt*)

**property numsteps**  
 number of integration steps forward problem (shape *nt*)

**property numstepsB**  
 number of integration steps backward problem (shape *nt*)

**property nw**  
 Number of common expressions

**property nx**  
 number of states (alias *nx\_rdata*, kept for backward compatibility)

**property nx\_rdata**  
 Number of states

**property nx\_solver**  
 Number of states with conservation laws applied

**property nx\_solver\_reinit**  
 Number of solver states subject to reinitialization

**property nxtrue**  
 number of states in the unaugmented system (alias *nxtrue\_rdata*, kept for backward compatibility)

**property nxtrue\_rdata**  
 Number of states in the unaugmented system

**property nxtrue\_solver**  
 Number of states in the unaugmented system with conservation laws applied

**property ny**  
 Number of observables

**property nytrue**  
 Number of observables in the unaugmented system

**property nz**

Number of event outputs

**property nztrue**

Number of event outputs in the unaugmented system

**property o2mode**

flag indicating whether second-order sensitivities were requested

**property order**

employed order forward problem (shape *nt*)

**property posteq\_cpu\_time**

computation time of the steady state solver [ms] (postequilibration)

**Warning:** If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

**property posteq\_cpu\_timeB**

computation time of the steady state solver of the backward problem [ms] (postequilibration)

**Warning:** If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

**property posteq\_numsteps**

number of Newton steps for steady state problem (preequilibration) [newton, simulation, newton] (shape 3) (postequilibration)

**property posteq\_numstepsB**

number of simulation steps for adjoint steady state problem (postequilibration) [== 0 if analytical solution worked, > 0 otherwise]

**property posteq\_status**

flags indicating success of steady state solver (postequilibration)

**property posteq\_t**

time when steadystate was reached via simulation (postequilibration)

**property posteq\_wrms**

weighted root-mean-square of the rhs when steadystate was reached (postequilibration)

**property preeq\_cpu\_time**

computation time of the steady state solver [ms] (preequilibration)

**Warning:** If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

**property preeq\_cpu\_timeB**

computation time of the steady state solver of the backward problem [ms] (preequilibration)

**Warning:** If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

**property preeq\_numsteps**

number of Newton steps for steady state problem (preequilibration) [newton, simulation, newton] (length = 3)

**property preeq\_numstepsB**

number of simulation steps for adjoint steady state problem (preequilibration) [== 0 if analytical solution worked, > 0 otherwise]

**property preeq\_status**

flags indicating success of steady state solver (preequilibration)

**property preeq\_t**

time when steadystate was reached via simulation (preequilibration)

**property preeq\_wrms**

weighted root-mean-square of the rhs when steadystate was reached (preequilibration)

**property pscale**

scaling of parameterization

**property rdata\_reporting**

reporting mode

**property res**

observable (shape  $nt \times ny$ , row-major)

**property rz**

event trigger output (shape  $nmaxevent \times nz$ , row-major)

**property s2llh**

second-order parameter derivative of log-likelihood (shape  $nJ-1 \times nplist$ , row-major)

**property s2rz**

second-order parameter derivative of event trigger output (shape  $nmaxevent \times nztrue \times nplist \times nplist$ , row-major)

**property sensi**

sensitivity order

**property sensi\_meth**

sensitivity method

**property sigma\_res**

boolean indicating whether residuals for standard deviations have been added

**property sigmay**

observable standard deviation (shape  $nt \times ny$ , row-major)

**property sigmaz**

event output sigma standard deviation (shape  $nmaxevent \times nz$ , row-major)

**property sllh**

parameter derivative of log-likelihood (shape  $nplist$ )

**property sres**

parameter derivative of residual (shape  $nt*ny \times nplist$ , row-major)

**property srz**

parameter derivative of event trigger output (shape  $nmaxevent \times nplist \times nz$ , row-major)

**property ssigmay**

parameter derivative of observable standard deviation (shape  $nt \times nplist \times ny$ , row-major)

**property ssigmaz**

parameter derivative of event output standard deviation (shape  $nmaxevent \times nplist \times nz$ , row-major)

**property status**

Simulation status code.

One of:

- AMICI\_SUCCESS, indicating successful simulation
- AMICI\_MAX\_TIME\_EXCEEDED, indicating that the simulation did not finish within the allowed time (see Solver.{set,get}MaxTime)
- AMICI\_ERROR, indicating that some error occurred during simulation (a more detailed error message will have been printed).
- AMICI\_NOT\_RUN, if no simulation was started

**property sx**

parameter derivative of state (shape  $nt \times nplist \times nx$ , row-major)

**property sx0**

initial sensitivities (shape  $nplist \times nx$ , row-major)

**property sx\_ss**

preequilibration sensitivities (shape  $nplist \times nx$ , row-major)

**property sy**

parameter derivative of observable (shape  $nt \times nplist \times ny$ , row-major)

**property sz**

parameter derivative of event output (shape  $nmaxevent \times nplist \times nz$ , row-major)

**property ts**

timepoints (shape  $nt$ )

**property ubw**

Upper bandwidth of the Jacobian

**property w**

w data from the model (recurring terms in `xdot`, for imported SBML models from python, this contains the flux vector) (shape  $nt \times nw$ , row major)

**property x**

state (shape  $nt \times nx$ , row-major)

**property x0**

initial state (shape  $nx$ )

**property x\_ss**

preequilibration steady state (shape  $nx$ )

**property xdot**time derivative (shape  $nx$ )**property y**observable (shape  $nt \times ny$ , row-major)**property z**event output (shape  $nmaxevent \times nz$ , row-major)**class amici.amici.ReturnDataPtr(\*args)**

Swig-Generated class that implements smart pointers to ReturnData as objects.

**property FIM**fisher information matrix (shape  $nplist \times nplist$ , row-major)**property J**Jacobian of differential equation right hand side (shape  $nx \times nx$ , row-major)**property chi2** $\chi^2$  value**property cpu\_time**

computation time of forward solve [ms]

**Warning:** If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

**property cpu\_timeB**

computation time of backward solve [ms]

**Warning:** If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

**property cpu\_time\_total**

total CPU time from entering runAmiciSimulation until exiting [ms]

**Warning:** If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

**property id**

Arbitrary (not necessarily unique) identifier.

**property lbw**

Lower bandwidth of the Jacobian

**property llh**

log-likelihood value

**property messages**

log messages

**property nj**

Dimension of the augmented objective function for 2nd order ASA

**property ndJydy**

Number of nonzero elements in the  $y$  derivative of  $dJy$  (dimension  $nytrue$ )

**property ndtotal\_cldx\_rdata**

Number of nonzero elements in the  $x_rdata$  derivative of  $total_{cl}$

**property ndwdp**

Number of nonzero elements in the  $p$  derivative of the repeating elements

**property ndwdw**

Number of nonzero elements in the  $w$  derivative of the repeating elements

**property ndwdx**

Number of nonzero elements in the  $x$  derivative of the repeating elements

**property ndxdotdw**

Number of nonzero elements in the  $w$  derivative of  $xdot$

**property ndxrdatadtcl**

Number of nonzero elements in the  $tcl$  derivative of  $x_rdata$

**property ndxrdatadxsolver**

Number of nonzero elements in the  $x$  derivative of  $x_rdata$

**property ne**

Number of events

**property newton\_maxsteps**

maximal number of newton iterations for steady state calculation

**property nk**

Number of constants

**property nmaxevent**

maximal number of occurring events (for every event type)

**property nnz**

Number of nonzero entries in Jacobian

**property np**

Number of parameters

**property nplist**

number of parameter for which sensitivities were requested

**property nspl**

number of spline functions in the model

**property nt**

number of considered timepoints

**property numerrtestfails**

number of error test failures forward problem (shape  $nt$ )



**property numerrtestfailsB**

number of error test failures backward problem (shape *nt*)

**property numnonlinsolvconvfails**

number of linear solver convergence failures forward problem (shape *nt*)

**property numnonlinsolvconvfailsB**

number of linear solver convergence failures backward problem (shape *nt*)

**property numrhsevals**

number of right hand side evaluations forward problem (shape *nt*)

**property numrhsevalsB**

number of right hand side evaluations backward problem (shape *nt*)

**property numsteps**

number of integration steps forward problem (shape *nt*)

**property numstepsB**

number of integration steps backward problem (shape *nt*)

**property nw**

Number of common expressions

**property nx**

number of states (alias *nx\_rdata*, kept for backward compatibility)

**property nx\_rdata**

Number of states

**property nx\_solver**

Number of states with conservation laws applied

**property nx\_solver\_reinit**

Number of solver states subject to reinitialization

**property nxtrue**

number of states in the unaugmented system (alias *nxtrue\_rdata*, kept for backward compatibility)

**property nxtrue\_rdata**

Number of states in the unaugmented system

**property nxtrue\_solver**

Number of states in the unaugmented system with conservation laws applied

**property ny**

Number of observables

**property nytrue**

Number of observables in the unaugmented system

**property nz**

Number of event outputs

**property nztrue**

Number of event outputs in the unaugmented system

**property o2mode**

flag indicating whether second-order sensitivities were requested

**property order**

employed order forward problem (shape *nt*)

**property posteq\_cpu\_time**

computation time of the steady state solver [ms] (postequilibration)

**Warning:** If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

**property posteq\_cpu\_timeB**

computation time of the steady state solver of the backward problem [ms] (postequilibration)

**Warning:** If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

**property posteq\_numsteps**

number of Newton steps for steady state problem (preequilibration) [newton, simulation, newton] (shape 3) (postequilibration)

**property posteq\_numstepsB**

number of simulation steps for adjoint steady state problem (postequilibration) [== 0 if analytical solution worked, > 0 otherwise]

**property posteq\_status**

flags indicating success of steady state solver (postequilibration)

**property posteq\_t**

time when steadystate was reached via simulation (postequilibration)

**property posteq\_wrms**

weighted root-mean-square of the rhs when steadystate was reached (postequilibration)

**property preeq\_cpu\_time**

computation time of the steady state solver [ms] (preequilibration)

**Warning:** If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

**property preeq\_cpu\_timeB**

computation time of the steady state solver of the backward problem [ms] (preequilibration)

**Warning:** If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

**property preeq\_numsteps**

number of Newton steps for steady state problem (preequilibration) [newton, simulation, newton] (length = 3)

**property preeq\_numstepsB**

number of simulation steps for adjoint steady state problem (preequilibration) [== 0 if analytical solution worked, > 0 otherwise]

**property preeq\_status**

flags indicating success of steady state solver (preequilibration)

**property preeq\_t**

time when steadystate was reached via simulation (preequilibration)

**property preeq\_wrms**

weighted root-mean-square of the rhs when steadystate was reached (preequilibration)

**property pscale**

scaling of parameterization

**property rdata\_reporting**

reporting mode

**property res**

observable (shape  $nt \times ny$ , row-major)

**property rz**

event trigger output (shape  $nmaxevent \times nz$ , row-major)

**property s2llh**

second-order parameter derivative of log-likelihood (shape  $nJ-1 \times nplist$ , row-major)

**property s2rz**

second-order parameter derivative of event trigger output (shape  $nmaxevent \times nztrue \times nplist \times nplist$ , row-major)

**property sensi**

sensitivity order

**property sensi\_meth**

sensitivity method

**property sigma\_res**

boolean indicating whether residuals for standard deviations have been added

**property sigmay**

observable standard deviation (shape  $nt \times ny$ , row-major)

**property sigmaz**

event output sigma standard deviation (shape  $nmaxevent \times nz$ , row-major)

**property sllh**

parameter derivative of log-likelihood (shape  $nplist$ )

**property sres**

parameter derivative of residual (shape  $nt \times ny \times nplist$ , row-major)

**property srz**

parameter derivative of event trigger output (shape  $n_{\text{maxevent}} \times n_{\text{plist}} \times n_z$ , row-major)

**property ssgmay**

parameter derivative of observable standard deviation (shape  $n_t \times n_{\text{plist}} \times n_y$ , row-major)

**property ssgmaz**

parameter derivative of event output standard deviation (shape  $n_{\text{maxevent}} \times n_{\text{plist}} \times n_z$ , row-major)

**property status**

Simulation status code.

One of:

- AMICI\_SUCCESS, indicating successful simulation
- AMICI\_MAX\_TIME\_EXCEEDED, indicating that the simulation did not finish within the allowed time (see `Solver.{set,get}MaxTime`)
- AMICI\_ERROR, indicating that some error occurred during simulation (a more detailed error message will have been printed).
- AMICI\_NOT\_RUN, if no simulation was started

**property sx**

parameter derivative of state (shape  $n_t \times n_{\text{plist}} \times n_x$ , row-major)

**property sx0**

initial sensitivities (shape  $n_{\text{plist}} \times n_x$ , row-major)

**property sx\_ss**

preequilibration sensitivities (shape  $n_{\text{plist}} \times n_x$ , row-major)

**property sy**

parameter derivative of observable (shape  $n_t \times n_{\text{plist}} \times n_y$ , row-major)

**property sz**

parameter derivative of event output (shape  $n_{\text{maxevent}} \times n_{\text{plist}} \times n_z$ , row-major)

**property ts**

timepoints (shape  $n_t$ )

**property ubw**

Upper bandwidth of the Jacobian

**property w**

w data from the model (recurring terms in `xdot`, for imported SBML models from python, this contains the flux vector) (shape  $n_t \times n_w$ , row major)

**property x**

state (shape  $n_t \times n_x$ , row-major)

**property x0**

initial state (shape  $n_x$ )

**property x\_ss**

preequilibration steady state (shape  $n_x$ )

**property xdot**

time derivative (shape  $n_x$ )

```

    property y
        observable (shape nt x ny, row-major)

    property z
        event output (shape nmaxevent x nz, row-major)
class amici.amici.SecondOrderMode(value)
    An enumeration.

    directional = 2

    full = 1

    none = 0

class amici.amici.SensitivityMethod(value)
    An enumeration.

    adjoint = 2

    forward = 1

    none = 0

amici.amici.SensitivityMethod_adjoint = 2
    Adjoint sensitivity analysis.

amici.amici.SensitivityMethod_forward = 1
    Forward sensitivity analysis.

amici.amici.SensitivityMethod_none = 0
    Don't compute sensitivities.

class amici.amici.SensitivityOrder(value)
    An enumeration.

    first = 1

    none = 0

    second = 2

amici.amici.SensitivityOrder_first = 1
    First-order sensitivities.

amici.amici.SensitivityOrder_none = 0
    Don't compute sensitivities.

amici.amici.SensitivityOrder_second = 2
    Second-order sensitivities.

class amici.amici.SimulationParameters(*args)
    Container for various simulation parameters.

    __init__(*args)
        Overload 1:
        Constructor

        Parameters timepoints (DoubleVector) – Timepoints for which simulation results are requested

```

*Overload 2:*

Constructor

**Parameters**

- **fixedParameters** ([DoubleVector](#)) – Model constants
- **parameters** ([DoubleVector](#)) – Model parameters

*Overload 3:*

Constructor

**Parameters**

- **fixedParameters** ([DoubleVector](#)) – Model constants
- **parameters** ([DoubleVector](#)) – Model parameters
- **plist** ([IntVector](#)) – Model parameter indices w.r.t. which sensitivities are to be computed

*Overload 4:*

Constructor

**Parameters**

- **timepoints** ([DoubleVector](#)) – Timepoints for which simulation results are requested
- **fixedParameters** ([DoubleVector](#)) – Model constants
- **parameters** ([DoubleVector](#)) – Model parameters

**property fixedParameters**

Model constants

Vector of size `Model::nk()` or empty

**property fixedParametersPreequilibration**

Model constants for pre-equilibration

Vector of size `Model::nk()` or empty.

**property fixedParametersPresimulation**

Model constants for pre-simulation

Vector of size `Model::nk()` or empty.

**property parameters**

Model parameters

Vector of size `Model::np()` or empty with parameter scaled according to `SimulationParameter::pscale`.

**property plist**

Parameter indices w.r.t. which to compute sensitivities

**property pscale**

Parameter scales

Vector of parameter scale of size `Model::np()`, indicating how/if each parameter is to be scaled.

**property reinitialization\_state\_idxes\_presim**

Indices of states to be reinitialized based on provided presimulation constants / fixed parameters.

**property reinitialization\_state\_idxes\_sim**

Indices of states to be reinitialized based on provided constants / fixed parameters.

**reinitializeAllFixedParameterDependentInitialStates(*nx\_rdata: int*)**

Set reinitialization of all states based on model constants for all simulation phases.

Convenience function to populate *reinitialization\_state\_idxes\_presim* and *reinitialization\_state\_idxes\_sim*

**Parameters** *nx\_rdata* (*int*) – Number of states (`Model::nx_rdata`)

**reinitializeAllFixedParameterDependentInitialStatesForPresimulation(*nx\_rdata: int*)**

Set reinitialization of all states based on model constants for presimulation (only meaningful if preequilibration is performed).

Convenience function to populate *reinitialization\_state\_idxes\_presim* and *reinitialization\_state\_idxes\_sim*

**Parameters** *nx\_rdata* (*int*) – Number of states (`Model::nx_rdata`)

**reinitializeAllFixedParameterDependentInitialStatesForSimulation(*nx\_rdata: int*)**

Set reinitialization of all states based on model constants for the ‘main’ simulation (only meaningful if presimulation or preequilibration is performed).

Convenience function to populate *reinitialization\_state\_idxes\_presim* and *reinitialization\_state\_idxes\_sim*

**Parameters** *nx\_rdata* (*int*) – Number of states (`Model::nx_rdata`)

**property reinitializeFixedParameterInitialStates**

Flag indicating whether reinitialization of states depending on fixed parameters is activated

**property sx0**

Initial state sensitivities

Dimensions: `Model::nx() * Model::nplist()`, `Model::nx() * ExpData::plist.size()`, if `ExpData::plist` is not empty, or empty

**property t\_presim**

Duration of pre-simulation.

If this is > 0, presimulation will be performed from (`model->t0 - t_presim`) to `model->t0` using the fixed-Parameters in `fixedParametersPresimulation`

**property ts\_**

Timepoints for which model state/outputs/... are requested

Vector of timepoints.

**property tstart\_**

starting time

**property x0**

Initial state

Vector of size `Model::nx()` or empty

**class amici.amici.Solver(\*args, \*\*kwargs)**

The Solver class provides a generic interface to CVODES and IDAS solvers, individual realizations are realized in the `CVodeSolver` and the `IDASolver` class. All transient private/protected members (CVODES/IDAS memory, interface variables and status flags) are specified as mutable and not included in serialization or equality checks. No solver setting parameter should be marked mutable.

NOTE: Any changes in data members here must be propagated to copy ctor, equality operator, serialization functions in `serialization.h`, and `amici::hdf5::(read/write)SolverSettings(From/To)HDF5` in `hdf5.cpp`.

**\_\_init\_\_(\*args, \*\*kwargs)**

**clone()** → *amici.amici.Solver*

Clone this instance

**Return type** *Solver*

**Returns** The clone

**computingASA()** → *bool*

check if ASA is being computed

**Return type** *boolean*

**Returns** flag

**computingFSA()** → *bool*

check if FSA is being computed

**Return type** *boolean*

**Returns** flag

**getAbsoluteTolerance()** → *float*

Get the absolute tolerances for the forward problem

Same tolerance is used for the backward problem if not specified differently via `setAbsoluteToleranceASA`.

**Return type** *float*

**Returns** absolute tolerances

**getAbsoluteToleranceB()** → *float*

Returns the absolute tolerances for the backward problem for adjoint sensitivity analysis

**Return type** *float*

**Returns** absolute tolerances

**getAbsoluteToleranceFSA()** → *float*

Returns the absolute tolerances for the forward sensitivity problem

**Return type** *float*

**Returns** absolute tolerances



**getAbsoluteToleranceQuadratures()** → float  
 returns the absolute tolerance for the quadrature problem  
**Return type** float  
**Returns** absolute tolerance

**getAbsoluteToleranceSteadyState()** → float  
 returns the absolute tolerance for the steady state problem  
**Return type** float  
**Returns** absolute tolerance

**getAbsoluteToleranceSteadyStateSensi()** → float  
 returns the absolute tolerance for the sensitivities of the steady state problem  
**Return type** float  
**Returns** absolute tolerance

**getInternalSensitivityMethod()** → *amici.amici.InternalSensitivityMethod*  
 returns the internal sensitivity method  
**Return type** *InternalSensitivityMethod*  
**Returns** internal sensitivity method

**getInterpolationType()** → *amici.amici.InterpolationType*  
**Return type** *InterpolationType*  
**Returns**

**getLinearMultistepMethod()** → *amici.amici.LinearMultistepMethod*  
 returns the linear system multistep method  
**Return type** *LinearMultistepMethod*  
**Returns** linear system multistep method

**getLinearSolver()** → *amici.amici.LinearSolver*  
**Return type** *LinearSolver*  
**Returns**

**getMaxSteps()** → int  
 returns the maximum number of solver steps for the forward problem  
**Return type** int  
**Returns** maximum number of solver steps

**getMaxStepsBackwardProblem()** → int  
 returns the maximum number of solver steps for the backward problem  
**Return type** int  
**Returns** maximum number of solver steps

**getMaxTime()** → float  
 Returns the maximum time allowed for integration  
**Return type** float

**Returns** Time in seconds

**getNewtonDampingFactorLowerBound()** → float

Get a lower bound of the damping factor used in the Newton solver

**Return type** float

**Returns**

**getNewtonDampingFactorMode()** → *amici.amici.NewtonDampingFactorMode*

Get a state of the damping factor used in the Newton solver

**Return type** *NewtonDampingFactorMode*

**Returns**

**getNewtonMaxSteps()** → int

Get maximum number of allowed Newton steps for steady state computation

**Return type** int

**Returns**

**getNewtonStepSteadyStateCheck()** → bool

Returns how convergence checks for steadystate computation are performed. If activated, convergence checks are limited to every 25 steps in the simulation solver to limit performance impact.

**Return type** boolean

**Returns** boolean flag indicating newton step (true) or the right hand side (false)

**getNonlinearSolverIteration()** → *amici.amici.NonlinearSolverIteration*

returns the nonlinear system solution method

**Return type** *NonlinearSolverIteration*

**Returns**

**getRelativeTolerance()** → float

Get the relative tolerances for the forward problem

Same tolerance is used for the backward problem if not specified differently via setRelativeToleranceASA.

**Return type** float

**Returns** relative tolerances

**getRelativeToleranceB()** → float

Returns the relative tolerances for the adjoint sensitivity problem

**Return type** float

**Returns** relative tolerances

**getRelativeToleranceFSA()** → float

Returns the relative tolerances for the forward sensitivity problem

**Return type** float

**Returns** relative tolerances

**getRelativeToleranceQuadratures()** → float

Returns the relative tolerance for the quadrature problem

**Return type** float

**Returns** relative tolerance

**getRelativeToleranceSteadyState()** → float

returns the relative tolerance for the steady state problem

**Return type** float

**Returns** relative tolerance

**getRelativeToleranceSteadyStateSensi()** → float

returns the relative tolerance for the sensitivities of the steady state problem

**Return type** float

**Returns** relative tolerance

**getReturnDataReportingMode()** → *amici.amici.RDataReporting*

returns the ReturnData reporting mode

**Return type** *RDataReporting*

**Returns** ReturnData reporting mode

**getSensiSteadyStateCheck()** → bool

Returns how convergence checks for steadystate computation are performed.

**Return type** boolean

**Returns** boolean flag indicating state and sensitivity equations (true) or only state variables (false).

**getSensitivityMethod()** → *amici.amici.SensitivityMethod*

Return current sensitivity method

**Return type** *SensitivityMethod*

**Returns** method enum

**getSensitivityMethodPreequilibration()** → *amici.amici.SensitivityMethod*

Return current sensitivity method during preequilibration

**Return type** *SensitivityMethod*

**Returns** method enum

**getSensitivityOrder()** → *amici.amici.SensitivityOrder*

Get sensitivity order

**Return type** *SensitivityOrder*

**Returns** sensitivity order

**getStabilityLimitFlag()** → bool

returns stability limit detection mode

**Return type** boolean

**Returns** stldet can be false (deactivated) or true (activated)

**getStateOrdering()** → int

Gets KLU / SuperLUMT state ordering mode

**Return type** int

**Returns** State-ordering as integer according to `SUNLinSolKLU::StateOrdering` or `SUNLinSolSuperLUMT::StateOrdering` (which differ).

**getSteadyStateSensiToleranceFactor()** → `float`

returns the steady state sensitivity simulation tolerance factor.

Steady state sensitivity simulation tolerances are the product of the sensitivity simulation tolerances and this factor, unless manually set with `set(Absolute/Relative)ToleranceSteadyStateSensi()`.

**Return type** `float`

**Returns** steady state simulation tolerance factor

**getSteadyStateToleranceFactor()** → `float`

returns the steady state simulation tolerance factor.

Steady state simulation tolerances are the product of the simulation tolerances and this factor, unless manually set with `set(Absolute/Relative)ToleranceSteadyState()`.

**Return type** `float`

**Returns** steady state simulation tolerance factor

**property logger**

**nplist()** → `int`

number of parameters with which the solver was initialized

**Return type** `int`

**Returns** `sx.getLength()`

**nquad()** → `int`

number of quadratures with which the solver was initialized

**Return type** `int`

**Returns** `xQB.getLength()`

**nx()** → `int`

number of states with which the solver was initialized

**Return type** `int`

**Returns** `x.getLength()`

**setAbsoluteTolerance( atol: float )**

Sets the absolute tolerances for the forward problem

Same tolerance is used for the backward problem if not specified differently via `setAbsoluteToleranceASA`.

**Parameters** `atol` (`float`) – absolute tolerance (non-negative number)

**setAbsoluteToleranceB( atol: float )**

Sets the absolute tolerances for the backward problem for adjoint sensitivity analysis

**Parameters** `atol` (`float`) – absolute tolerance (non-negative number)

**setAbsoluteToleranceFSA( atol: float )**

Sets the absolute tolerances for the forward sensitivity problem

**Parameters** `atol` (`float`) – absolute tolerance (non-negative number)

**setAbsoluteToleranceQuadratures**(*atol*: *float*)

sets the absolute tolerance for the quadrature problem

**Parameters** *atol* (*float*) – absolute tolerance (non-negative number)

**setAbsoluteToleranceSteadyState**(*atol*: *float*)

sets the absolute tolerance for the steady state problem

**Parameters** *atol* (*float*) – absolute tolerance (non-negative number)

**setAbsoluteToleranceSteadyStateSensi**(*atol*: *float*)

sets the absolute tolerance for the sensitivities of the steady state problem

**Parameters** *atol* (*float*) – absolute tolerance (non-negative number)

**setInternalSensitivityMethod**(*ism*: *amici.amici.InternalSensitivityMethod*)

sets the internal sensitivity method

**Parameters** *ism* (*amici.amici.InternalSensitivityMethod*) – internal sensitivity method

**setInterpolationType**(*interpType*: *amici.amici.InterpolationType*)

sets the interpolation of the forward solution that is used for the backwards problem

**Parameters** *interpType* (*amici.amici.InterpolationType*) – interpolation type

**setLinearMultistepMethod**(*lmm*: *amici.amici.LinearMultistepMethod*)

sets the linear system multistep method

**Parameters** *lmm* (*amici.amici.LinearMultistepMethod*) – linear system multistep method

**setLinearSolver**(*linsol*: *amici.amici.LinearSolver*)

**Parameters** *linsol* (*amici.amici.LinearSolver*) –

**setMaxSteps**(*maxsteps*: *int*)

sets the maximum number of solver steps for the forward problem

**Parameters** *maxsteps* (*int*) – maximum number of solver steps (positive number)

**setMaxStepsBackwardProblem**(*maxsteps*: *int*)

sets the maximum number of solver steps for the backward problem

**Parameters** *maxsteps* (*int*) – maximum number of solver steps (non-negative number)

Notes: default behaviour (100 times the value for the forward problem) can be restored by passing *maxsteps*=0

**setMaxTime**(*maxtime*: *float*)

Set the maximum CPU time allowed for integration

**Parameters** *maxtime* (*float*) – Time in seconds. Zero means infinite time.

**setNewtonDampingFactorLowerBound**(*dampingFactorLowerBound*: *float*)

Set a lower bound of the damping factor in the Newton solver

**Parameters** *dampingFactorLowerBound* (*float*) –

**setNewtonDampingFactorMode**(*dampingFactorMode*: *amici.amici.NewtonDampingFactorMode*)

Turn on/off a damping factor in the Newton method

**Parameters** *dampingFactorMode* (*amici.amici.NewtonDampingFactorMode*) –

**setNewtonMaxSteps**(*newton\_maxsteps*: *int*)

Set maximum number of allowed Newton steps for steady state computation

**Parameters** **newton\_maxsteps** (*int*) –

**setNewtonStepSteadyStateCheck**(*flag*: *bool*)

Sets how convergence checks for steadystate computation are performed.

**Parameters** **flag** (*bool*) – boolean flag to pick newton step (true) or the right hand side (false, default)

**setNonlinearSolverIteration**(*iter*: *amici.amici.NonlinearSolverIteration*)

sets the nonlinear system solution method

**Parameters** **iter** (*amici.amici.NonlinearSolverIteration*) – nonlinear system solution method

**setRelativeTolerance**(*rtol*: *float*)

Sets the relative tolerances for the forward problem

Same tolerance is used for the backward problem if not specified differently via `setRelativeToleranceASA`.

**Parameters** **rtol** (*float*) – relative tolerance (non-negative number)

**setRelativeToleranceB**(*rtol*: *float*)

Sets the relative tolerances for the adjoint sensitivity problem

**Parameters** **rtol** (*float*) – relative tolerance (non-negative number)

**setRelativeToleranceFSA**(*rtol*: *float*)

Sets the relative tolerances for the forward sensitivity problem

**Parameters** **rtol** (*float*) – relative tolerance (non-negative number)

**setRelativeToleranceQuadratures**(*rtol*: *float*)

sets the relative tolerance for the quadrature problem

**Parameters** **rtol** (*float*) – relative tolerance (non-negative number)

**setRelativeToleranceSteadyState**(*rtol*: *float*)

sets the relative tolerance for the steady state problem

**Parameters** **rtol** (*float*) – relative tolerance (non-negative number)

**setRelativeToleranceSteadyStateSensi**(*rtol*: *float*)

sets the relative tolerance for the sensitivities of the steady state problem

**Parameters** **rtol** (*float*) – relative tolerance (non-negative number)

**setReturnDataReportingMode**(*rdrm*: *amici.amici.RDataReporting*)

sets the ReturnData reporting mode

**Parameters** **rdrm** (*amici.amici.RDataReporting*) – ReturnData reporting mode

**setSensiSteadyStateCheck**(*flag*: *bool*)

Sets for which variables convergence checks for steadystate computation are performed.

**Parameters** **flag** (*bool*) – boolean flag to pick state and sensitivity equations (true, default) or only state variables (false).

**setSensitivityMethod**(*sensi\_meth*: amici.amici.SensitivityMethod)

Set sensitivity method

**Parameters** *sensi\_meth* (*amici.amici.SensitivityMethod*) –

**setSensitivityMethodPreequilibration**(*sensi\_meth\_preeq*: amici.amici.SensitivityMethod)

Set sensitivity method for preequilibration

**Parameters** *sensi\_meth\_preeq* (*amici.amici.SensitivityMethod*) –

**setSensitivityOrder**(*sensi*: amici.amici.SensitivityOrder)

Set the sensitivity order

**Parameters** *sensi* (*amici.amici.SensitivityOrder*) – sensitivity order

**setStabilityLimitFlag**(*stldet*: bool)

set stability limit detection mode

**Parameters** *stldet* (bool) – can be false (deactivated) or true (activated)

**setStateOrdering**(*ordering*: int)

Sets KLU / SuperLUMT state ordering mode

This only applies when *linsol* is set to *LinearSolver::KLU* or *LinearSolver::SuperLUMT*. Mind the difference between *SUNLinSolKLU::StateOrdering* and *SUNLinSolSuperLUMT::StateOrdering*.

**Parameters** *ordering* (int) – state ordering

**setSteadyStateSensiToleranceFactor**(*factor*: float)

set the steady state sensitivity simulation tolerance factor.

Steady state sensitivity simulation tolerances are the product of the sensitivity simulation tolerances and this factor, unless manually set with *set(Absolute/Relative)ToleranceSteadyStateSensi()*.

**Parameters** *factor* (float) – tolerance factor (non-negative number)

**setSteadyStateToleranceFactor**(*factor*: float)

set the steady state simulation tolerance factor.

Steady state simulation tolerances are the product of the simulation tolerances and this factor, unless manually set with *set(Absolute/Relative)ToleranceSteadyState()*.

**Parameters** *factor* (float) – tolerance factor (non-negative number)

**class** amici.amici.SolverPtr(\*args)

Swig-Generated class that implements smart pointers to Solver as objects.

**property** logger

**class** amici.amici.SteadyStateComputationMode(value)

An enumeration.

**integrateIfNewtonFails** = 2

**integrationOnly** = 1

**newtonOnly** = 0

**class** amici.amici.SteadyStateSensitivityMode(value)

An enumeration.

**integrateIfNewtonFails** = 2

```

    integrationOnly = 1

    newtonOnly = 0

class amici.amici.SteadyStateStatus(value)
    An enumeration.

    failed = -1

    failed_convergence = -2

    failed_damping = -4

    failed_factorization = -3

    failed_too_long_simulation = -5

    not_run = 0

    success = 1

class amici.amici.SteadyStateStatusVector(*args)

class amici.amici.StringDoubleMap(*args)
    Swig-Generated class templating Dict [str, float] to facilitate interfacing with C++ bindings.

class amici.amici.StringVector(*args)
    Swig-Generated class templating common python types including Iterable [str] and numpy.array [str] to
    facilitate interfacing with C++ bindings.

amici.amici.compiledWithOpenMP() → bool
    AMICI extension was compiled with OpenMP?

    Return type bool

amici.amici.enum(prefix)

amici.amici.getScaledParameter(unscaledParameter: float, scaling: amici.amici.ParameterScaling) → float
    Apply parameter scaling according to scaling

    Parameters
        • unscaledParameter (float) –
        • scaling (amici.amici.ParameterScaling) – parameter scaling

    Return type float

    Returns Scaled parameter

amici.amici.getUnscaledParameter(scaledParameter: float, scaling: amici.amici.ParameterScaling) → float
    Remove parameter scaling according to scaling

    Parameters
        • scaledParameter (float) – scaled parameter
        • scaling (amici.amici.ParameterScaling) – parameter scaling

    Return type float

    Returns Unscaled parameter

```



```
amici.amici.parameterScalingFromIntVector(intVec: Sequence[int]) →
    Tuple[amici.amici.ParameterScaling]
```

Swig-Generated class, which, in contrast to other Vector classes, does not allow for simple interoperability with common Python types, but must be created using `amici.amici.parameterScalingFromIntVector()`

**Return type** `typing.Tuple[amici.amici.ParameterScaling]`

```
amici.amici.runAmiciSimulation(solver: amici.amici.Solver, edata: amici.amici.ExpData, model:
    amici.amici.Model, rethrow: bool = False) → amici.amici.ReturnData
```

Core integration routine. Initializes the solver and runs the forward and backward problem.

#### Parameters

- **solver** (`amici.amici.Solver`) – Solver instance
- **edata** (`amici.amici.ExpData`) – pointer to experimental data object
- **model** (`amici.amici.Model`) – model specification object
- **rethrow** (`bool`) – rethrow integration exceptions?

**Return type** `ReturnData`

**Returns** rdata pointer to return data object

```
amici.amici.runAmiciSimulations(solver: amici.amici.Solver, edatas: amici.amici.ExpDataPtrVector, model:
    amici.amici.Model, failfast: bool, num_threads: int) →
    Tuple[amici.amici.ReturnData]
```

Same as `runAmiciSimulation`, but for multiple `ExpData` instances. When compiled with OpenMP support, this function runs multi-threaded.

#### Parameters

- **solver** (`amici.amici.Solver`) – Solver instance
- **edatas** (`amici.amici.ExpDataPtrVector`) – experimental data objects
- **model** (`amici.amici.Model`) – model specification object
- **failfast** (`bool`) – flag to allow early termination
- **num\_threads** (`int`) – number of threads for parallel execution

**Return type** `Iterable[ReturnData]`

**Returns** vector of pointers to return data objects

```
amici.amici.scaleParameters(bufferUnscaled: Iterable[float], pscale: Iterable[ParameterScaling],
    bufferScaled: Iterable[float])
```

Apply parameter scaling according to *scaling*

#### Parameters

- **bufferUnscaled** (`typing.Iterable[float]`) –
- **pscale** (`typing.Iterable[amici.amici.ParameterScaling]`) – parameter scaling
- **bufferScaled** (`typing.Iterable[float]`) – destination

```
amici.amici.simulation_status_to_str(status: int) → str
```

Get the string representation of the given simulation status code (see `ReturnData::status`).

**Parameters** **status** (`int`) – Status code

**Return type** `str`

**Returns** Name of the variable representing this status code.

`amici.amici.unscaleParameters(bufferScaled: Iterable[float], pscale: Iterable[ParameterScaling],  
bufferUnscaled: Iterable[float])`

Remove parameter scaling according to the parameter scaling in pscale

All vectors must be of same length.

#### Parameters

- **bufferScaled** (`typing.Iterable[float]`) – scaled parameters
- **pscale** (`typing.Iterable[amici.amici.ParameterScaling]`) – parameter scaling
- **bufferUnscaled** (`typing.Iterable[float]`) – unscaled parameters are written to the array

## 10.5.3 amici.sbml\_import

### SBML Import

This module provides all necessary functionality to import a model specified in the [Systems Biology Markup Language \(SBML\)](#).

#### Functions

<code>assignmentRules2observables(sbml_model[, ...])</code>	Turn assignment rules into observables.
<code>get_species_initial(species)</code>	Extract the initial concentration from a given species
<code>replace_logx(math_str)</code>	Replace $\log X(.)$ by $\log(., X)$ since sympy cannot parse the former

#### Classes

<code>SbmlImporter(sbml_source[, ...])</code>	Class to generate AMICI C++ files for a model provided in the Systems Biology Markup Language (SBML).
---	---

**class** `amici.sbml_import.SbmlImporter`(*sbml\_source*, *show\_sbml\_warnings=False*, *from\_file=True*, *discard\_annotations=False*)

Class to generate AMICI C++ files for a model provided in the Systems Biology Markup Language (SBML).

#### Variables

- **show\_sbml\_warnings** – indicates whether libSBML warnings should be displayed
- **symbols** – dict carrying symbolic definitions
- **sbml\_reader** – The libSBML sbml reader

**Warning:** Not storing this may result in a segfault.

- **sbml\_doc** – document carrying the sbml definition

**Warning:** Not storing this may result in a segfault.

- **sbml** – SBML model to import
- **compartments** – dict of compartment ids and compartment volumes
- **stoichiometric\_matrix** – stoichiometric matrix of the model
- **flux\_vector** – reaction kinetic laws
- **flux\_ids** – identifiers for elements of flux\_vector
- **\_local\_symbols** – model symbols for sympy to consider during sympification see *locals* argument in ``sympy.simplify``
- **species\_assignment\_rules** – Assignment rules for species. Key is symbolic identifier and value is assignment value
- **compartment\_assignment\_rules** – Assignment rules for compartments. Key is symbolic identifier and value is assignment value
- **parameter\_assignment\_rules** – assignment rules for parameters, these parameters are not permissible for sensitivity analysis
- **initial\_assignments** – initial assignments for parameters, these parameters are not permissible for sensitivity analysis
- **sbml\_parser\_settings** – sets behaviour of SBML Formula parsing

**\_\_init\_\_**(*sbml\_source*, *show\_sbml\_warnings=False*, *from\_file=True*, *discard\_annotations=False*)

Create a new Model instance.

#### Parameters

- **sbml\_source** (`typing.Union[str, pathlib.Path, libsbml.Model]`) – Either a path to SBML file where the model is specified, or a model string as created by `sbml.sbmlWriter().writeSBMLToString()` or an instance of `libsbml.Model`.
- **show\_sbml\_warnings** (`bool`) – Indicates whether libSBML warnings should be displayed.
- **from\_file** (`bool`) – Whether *sbml\_source* is a file name (True, default), or an SBML string
- **discard\_annotations** (`bool`) – discard information contained in AMICI SBML annotations (debug).

**add\_d\_dt**(*d\_dt*, *variable*, *variable0*, *name*)

Creates or modifies species, to implement rate rules for compartments and species, respectively.

#### Parameters

- **d\_dt** (`sympy.core.expr.Expr`) – The rate rule (or, right-hand side of an ODE).
- **variable** (`sympy.core.symbol.Symbol`) – The subject of the rate rule.
- **variable0** (`typing.Union[float, sympy.core.expr.Expr]`) – The initial value of the variable.
- **name** (`str`) – Species name, only applicable if this function generates a new species

**Return type** `None`

**add\_local\_symbol**(*key*, *value*)

Add local symbols with some sanity checking for duplication which would indicate redefinition of internals, which SBML permits, but we don't.

**Parameters**

- **key** (`str`) – local symbol key
- **value** (`sympy.core.expr.Expr`) – local symbol value

**check\_event\_support**()

Check possible events in the model, as AMICI does currently not support :rtype: `None`

- delays in events
- priorities of events
- events fired at initial time

Furthermore, event triggers are optional (e.g., if an event is fired at initial time, no trigger function is necessary). In this case, warn that this event will have no effect.

**check\_support**()

Check whether all required SBML features are supported. Also ensures that the SBML contains at least one reaction, or rate rule, or assignment rule, to produce change in the system over time.

**Return type** `None`

**is\_assignment\_rule\_target**(*element*)

Checks if an element has a valid assignment rule in the specified model.

**Parameters** **element** (`libsbml.SBase`) – SBML variable

**Return type** `bool`

**Returns** boolean indicating truth of function name

**is\_rate\_rule\_target**(*element*)

Checks if an element has a valid assignment rule in the specified model.

**Parameters** **element** (`libsbml.SBase`) – SBML variable

**Return type** `bool`

**Returns** boolean indicating truth of function name

**process\_conservation\_laws**(*ode\_model*)

Find conservation laws in reactions and species.

**Parameters** **ode\_model** – ODEModel object with basic definitions

**Return type** `None`

**sbml2amici**(*model\_name*, *output\_dir*=`None`, *observables*=`None`, *event\_observables*=`None`, *constant\_parameters*=`None`, *sigmas*=`None`, *event\_sigmas*=`None`, *noise\_distributions*=`None`, *event\_noise\_distributions*=`None`, *verbose*=40, *assume\_pow\_positivity*=`False`, *compiler*=`None`, *allow\_reinit\_fixpar\_initcond*=`True`, *compile*=`True`, *compute\_conservation\_laws*=`True`, *simplify*=`<function _default_simplify>`, *cache\_simplify*=`False`, *log\_as\_log10*=`True`, *generate\_sensitivity\_code*=`True`, *hardcode\_symbols*=`None`)

Generate and compile AMICI C++ files for the model provided to the constructor.

The resulting model can be imported as a regular Python module (if *compile*=`True`), or used from Matlab or C++ as described in the documentation of the respective AMICI interface.

Note that this generates model ODEs for changes in concentrations, not amounts unless the *hasOnlySubstanceUnits* attribute has been defined for a particular species.

Sensitivity analysis for local parameters is enabled by creating global parameters `_{reactionId}_{localParameterName}`.

### Parameters

- **model\_name** (`str`) – Name of the generated model package. Note that in a given Python session, only one model with a given name can be loaded at a time. The generated Python extensions cannot be unloaded. Therefore, make sure to choose a unique name for each model.
- **output\_dir** (`typing.Union[str, pathlib.Path, None]`) – Directory where the generated model package will be stored.
- **observables** (`typing.Optional[typing.Dict[str, typing.Dict[str, str]]]`) – Observables to be added to the model: `dictionary( observableId: {'name': observableName (optional), 'formula': formulaString} )`.
- **event\_observables** (`typing.Optional[typing.Dict[str, typing.Dict[str, str]]]`) – Event observables to be added to the model: `dictionary( eventObservableId: {'name': eventObservableName (optional), 'event': eventId, 'formula': formulaString} )`
- **constant\_parameters** (`typing.Optional[typing.Iterable[str]]`) – list of SBML Ids identifying constant parameters
- **sigmas** (`typing.Optional[typing.Dict[str, typing.Union[str, float]]]`) – `dictionary(observableId: sigma value or (existing) parameter name)`
- **event\_sigmas** (`typing.Optional[typing.Dict[str, typing.Union[str, float]]]`) – `dictionary(eventObservableId: sigma value or (existing) parameter name)`
- **noise\_distributions** (`typing.Optional[typing.Dict[str, typing.Union[str, typing.Callable]]]`) – `dictionary(observableId: noise type)`. If nothing is passed for some observable id, a normal model is assumed as default. Either pass a noise type identifier, or a callable generating a custom noise string. For noise identifiers, see `amici.import_utils.noise_distribution_to_cost_function()`.
- **event\_noise\_distributions** (`typing.Optional[typing.Dict[str, typing.Union[str, typing.Callable]]]`) – `dictionary(eventObservableId: noise type)`. If nothing is passed for some observable id, a normal model is assumed as default. Either pass a noise type identifier, or a callable generating a custom noise string. For noise identifiers, see `amici.import_utils.noise_distribution_to_cost_function()`.
- **verbose** (`typing.Union[int, bool]`) – verbosity level for logging, True/False default to `logging.Error/logging.DEBUG`
- **assume\_pow\_positivity** (`bool`) – if set to True, a special pow function is used to avoid problems with state variables that may become negative due to numerical errors
- **compiler** (`typing.Optional[str]`) – Absolute path to the compiler executable to be used to build the Python extension, e.g. `/usr/bin/clang`.
- **allow\_reinit\_fixpar\_initcond** (`bool`) – see `amici.de_export.ODEExporter`
- **compile** (`bool`) – If True, compile the generated Python package, if False, just generate code.
- **compute\_conservation\_laws** (`bool`) – if set to True, conservation laws are automatically computed and applied such that the state-jacobian of the ODE right-hand-side has

full rank. This option should be set to `True` when using the Newton algorithm to compute steadystate sensitivities. Conservation laws for constant species are enabled by default. Support for conservation laws for non-constant species is experimental and may be enabled by setting an environment variable `AMICI_EXPERIMENTAL_SBML_NONCONST_CLS` to either `demartino` to use the algorithm proposed by De Martino et al. (2014) <https://doi.org/10.1371/journal.pone.0100750>, or to any other value to use the deterministic algorithm implemented in `conserved_moieties2.py`. In some cases, the `demartino` may run for a very long time. This has been observed for example in the case of stoichiometric coefficients with many significant digits.

- **simplify** (`typing.Optional[typing.Callable]`) – see `amici.ODEModel._simplify`
- **cache\_simplify** (`bool`) – see `amici.ODEModel.__init__()`
- **log\_as\_log10** (`bool`) – If `True`, log in the SBML model will be parsed as `log10` (default), if `False`, log will be parsed as natural logarithm `ln`.
- **generate\_sensitivity\_code** (`bool`) – If `False`, the code required for sensitivity computation will not be generated.
- **hardcode\_symbols** (`typing.Optional[typing.Sequence[str]]`) – List of SBML entity IDs that are to be hardcoded in the generated model. Their values cannot be changed anymore after model import. Currently only parameters that are not targets of rules or initial assignments are supported.

**Return type** `None`

`amici.sbml_import.assignmentRules2observables(sbml_model, filter_function=<function <lambda>>)`

Turn assignment rules into observables.

**Parameters**

- **sbml\_model** (`libsbml.Model`) – Model to operate on
- **filter\_function** (`typing.Callable`) – Callback function taking assignment variable as input and returning `True/False` to indicate if the respective rule should be turned into an observable.

**Returns** A dictionary(`observableId`:{ ‘name’: observableName, ‘formula’: formulaString })

`amici.sbml_import.get_species_initial(species)`

Extract the initial concentration from a given species

**Parameters** **species** (`libsbml.Species`) – species index

**Return type** `sympy.core.expr.Expr`

**Returns** initial species concentration

`amici.sbml_import.replace_logx(math_str)`

Replace `logX(.)` by `log(., X)` since sympy cannot parse the former

**Parameters** **math\_str** (`typing.Union[str, float, None]`) – string for sympification

**Return type** `typing.Union[str, float, None]`

**Returns** sympifiable string

## 10.5.4 amici.pysb\_import

### PySB Import

This module provides all necessary functionality to import a model specified in the `pysb.core.Model` format.

### Functions

<code>extract_monomers(complex_patterns)</code>	Constructs a list of monomer names contained in complex patterns.
<code>has_fixed_parameter_ic(specie, pysb_model, ...)</code>	Wrapper to interface <code>de_export.DEModel.state_has_fixed_parameter_initial_condition()</code> from a <code>pysb</code> specie/model arguments
<code>ode_model_from_pysb_importer(model[, ...])</code>	Creates an <code>amici.DEModel</code> instance from a <code>pysb.Model</code> instance.
<code>pysb2amici(model[, output_dir, observables, ...])</code>	Generate AMICI C++ files for the provided model.
<code>pysb_model_from_path(pysb_model_file)</code>	Load a <code>pysb</code> model module and return the <code>pysb.Model</code> instance

`amici.pysb_import.extract_monomers(complex_patterns)`

Constructs a list of monomer names contained in complex patterns. Multiplicity of names corresponds to the stoichiometry in the complex.

**Parameters** `complex_patterns` (`typing.Union[pysb.core.ComplexPattern, typing.List[pysb.core.ComplexPattern]]`) – (list of) complex pattern(s)

**Return type** `typing.List[str]`

**Returns** list of monomer names

`amici.pysb_import.has_fixed_parameter_ic(specie, pysb_model, ode_model)`

Wrapper to interface `de_export.DEModel.state_has_fixed_parameter_initial_condition()` from a `pysb` specie/model arguments

**Parameters**

- **specie** (`pysb.core.ComplexPattern`) – `pysb` species
- **pysb\_model** (`pysb.core.Model`) – `pysb` model
- **ode\_model** (`amici.de_export.DEModel`) – ODE model

**Return type** `bool`

**Returns** False if the species does not have an initial condition at all. Otherwise the return value of `de_export.DEModel.state_has_fixed_parameter_initial_condition()`

`amici.pysb_import.ode_model_from_pysb_importer(model, constant_parameters=None, observables=None, sigmas=None, noise_distributions=None, compute_conservation_laws=True, simplify=<function powsimp>, cache_simplify=False, verbose=False)`

Creates an `amici.DEModel` instance from a `pysb.Model` instance.

**Parameters**

- **model** (`pysb.core.Model`) – see `amici.pysb_import.pysb2amici()`

- **constant\_parameters** (`typing.Optional[typing.List[str]]`) – see `amici.pysb_import.pysb2amici()`
- **observables** (`typing.Optional[typing.List[str]]`) – see `amici.pysb_import.pysb2amici()`
- **sigmas** (`typing.Optional[typing.Dict[str, str]]`) – dict with names of observable Expressions as keys and names of sigma Expressions as value sigma
- **noise\_distributions** (`typing.Optional[typing.Dict[str, typing.Union[str, typing.Callable]]]`) – see `amici.pysb_import.pysb2amici()`
- **compute\_conservation\_laws** (`bool`) – see `amici.pysb_import.pysb2amici()`
- **simplify** (`typing.Callable`) – see `amici.DEModel._simplify`
- **cache\_simplify** (`bool`) – see `amici.DEModel.__init__()` Note that there are possible issues with PySB models: <https://github.com/AMICI-dev/AMICI/pull/1672>
- **verbose** (`typing.Union[int, bool]`) – verbosity level for logging, True/False default to `logging.DEBUG/logging.ERROR`

**Return type** `amici.de_export.DEModel`

**Returns** New `DEModel` instance according to `pysbModel`

```
amici.pysb_import.pysb2amici(model, output_dir=None, observables=None, constant_parameters=None,
                             sigmas=None, noise_distributions=None, verbose=False,
                             assume_pow_positivity=False, compiler=None,
                             compute_conservation_laws=True, compile=True, simplify=<function
                             _default_simplify>, cache_simplify=False, generate_sensitivity_code=True,
                             model_name=None)
```

Generate AMICI C++ files for the provided model.

#### **Warning: PySB models with Compartments**

When importing a PySB model with `pysb.Compartments`, BioNetGen scales reaction fluxes with the compartment size. Instead of using the respective symbols, the compartment size Parameter or Expression is evaluated when generating equations. This may lead to unexpected results if the compartment size parameter is changed for AMICI simulations.

#### **Parameters**

- **model** (`pysb.core.Model`) – `pysb` model, `pysb.Model.name` will determine the name of the generated module
- **output\_dir** (`typing.Union[pathlib.Path, str, None]`) – see `amici.de_export.ODEExporter.set_paths()`
- **observables** (`typing.Optional[typing.List[str]]`) – list of `pysb.core.Expression` or `pysb.core.Observable` names in the provided model that should be mapped to observables
- **sigmas** (`typing.Optional[typing.Dict[str, str]]`) – dict of `pysb.core.Expression` names that should be mapped to sigmas
- **noise\_distributions** (`typing.Optional[typing.Dict[str, typing.Union[str, typing.Callable]]]`) – dict with names of observable Expressions as keys and a noise



type identifier, or a callable generating a custom noise formula string (see `amici.import_utils.noise_distribution_to_cost_function()`). If nothing is passed for some observable id, a normal model is assumed as default.

- **constant\_parameters** (`typing.Optional[typing.List[str]]`) – list of `pysb.core.Parameter` names that should be mapped as fixed parameters
- **verbose** (`typing.Union[int, bool]`) – verbosity level for logging, True/False default to `logging.DEBUG/logging.ERROR`
- **assume\_pow\_positivity** (`bool`) – if set to True, a special pow function is used to avoid problems with state variables that may become negative due to numerical errors
- **compiler** (`typing.Optional[str]`) – Absolute path to the compiler executable to be used to build the Python extension, e.g. `/usr/bin/clang`.
- **compute\_conservation\_laws** (`bool`) – if set to True, conservation laws are automatically computed and applied such that the state-jacobian of the ODE right-hand-side has full rank. This option should be set to True when using the Newton algorithm to compute steadystates
- **compile** (`bool`) – If True, build the python module for the generated model. If false, just generate the source code.
- **simplify** (`typing.Callable`) – see `amici.DEModel._simplify`
- **cache\_simplify** (`bool`) – see `amici.DEModel.__init__()` Note that there are possible issues with PySB models: <https://github.com/AMICI-dev/AMICI/pull/1672>
- **generate\_sensitivity\_code** (`bool`) – if set to False, code for sensitivity computation will not be generated
- **model\_name** (`typing.Optional[str]`) – Name for the generated model module. If None, `pysb.Model.name` will be used.

`amici.pysb_import.pysb_model_from_path(pysb_model_file)`

Load a pysb model module and return the `pysb.Model` instance

**Parameters** `pysb_model_file` (`typing.Union[str, pathlib.Path]`) – Full or relative path to the PySB model module

**Return type** `pysb.core.Model`

**Returns** The pysb Model instance

### 10.5.5 amici.bngl\_import

#### BNGL Import

This module provides all necessary functionality to import a model specified in the *BNGL* format.

## Functions

<code>bngl2amici(bngl_model, *args, **kwargs)</code>	Generate AMICI C++ files for the provided model.
--	--

`amici.bngl_import.bngl2amici(bngl_model, *args, **kwargs)`

Generate AMICI C++ files for the provided model.

### Parameters

- **bngl\_model** (`str`) – bngl model file, model name will determine the name of the generated module
- **args** – see `amici.pysb_import.pysb2amici()` for additional arguments
- **kwargs** – see `amici.pysb_import.pysb2amici()` for additional arguments

**Return type** `None`

## 10.5.6 amici.petab\_import

### PEtab Import

Import a model in the `petab` (<https://github.com/PEtab-dev/PEtab>) format into AMICI.

## Functions

<code>check_model(amici_model, petab_problem)</code>	Check that the model is consistent with the PETA problem.
<code>get_fixed_parameters(petab_problem[, ...])</code>	Determine, set and return fixed model parameters.
<code>get_observation_model(observable_df)</code>	Get observables, sigmas, and noise distributions from PETA observation table in a format suitable for <code>amici.sbml_import.SbmlImporter.sbml2amici()</code> .
<code>import_model([sbml_model, condition_table, ...])</code>	Create AMICI model from PETA problem
<code>import_model_sbml([sbml_model, ...])</code>	Create AMICI model from PETA problem
<code>import_petab_problem(petab_problem[, ...])</code>	Import model from petab problem.
<code>petab_noise_distributions_to_amici(observable_df)</code>	Map from the petab to the amici format of noise distribution identifiers.
<code>petab_scale_to_amici_scale(scale_str)</code>	Convert PETA parameter scaling string to AMICI scaling integer
<code>show_model_info(sbml_model)</code>	Log some model quantities
<code>species_to_parameters(species_ids, sbml_model)</code>	Turn a SBML species into parameters and replace species references inside the model instance.

`amici.petab_import.check_model(amici_model, petab_problem)`

Check that the model is consistent with the PETA problem.

**Return type** `None`

`amici.petab_import.get_fixed_parameters(petab_problem, non_estimated_parameters_as_constants=True)`

Determine, set and return fixed model parameters.

Non-estimated parameters and parameters specified in the condition table are turned into constants (unless they are overridden). Only global SBML parameters are considered. Local parameters are ignored.

**Parameters**

- **petab\_problem** (`petab.problem.Problem`) – The PETab problem instance
- **non\_estimated\_parameters\_as\_constants** – Whether parameters marked as non-estimated in PETab should be considered constant in AMICI. Setting this to `True` will reduce model size and simulation times. If sensitivities with respect to those parameters are required, this should be set to `False`.

**Return type** `typing.List[str]`

**Returns** List of IDs of parameters which are to be considered constant.

`amici.petab_import.get_observation_model(observable_df)`

Get observables, sigmas, and noise distributions from PETab observation table in a format suitable for `amici.sbml_import.SbmlImporter.sbml2amici()`.

**Parameters** **observable\_df** (`pandas.core.frame.DataFrame`) – PETab observables table

**Return type** `typing.Tuple[typing.Dict[str, typing.Dict[str, str]], typing.Dict[str, str], typing.Dict[str, typing.Union[str, float]]]`

**Returns** Tuple of dicts with observables, noise distributions, and sigmas.

`amici.petab_import.import_model(sbml_model=None, condition_table=None, observable_table=None, measurement_table=None, petab_problem=None, model_name=None, model_output_dir=None, verbose=True, allow_reinit_fixpar_initcond=True, validate=True, non_estimated_parameters_as_constants=True, output_parameter_defaults=None, discard_sbml_annotations=False, **kwargs)`

Create AMICI model from PETab problem

**Parameters**

- **sbml\_model** (`typing.Union[str, pathlib.Path, libsbml.Model, None]`) – PETab SBML model or SBML file name. Deprecated, pass `petab_problem` instead.
- **condition\_table** (`typing.Union[str, pathlib.Path, pandas.core.frame.DataFrame, None]`) – PETab condition table. If provided, parameters from there will be turned into AMICI constant parameters (i.e. parameters w.r.t. which no sensitivities will be computed). Deprecated, pass `petab_problem` instead.
- **observable\_table** (`typing.Union[str, pathlib.Path, pandas.core.frame.DataFrame, None]`) – PETab observable table. Deprecated, pass `petab_problem` instead.
- **measurement\_table** (`typing.Union[str, pathlib.Path, pandas.core.frame.DataFrame, None]`) – PETab measurement table. Deprecated, pass `petab_problem` instead.
- **petab\_problem** (`typing.Optional[petab.problem.Problem]`) – PETab problem.
- **model\_name** (`typing.Optional[str]`) – Name of the generated model. If model file name was provided, this defaults to the file name without extension, otherwise the SBML model ID will be used.
- **model\_output\_dir** (`typing.Union[pathlib.Path, str, None]`) – Directory to write the model code to. Will be created if doesn't exist. Defaults to current directory.
- **verbose** (`typing.Union[bool, int, None]`) – Print/log extra information.
- **allow\_reinit\_fixpar\_initcond** (`bool`) – See `amici.de_export.ODEExporter`. Must be enabled if initial states are to be reset after preequilibration.

- **validate** (*bool*) – Whether to validate the PETab problem
- **non\_estimated\_parameters\_as\_constants** – Whether parameters marked as non-estimated in PETab should be considered constant in AMICI. Setting this to *True* will reduce model size and simulation times. If sensitivities with respect to those parameters are required, this should be set to *False*.
- **output\_parameter\_defaults** (*typing.Optional[typing.Dict[str, float]]*) – Optional default parameter values for output parameters introduced in the PETab observables table, in particular for placeholder parameters. Dictionary mapping parameter IDs to default values.
- **discard\_sbml\_annotations** (*bool*) – Discard information contained in AMICI SBML annotations (debug).
- **kwargs** – Additional keyword arguments to be passed to `amici.sbml_import.SbmlImporter.sbml2amici()`.

**Return type** `amici.sbml_import.SbmlImporter`

**Returns** The created `amici.sbml_import.SbmlImporter` instance.

```
amici.petab_import.import_model_sbml(sbml_model=None, condition_table=None, observable_table=None,
                                     measurement_table=None, petab_problem=None,
                                     model_name=None, model_output_dir=None, verbose=True,
                                     allow_reinit_fixpar_initcond=True, validate=True,
                                     non_estimated_parameters_as_constants=True,
                                     output_parameter_defaults=None,
                                     discard_sbml_annotations=False, **kwargs)
```

Create AMICI model from PETab problem

#### Parameters

- **sbml\_model** (*typing.Union[str, pathlib.Path, libsbml.Model, None]*) – PETab SBML model or SBML file name. Deprecated, pass `petab_problem` instead.
- **condition\_table** (*typing.Union[str, pathlib.Path, pandas.core.frame.DataFrame, None]*) – PETab condition table. If provided, parameters from there will be turned into AMICI constant parameters (i.e. parameters w.r.t. which no sensitivities will be computed). Deprecated, pass `petab_problem` instead.
- **observable\_table** (*typing.Union[str, pathlib.Path, pandas.core.frame.DataFrame, None]*) – PETab observable table. Deprecated, pass `petab_problem` instead.
- **measurement\_table** (*typing.Union[str, pathlib.Path, pandas.core.frame.DataFrame, None]*) – PETab measurement table. Deprecated, pass `petab_problem` instead.
- **petab\_problem** (*typing.Optional[petab.problem.Problem]*) – PETab problem.
- **model\_name** (*typing.Optional[str]*) – Name of the generated model. If model file name was provided, this defaults to the file name without extension, otherwise the SBML model ID will be used.
- **model\_output\_dir** (*typing.Union[pathlib.Path, str, None]*) – Directory to write the model code to. Will be created if doesn't exist. Defaults to current directory.
- **verbose** (*typing.Union[bool, int, None]*) – Print/log extra information.
- **allow\_reinit\_fixpar\_initcond** (*bool*) – See `amici.de_export.ODEExporter`. Must be enabled if initial states are to be reset after preequilibration.
- **validate** (*bool*) – Whether to validate the PETab problem

- **non\_estimated\_parameters\_as\_constants** – Whether parameters marked as non-estimated in PETab should be considered constant in AMICI. Setting this to `True` will reduce model size and simulation times. If sensitivities with respect to those parameters are required, this should be set to `False`.
- **output\_parameter\_defaults** (`typing.Optional[typing.Dict[str, float]]`) – Optional default parameter values for output parameters introduced in the PETab observables table, in particular for placeholder parameters. Dictionary mapping parameter IDs to default values.
- **discard\_sbml\_annotations** (`bool`) – Discard information contained in AMICI SBML annotations (debug).
- **kwargs** – Additional keyword arguments to be passed to `amici.sbml_import.SbmlImporter.sbml2amici()`.

**Return type** `amici.sbml_import.SbmlImporter`

**Returns** The created `amici.sbml_import.SbmlImporter` instance.

```
amici.petab_import.import_petab_problem(petab_problem, model_output_dir=None, model_name=None,
                                       force_compile=False,
                                       non_estimated_parameters_as_constants=True, **kwargs)
```

Import model from petab problem.

#### Parameters

- **petab\_problem** (`petab.problem.Problem`) – A petab problem containing all relevant information on the model.
- **model\_output\_dir** (`typing.Union[pathlib.Path, str, None]`) – Directory to write the model code to. Will be created if doesn't exist. Defaults to current directory.
- **model\_name** (`typing.Optional[str]`) – Name of the generated model. If model file name was provided, this defaults to the file name without extension, otherwise the model ID will be used.
- **force\_compile** (`bool`) – Whether to compile the model even if the target folder is not empty, or the model exists already.
- **non\_estimated\_parameters\_as\_constants** – Whether parameters marked as non-estimated in PETab should be considered constant in AMICI. Setting this to `True` will reduce model size and simulation times. If sensitivities with respect to those parameters are required, this should be set to `False`.
- **kwargs** – Additional keyword arguments to be passed to `amici.sbml_import.SbmlImporter.sbml2amici()`.

**Return type** `amici.amici.Model`

**Returns** The imported model.

```
amici.petab_import.petab_noise_distributions_to_amici(observable_df)
```

Map from the petab to the amici format of noise distribution identifiers.

**Parameters** **observable\_df** (`pandas.core.frame.DataFrame`) – PETab observable table

**Return type** `typing.Dict[str, str]`

**Returns** Dictionary of `observable_id => AMICI noise-distributions`

`amici.petab_import.petab_scale_to_amici_scale(scale_str)`

Convert PETab parameter scaling string to AMICI scaling integer

**Return type** `int`

`amici.petab_import.show_model_info(sbml_model)`

Log some model quantities

`amici.petab_import.species_to_parameters(species_ids, sbml_model)`

Turn a SBML species into parameters and replace species references inside the model instance.

**Parameters**

- **species\_ids** (`typing.List[str]`) – List of SBML species ID to convert to parameters with the same ID as the replaced species.
- **sbml\_model** (`libsbml.Model`) – SBML model to modify

**Return type** `typing.List[str]`

**Returns** List of IDs of species which have been converted to parameters

## 10.5.7 amici.petab\_import\_pysb

### PySB-PETab Import

Import a model in the PySB-adapted `petab` (<https://github.com/PETab-dev/PETab>) format into AMICI.

### Functions

---

<code>import_model_pysb(petab_problem[, ...])</code>	Create AMICI model from PySB-PETab problem
--	--

---

`amici.petab_import_pysb.import_model_pysb(petab_problem, model_output_dir=None, verbose=True, model_name=None, **kwargs)`

Create AMICI model from PySB-PETab problem

**Parameters**

- **petab\_problem** (`petab.problem.Problem`) – PySB PETab problem
- **model\_output\_dir** (`typing.Union[pathlib.Path, str, None]`) – Directory to write the model code to. Will be created if doesn't exist. Defaults to current directory.
- **verbose** (`typing.Union[bool, int, None]`) – Print/log extra information.
- **model\_name** (`typing.Optional[str]`) – Name of the generated model module
- **kwargs** – Additional keyword arguments to be passed to `amici.pysb_import.pysb2amici()`.

**Return type** `None`

## 10.5.8 amici.petab\_objective

### PETab Objective

Functionality related to running simulations or evaluating the objective function as defined by a PETab problem

### Functions

<code>aggregate_sllh(amici_model, rdatas, ...[, ...])</code>	Aggregate likelihood gradient for all conditions, according to PETab parameter mapping.
<code>create_edata_for_condition(condition, ...)</code>	Get <code>amici.amici.ExpData</code> for the given PETab condition.
<code>create_edatas(amici_model, petab_problem[, ...])</code>	Create list of <code>amici.amici.ExpData</code> objects for PETab problem.
<code>create_parameter_mapping(petab_problem, ...)</code>	Generate AMICI specific parameter mapping.
<code>create_parameter_mapping_for_condition(...)</code>	Generate AMICI specific parameter mapping for condition.
<code>create_parameterized_edatas(amici_model, ...)</code>	Create list of <code>:class:amici.ExpData</code> objects with parameters filled in.
<code>rdatas_to_measurement_df(rdatas, model, ...)</code>	Create a measurement dataframe in the PETab format from the passed rdatas and own information.
<code>rdatas_to_simulation_df(rdatas, model, ...)</code>	Create a PETab simulation dataframe from <code>amici.amici.ReturnData</code> s.
<code>rescale_sensitivity(sensitivity, ...)</code>	Rescale a sensitivity between parameter scales.
<code>simulate_petab(petab_problem, amici_model[, ...])</code>	Simulate PETab model.

`amici.petab_objective.aggregate_sllh(amici_model, rdatas, parameter_mapping, edatas, petab_scale=True, petab_problem=None)`

Aggregate likelihood gradient for all conditions, according to PETab parameter mapping.

#### Parameters

- **amici\_model** (`typing.Union[amici.amici.Model, amici.amici.ModelPtr]`) – AMICI model from which rdatas were obtained.
- **rdatas** (`typing.Sequence[amici.numpy.ReturnDataView]`) – Simulation results.
- **parameter\_mapping** (`typing.Optional[amici.parameter_mapping.ParameterMapping]`) – PETab parameter mapping to condition-specific simulation parameters.
- **edatas** (`typing.List[typing.Union[amici.swig_wrappers.ExpData, amici.amici.ExpDataPtr]]`) – Experimental data used for simulation.
- **petab\_scale** (`bool`) – Whether to check that sensitivities were computed with parameters on the scales provided in the PETab parameters table.
- **petab\_problem** (`typing.Optional[petab.problem.Problem]`) – The PETab problem that defines the parameter scales.

**Return type** `typing.Optional[typing.Dict[str, float]]`

**Returns** Aggregated likelihood sensitivities.



```
amici.petab_objective.create_edata_for_condition(condition, measurement_df, amici_model,
                                                petab_problem, observable_ids)
```

Get `amici.amici.ExpData` for the given PETab condition.

Sets timepoints, observed data and sigmas.

#### Parameters

- **condition** (`typing.Union[typing.Dict, pandas.core.series.Series]`) – `pandas.DataFrame` row with `preequilibrationConditionId` and `simulationConditionId`.
- **measurement\_df** (`pandas.core.frame.DataFrame`) – `pandas.DataFrame` with measurements for the given condition.
- **amici\_model** (`typing.Union[amici.amici.Model, amici.amici.ModelPtr]`) – AMICI model
- **petab\_problem** (`petab.problem.Problem`) – Underlying PETab problem
- **observable\_ids** (`typing.List[str]`) – List of observable IDs

**Return type** `amici.swig_wrappers.ExpData`

**Returns** `ExpData` instance.

```
amici.petab_objective.create_edatas(amici_model, petab_problem, simulation_conditions=None)
```

Create list of `amici.amici.ExpData` objects for PETab problem.

#### Parameters

- **amici\_model** (`typing.Union[amici.amici.Model, amici.amici.ModelPtr]`) – AMICI model.
- **petab\_problem** (`petab.problem.Problem`) – Underlying PETab problem.
- **simulation\_conditions** (`typing.Union[pandas.core.frame.DataFrame, typing.Dict, None]`) – Result of `petab.get_simulation_conditions()`. Can be provided to save time if this has been obtained before.

**Return type** `typing.List[amici.swig_wrappers.ExpData]`

**Returns** List with one `amici.amici.ExpData` per simulation condition, with filled in timepoints and data.

```
amici.petab_objective.create_parameter_mapping(petab_problem, simulation_conditions,
                                              scaled_parameters, amici_model,
                                              **parameter_mapping_kwargs)
```

Generate AMICI specific parameter mapping.

#### Parameters

- **petab\_problem** (`petab.problem.Problem`) – PETab problem
- **simulation\_conditions** (`typing.Union[pandas.core.frame.DataFrame, typing.List[typing.Dict]]`) – Result of `petab.get_simulation_conditions()`. Can be provided to save time if this has been obtained before.
- **scaled\_parameters** (`bool`) – If `True`, problem parameters are assumed to be on the scale provided in the PETab parameter table and will be unscaled. If `False`, they are assumed to be in linear scale.
- **amici\_model** (`typing.Union[amici.amici.Model, amici.amici.ModelPtr]`) – AMICI model.



- **parameter\_mapping\_kwargs** – Optional keyword arguments passed to `petab.get_optimization_to_simulation_parameter_mapping()`. To allow changing fixed PETab problem parameters (`estimate=0`), use `fill_fixed_parameters=False`.

**Return type** `amici.parameter_mapping.ParameterMapping`

**Returns** List of the parameter mappings.

```
amici.petab_objective.create_parameter_mapping_for_condition(parameter_mapping_for_condition,
                                                           condition, petab_problem,
                                                           amici_model)
```

Generate AMICI specific parameter mapping for condition.

#### Parameters

- **parameter\_mapping\_for\_condition** (`typing.Tuple[typing.Dict[str, typing.Union[str, numbers.Number]], typing.Dict[str, typing.Union[str, numbers.Number]], typing.Dict[str, str], typing.Dict[str, str]]`) – Preliminary parameter mapping for condition.
- **condition** (`typing.Union[pandas.core.series.Series, typing.Dict]`) – `pandas.DataFrame` row with `preequilibrationConditionId` and `simulationConditionId`.
- **petab\_problem** (`petab.problem.Problem`) – Underlying PETab problem.
- **amici\_model** (`typing.Union[amici.amici.Model, amici.amici.ModelPtr]`) – AMICI model.

**Return type** `amici.parameter_mapping.ParameterMappingForCondition`

**Returns** The parameter and parameter scale mappings, for fixed preequilibration, fixed simulation, and variable simulation parameters, and then the respective scalings.

```
amici.petab_objective.create_parameterized_edatas(amici_model, petab_problem, problem_parameters,
                                                  scaled_parameters=False,
                                                  parameter_mapping=None,
                                                  simulation_conditions=None)
```

Create list of :class:amici.ExpData objects with parameters filled in.

#### Parameters

- **amici\_model** (`typing.Union[amici.amici.Model, amici.amici.ModelPtr]`) – AMICI Model assumed to be compatible with `petab_problem`.
- **petab\_problem** (`petab.problem.Problem`) – PETab problem to work on.
- **problem\_parameters** (`typing.Dict[str, numbers.Number]`) – Run simulation with these parameters. If `None`, PETab `nominalValues` will be used. To be provided as dict, mapping PETab problem parameters to SBML IDs.
- **scaled\_parameters** (`bool`) – If `True`, `problem_parameters` are assumed to be on the scale provided in the PETab parameter table and will be unscaled. If `False`, they are assumed to be in linear scale.
- **parameter\_mapping** (`typing.Optional[amici.parameter_mapping.ParameterMapping]`) – Optional precomputed PETab parameter mapping for efficiency, as generated by `create_parameter_mapping()`.
- **simulation\_conditions** (`typing.Union[pandas.core.frame.DataFrame, typing.Dict, None]`) – Result of `petab.get_simulation_conditions()`. Can be provided to save time if this has been obtained before.

**Return type** `typing.List[amici.swig_wrappers.ExpData]`

**Returns** List with one `amici.amici.ExpData` per simulation condition, with filled in timepoints, data and parameters.

`amici.petab_objective.rdatas_to_measurement_df(rdatas, model, measurement_df)`

Create a measurement dataframe in the PETab format from the passed `rdatas` and own information.

**Parameters**

- **rdatas** (`typing.Sequence[amici.amici.ReturnData]`) – A sequence of `rdatas` with the ordering of `petab.get_simulation_conditions()`.
- **model** (`typing.Union[amici.amici.Model, amici.amici.ModelPtr]`) – AMICI model used to generate `rdatas`.
- **measurement\_df** (`pandas.core.frame.DataFrame`) – PETab measurement table used to generate `rdatas`.

**Return type** `pandas.core.frame.DataFrame`

**Returns** A dataframe built from the `rdatas` in the format of `measurement_df`.

`amici.petab_objective.rdatas_to_simulation_df(rdatas, model, measurement_df)`

Create a PETab simulation dataframe from `amici.amici.ReturnData` s.

See `rdatas_to_measurement_df()` for details, only that model outputs will appear in column `simulation` instead of `measurement`.

**Return type** `pandas.core.frame.DataFrame`

`amici.petab_objective.rescale_sensitivity(sensitivity, parameter_value, old_scale, new_scale)`

Rescale a sensitivity between parameter scales.

**Parameters**

- **sensitivity** (`float`) – The sensitivity corresponding to the parameter value.
- **parameter\_value** (`float`) – The parameter vector element, on `old_scale`.
- **old\_scale** (`str`) – The scale of the parameter value.
- **new\_scale** (`str`) – The parameter scale on which to rescale the sensitivity.

**Return type** `float`

**Returns** The rescaled sensitivity.

`amici.petab_objective.simulate_petab(petab_problem, amici_model, solver=None,  
 problem_parameters=None, simulation_conditions=None,  
 edatas=None, parameter_mapping=None,  
 scaled_parameters=False, log_level=30, num_threads=1,  
 failfast=True, scaled_gradients=False)`

Simulate PETab model.

---

**Note:** Regardless of `scaled_parameters`, unscaled sensitivities are returned, unless `scaled_gradients=True`.

---

**Parameters**

- **petab\_problem** (`petab.problem.Problem`) – PETab problem to work on.
- **amici\_model** (`typing.Union[amici.amici.Model, amici.amici.ModelPtr]`) – AMICI Model assumed to be compatible with `petab_problem`.

- **solver** (`typing.Optional[amici.amici.Solver]`) – An AMICI solver. Will use default options if None.
- **problem\_parameters** (`typing.Optional[typing.Dict[str, float]]`) – Run simulation with these parameters. If None, PETab nominalValues will be used. To be provided as dict, mapping PETab problem parameters to SBML IDs.
- **simulation\_conditions** (`typing.Union[pandas.core.frame.DataFrame, typing.Dict, None]`) – Result of `petab.get_simulation_conditions()`. Can be provided to save time if this has been obtained before. Not required if `edatas` and `parameter_mapping` are provided.
- **edatas** (`typing.Optional[typing.List[typing.Union[amici.swig_wrappers.ExpData, amici.amici.ExpDataPtr]]]`) – Experimental data. Parameters are inserted in-place for simulation.
- **parameter\_mapping** (`typing.Optional[amici.parameter_mapping.ParameterMapping]`) – Optional precomputed PETab parameter mapping for efficiency, as generated by `create_parameter_mapping()`.
- **scaled\_parameters** (`typing.Optional[bool]`) – If True, `problem_parameters` are assumed to be on the scale provided in the PETab parameter table and will be unscaled. If False, they are assumed to be in linear scale.
- **log\_level** (`int`) – Log level, see `amici.logging` module.
- **num\_threads** (`int`) – Number of threads to use for simulating multiple conditions (only used if compiled with OpenMP).
- **failfast** (`bool`) – Returns as soon as an integration failure is encountered, skipping any remaining simulations.
- **scaled\_gradients** (`bool`) – Whether to compute gradients on parameter scale (True) or not (False).

**Return type** `typing.Dict[str, typing.Any]`

#### Returns

Dictionary of

- cost function value (LLH),
- list of `amici.amici.ReturnData` (RDATAS),
- list of `amici.amici.ExpData` (EDATAS),

corresponding to the different simulation conditions. For ordering of simulation conditions, see `petab.Problem.get_simulation_conditions_from_measurement_df()`.

## 10.5.9 amici.petab\_simulate

### PETab Simulate

Functionality related to the use of AMICI for simulation with PETab's Simulator class.

Use cases:

- generate data for use with PETab's plotting methods
- generate synthetic data

## Classes

<code>PetabSimulator(*args[, amici_model])</code>	Implementation of the <i>PEtab Simulator</i> class that uses AMICI.
---	---

---

**class** amici.petab\_simulate.**PetabSimulator**(\*args, amici\_model=None, \*\*kwargs)

Implementation of the *PEtab Simulator* class that uses AMICI.

**\_\_init\_\_**(\*args, amici\_model=None, \*\*kwargs)

Initialize the simulator.

Initialize the simulator with sufficient information to perform a simulation. If no working directory is specified, a temporary one is created.

### Parameters

- **petab\_problem** – A *PEtab* problem.
- **working\_dir** – All simulator-specific output files will be saved here. This directory and its contents may be modified and deleted, and should be considered ephemeral.

**add\_noise**(simulation\_df, noise\_scaling\_factor=1, \*\*kwargs)

Add noise to simulated data.

### Parameters

- **simulation\_df** (`pandas.core.frame.DataFrame`) – A *PEtab* measurements table that contains simulated data.
- **noise\_scaling\_factor** (`float`) – A multiplier of the scale of the noise distribution.
- **\*\*kwargs** – Additional keyword arguments are passed to `sample_noise()`.

**Return type** `pandas.core.frame.DataFrame`

**Returns** Simulated data with noise, as a *PEtab* measurements table.

**remove\_working\_dir**(force=False, \*\*kwargs)

Remove the simulator working directory, and all files within.

See the `petab.simulate.Simulator.__init__()` method arguments.

### Parameters

- **force** (`bool`) – If True, the working directory is removed regardless of whether it is a temporary directory.
- **\*\*kwargs** – Additional keyword arguments are passed to `shutil.rmtree()`.

**Return type** `None`

**simulate**(noise=False, noise\_scaling\_factor=1, as\_measurement=False, \*\*kwargs)

Simulate a *PEtab* problem, optionally with noise.

### Parameters

- **noise** (`bool`) – If True, noise is added to simulated data.
- **noise\_scaling\_factor** (`float`) – A multiplier of the scale of the noise distribution.
- **as\_measurement** (`bool`) – Whether the data column is named `petab.C.MEASUREMENT` (True) or `petab.C.SIMULATION` (False).

- **\*\*kwargs** – Additional keyword arguments are passed to `petab.simulate.Simulator.simulate_without_noise()`.

**Return type** `pandas.core.frame.DataFrame`

**Returns** Simulated data, as a PETab measurements table.

**`simulate_without_noise(**kwargs)`**

See `petab.simulate.Simulator.simulate()` docstring.

Additional keyword arguments can be supplied to specify arguments for the AMICI PETab import, simulate, and export methods. See the docstrings for the respective methods for argument options: - `amici.petab_import.import_petab_problem()`, and - `amici.petab_objective.simulate_petab()`.

Note that some arguments are expected to have already been specified in the Simulator constructor (including the PETab problem).

**Return type** `pandas.core.frame.DataFrame`

### 10.5.10 amici.import\_utils

Miscellaneous functions related to model import, independent of any specific model format

#### Functions

<code>cast_to_sym(value, input_name)</code>	Typecasts the value to <code>sympy.Float</code> if possible, and ensures the value is a symbolic expression.
<code>generate_flux_symbol(reaction_index[, name])</code>	Generate identifier symbol for a reaction flux.
<code>generate_measurement_symbol(observable_id)</code>	Generates the appropriate measurement symbol for the provided observable
<code>generate_regularization_symbol(observable_id)</code>	Generates the appropriate regularization symbol for the provided observable
<code>grouper(iterable, n[, fillvalue])</code>	Collect data into fixed-length chunks or blocks
<code>noise_distribution_to_cost_function(...)</code>	Parse noise distribution string to a cost function definition amici can work with.
<code>noise_distribution_to_observable_transformation(...)</code>	Parse noise distribution string and extract observable transformation
<code>smart_subs(element, old, new)</code>	Optimized substitution that checks whether anything needs to be done first
<code>smart_subs_dict(sym, subs[, field, reverse])</code>	Substitutes expressions completely flattening them out.
<code>strip_pysb(symbol)</code>	Strips pysb info from a <code>pysb.Component</code> object
<code>symbol_with_assumptions(name)</code>	Central function to create symbols with consistent, canonical assumptions
<code>toposort_symbols(symbols[, field])</code>	Topologically sort symbol definitions according to their interdependency

## Classes

---

<i>ObservableTransformation</i> (value)	Different modes of observable transformation.
---	---

---

## Exceptions

---

<i>CircularDependencyError</i> (data)
---------------------------------------

---

---

<i>SBMLException</i>
----------------------

---

**exception** amici.import\_utils.**CircularDependencyError**(data)

    \_\_init\_\_(data)

    args

    with\_traceback()

        Exception.with\_traceback(tb) – set self.\_\_traceback\_\_ to tb and return self.

**class** amici.import\_utils.**ObservableTransformation**(value)

    Different modes of observable transformation.

    LIN = 'lin'

    LOG = 'log'

    LOG10 = 'log10'

**exception** amici.import\_utils.**SBMLException**

    \_\_init\_\_(\*args, \*\*kwargs)

    args

    with\_traceback()

        Exception.with\_traceback(tb) – set self.\_\_traceback\_\_ to tb and return self.

amici.import\_utils.**cast\_to\_sym**(value, input\_name)

    Typecasts the value to sympy.Float if possible, and ensures the value is a symbolic expression.

**Parameters**

- **value** (`typing.Union[typing.SupportsFloat, sympy.core.expr.Expr, sympy.logic.boolalg.BooleanAtom]`) – value to be cast
- **input\_name** (`str`) – name of input variable

**Return type** `sympy.core.expr.Expr`

**Returns** typecast value

amici.import\_utils.**generate\_flux\_symbol**(reaction\_index, name=None)

    Generate identifier symbol for a reaction flux. This function will always return the same unique python object for a given entity.

**Parameters**

- **reaction\_index** (`int`) – index of the reaction to which the flux corresponds
- **name** (`typing.Optional[str]`) – an optional identifier of the reaction to which the flux corresponds

**Return type** `sympy.core.symbol.Symbol`

**Returns** identifier symbol

`amici.import_utils.generate_measurement_symbol(observable_id)`

Generates the appropriate measurement symbol for the provided observable

**Parameters** **observable\_id** (`typing.Union[str, sympy.core.symbol.Symbol]`) – symbol (or string representation) of the observable

**Returns** symbol for the corresponding measurement

`amici.import_utils.generate_regularization_symbol(observable_id)`

Generates the appropriate regularization symbol for the provided observable

**Parameters** **observable\_id** (`typing.Union[str, sympy.core.symbol.Symbol]`) – symbol (or string representation) of the observable

**Returns** symbol for the corresponding regularization

`amici.import_utils.grouper(iterable, n, fillvalue=None)`

Collect data into fixed-length chunks or blocks

`grouper('ABCDEFGH', 3, 'x') → ABC DEF Gxx`

**Parameters**

- **iterable** (`typing.Iterable`) – any iterable
- **n** (`int`) – chunk length
- **fillvalue** (`typing.Optional[typing.Any]`) – padding for last chunk if length < n

**Return type** `typing.Iterable[typing.Tuple[typing.Any]]`

**Returns** `itertools.zip_longest` of requested chunks

`amici.import_utils.noise_distribution_to_cost_function(noise_distribution)`

Parse noise distribution string to a cost function definition amici can work with.

The noise distributions listed in the following are supported.  $m$  denotes the measurement,  $y$  the simulation, and  $\sigma$  a distribution scale parameter (currently, AMICI only supports a single distribution parameter).

- *'normal'*, *'lin-normal'*: A normal distribution:

$$\pi(m|y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(m - y)^2}{2\sigma^2}\right)$$

- *'log-normal'*: A log-normal distribution (i.e.  $\log(m)$  is normally distributed):

$$\pi(m|y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma m} \exp\left(-\frac{(\log m - \log y)^2}{2\sigma^2}\right)$$

- *'log10-normal'*: A log10-normal distribution (i.e.  $\log_{10}(m)$  is normally distributed):

$$\pi(m|y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma m \log(10)} \exp\left(-\frac{(\log_{10} m - \log_{10} y)^2}{2\sigma^2}\right)$$

- ‘laplace’, ‘lin-laplace’: A laplace distribution:

$$\pi(m|y, \sigma) = \frac{1}{2\sigma} \exp\left(-\frac{|m - y|}{\sigma}\right)$$

- ‘log-laplace’: A log-Laplace distribution (i.e.  $\log(m)$  is Laplace distributed):

$$\pi(m|y, \sigma) = \frac{1}{2\sigma m} \exp\left(-\frac{|\log m - \log y|}{\sigma}\right)$$

- ‘log10-laplace’: A log10-Laplace distribution (i.e.  $\log_{10}(m)$  is Laplace distributed):

$$\pi(m|y, \sigma) = \frac{1}{2\sigma m \log(10)} \exp\left(-\frac{|\log_{10} m - \log_{10} y|}{\sigma}\right)$$

- ‘binomial’, ‘lin-binomial’: A (continuation of a discrete) binomial distribution, parameterized via the success probability  $p = \sigma$ :

$$\pi(m|y, \sigma) = \text{Heaviside}(y - m) \cdot \frac{\Gamma(y + 1)}{\Gamma(m + 1)\Gamma(y - m + 1)} \sigma^m (1 - \sigma)^{(y-m)}$$

- ‘negative-binomial’, ‘lin-negative-binomial’: A (continuation of a discrete) negative binomial distribution, with  $\text{mean} = y$ , parameterized via success probability  $p$ :

$$\pi(m|y, \sigma) = \frac{\Gamma(m + r)}{\Gamma(m + 1)\Gamma(r)} (1 - \sigma)^m \sigma^r$$

where

$$r = \frac{1 - \sigma}{\sigma} y$$

The distributions above are for a single data point. For a collection  $D = \{m_i\}_i$  of data points and corresponding simulations  $Y = \{y_i\}_i$  and noise parameters  $\Sigma = \{\sigma_i\}_i$ , AMICI assumes independence, i.e. the full distributions is

$$\pi(D|Y, \Sigma) = \prod_i \pi(m_i|y_i, \sigma_i)$$

AMICI uses the logarithm  $\log(\pi(m|y, \sigma))$ .

In addition to the above mentioned distributions, it is also possible to pass a function taking a symbol string and returning a log-distribution string with variables ‘{str\_symbol}’, ‘m{str\_symbol}’, ‘sigma{str\_symbol}’ for  $y$ ,  $m$ ,  $\sigma$ , respectively.

**Parameters** `noise_distribution` (`typing.Union[str, typing.Callable]`) – An identifier specifying a noise model. Possible values are

{‘normal’, ‘lin-normal’, ‘log-normal’, ‘log10-normal’, ‘laplace’, ‘lin-laplace’, ‘log-laplace’, ‘log10-laplace’, ‘binomial’, ‘lin-binomial’, ‘negative-binomial’, ‘lin-negative-binomial’, <Callable>}

For the meaning of the values see above.

**Return type** `typing.Callable[[str], str]`

**Returns** A function that takes a `strSymbol` and then creates a cost function string (negative log-likelihood) from it, which can be sympified.



`amici.import_utils.noise_distribution_to_observable_transformation(noise_distribution)`

Parse noise distribution string and extract observable transformation

**Parameters** `noise_distribution` (`typing.Union[str, typing.Callable]`) – see `noise_distribution_to_cost_function()`

**Return type** `amici.import_utils.ObservableTransformation`

**Returns** observable transformation

`amici.import_utils.smart_subs(element, old, new)`

Optimized substitution that checks whether anything needs to be done first

**Parameters**

- **element** (`sympy.core.expr.Expr`) – substitution target
- **old** (`sympy.core.symbol.Symbol`) – to be substituted
- **new** (`sympy.core.expr.Expr`) – substitution value

**Return type** `sympy.core.expr.Expr`

**Returns** substituted expression

`amici.import_utils.smart_subs_dict(sym, subs, field=None, reverse=True)`

Substitutes expressions completely flattening them out. Requires sorting of expressions with `toposort`.

**Parameters**

- **sym** (`sympy.core.expr.Expr`) – Symbolic expression in which expressions will be substituted
- **subs** (`typing.Dict[sympy.core.symbol.Symbol, typing.Union[typing.Dict[str, sympy.core.expr.Expr], sympy.core.expr.Expr]]`) – Substitutions
- **field** (`typing.Optional[str]`) – Field of substitution expressions in `subs.values()`, if applicable
- **reverse** (`bool`) – Whether ordering in `subs` should be reversed. Note that substitution requires the reverse order of what is required for evaluation.

**Return type** `sympy.core.expr.Expr`

**Returns** Substituted symbolic expression

`amici.import_utils.strip_pysb(symbol)`

Strips pysb info from a `pysb.Component` object

**Parameters** **symbol** (`sympy.core.basic.Basic`) – symbolic expression

**Return type** `sympy.core.basic.Basic`

**Returns** stripped expression

`amici.import_utils.symbol_with_assumptions(name)`

Central function to create symbols with consistent, canonical assumptions

**Parameters** **name** (`str`) – name of the symbol

**Returns** symbol with canonical assumptions

`amici.import_utils.toposort_symbols(symbols, field=None)`

Topologically sort symbol definitions according to their interdependency

**Parameters**

- **symbols** (`typing.Dict[sympy.core.symbol.Symbol, typing.Union[typing.Dict[str, sympy.core.expr.Expr], sympy.core.expr.Expr]]`) – symbol definitions
- **field** (`typing.Optional[str]`) – field of `definition.values()` that is used to compute inter-dependency

**Return type** `typing.Dict[sympy.core.symbol.Symbol, typing.Union[typing.Dict[str, sympy.core.expr.Expr], sympy.core.expr.Expr]]`

**Returns** ordered symbol definitions

## 10.5.11 amici.de\_export

### C++ Export

This module provides all necessary functionality specify an DE model and generate executable C++ simulation code. The user generally won't have to directly call any function from this module as this will be done by `amici.pysb_import.pysb2amici()`, `amici.sbml_import.SbmlImporter.sbml2amici()` and `amici.petab_import.import_model()`.

### Functions

<code>apply_template(source_file, target_file, ...)</code>	Load source file, apply template substitution as provided in <code>templateData</code> and save as <code>targetFile</code> .
<code>get_function_extern_declaration(fun, name, ode)</code>	Constructs the extern function declaration for a given function
<code>get_model_override_implementation(fun, name, ode)</code>	Constructs <code>amici::Model::*</code> override implementation for a given function
<code>get_sunindex_extern_declaration(fun, name, ...)</code>	Constructs the function declaration for an index function of a given function
<code>get_sunindex_override_implementation(fun, ...)</code>	Constructs the <code>amici::Model</code> function implementation for an index function of a given function
<code>is_valid_identifier(x)</code>	Check whether $x$ is a valid identifier for conditions, parameters, observables.
<code>remove_argument_types(signature)</code>	Strips argument types from a function signature
<code>smart_is_zero_matrix(x)</code>	A faster implementation of <code>sympy's is_zero_matrix</code>
<code>smart_jacobian(eq, sym_var)</code>	Wrapper around symbolic jacobian with some additional checks that reduce computation time for large matrices
<code>smart_multiply(x, y)</code>	Wrapper around symbolic multiplication with some additional checks that reduce computation time for large matrices
<code>var_in_function_signature(name, varname, ode)</code>	Checks if the values for a symbolic variable is passed in the signature of a function

## Classes

<code>DEExporter</code> ( <code>de_model</code> [, <code>outdir</code> , <code>verbose</code> , ...])	The DEExporter class generates AMICI C++ files for a model as defined in symbolic expressions.
<code>DEModel</code> ([, <code>verbose</code> , <code>simplify</code> , <code>cache_simplify</code> ])	Defines a Differential Equation as set of ModelQuantities.
<code>TemplateAmici</code> ( <code>template</code> )	Template format used in AMICI (see <code>string.Template</code> for more details).

```
class amici.de_export.DEExporter(de_model, outdir=None, verbose=False, assume_pow_positivity=False,
                                compiler=None, allow_reinit_fixpar_initcond=True,
                                generate_sensitivity_code=True, model_name='model')
```

The DEExporter class generates AMICI C++ files for a model as defined in symbolic expressions.

### Variables

- **model** – DE definition
- **verbose** – more verbose output if True
- **assume\_pow\_positivity** – if set to true, a special pow function is used to avoid problems with state variables that may become negative due to numerical errors
- **compiler** – Absolute path to the compiler executable to be used to build the Python extension, e.g. `/usr/bin/clang`.
- **functions** – carries C++ function signatures and other specifications
- **model\_name** – name of the model that will be used for compilation
- **model\_path** – path to the generated model specific files
- **model\_swig\_path** – path to the generated swig files
- **allow\_reinit\_fixpar\_initcond** – indicates whether reinitialization of initial states depending on fixedParameters is allowed for this model
- **\_build\_hints** – If the given model uses special functions, this set contains hints for model building.
- **generate\_sensitivity\_code** – Specifies whether code for sensitivity computation is to be generated

**Note:** When importing large models (several hundreds of species or parameters), import time can potentially be reduced by using multiple CPU cores. This is controlled by setting the `AMICI_IMPORT_NPROCS` environment variable to the number of parallel processes that are to be used (default: 1). Note that for small models this may (slightly) increase import times.

```
__init__(de_model, outdir=None, verbose=False, assume_pow_positivity=False, compiler=None,
         allow_reinit_fixpar_initcond=True, generate_sensitivity_code=True, model_name='model')
```

Generate AMICI C++ files for the DE provided to the constructor.

### Parameters

- **de\_model** (`amici.de_export.DEModel`) – DE model definition
- **outdir** (`typing.Union[pathlib.Path, str, None]`) – see `amici.de_export.DEExporter.set_paths()`

- **verbose** (`typing.Union[bool, int, None]`) – verbosity level for logging, True/False default to `logging.Error/logging.DEBUG`
- **assume\_pow\_positivity** (`typing.Optional[bool]`) – if set to true, a special pow function is used to avoid problems with state variables that may become negative due to numerical errors
- **compiler** (`typing.Optional[str]`) – Absolute path to the compiler executable to be used to build the Python extension, e.g. `/usr/bin/clang`.
- **allow\_reinit\_fixpar\_initcond** (`typing.Optional[bool]`) – see [`amici.de\_export.DDExporter`](#)
- **generate\_sensitivity\_code** (`typing.Optional[bool]`) – specifies whether code required for sensitivity computation will be generated
- **model\_name** (`typing.Optional[str]`) – name of the model to be used during code generation

**compile\_model()**

Compiles the generated code it into a simulatable module

**Return type** `None`

**generate\_model\_code()**

Generates the native C++ code for the loaded model and a Matlab script that can be run to compile a mex file from the C++ code

**Return type** `None`

**set\_name(model\_name)**

Sets the model name

**Parameters** **model\_name** (`str`) – name of the model (may only contain upper and lower case letters, digits and underscores, and must not start with a digit)

**Return type** `None`

**set\_paths(output\_dir=None)**

Set output paths for the model and create if necessary

**Parameters** **output\_dir** (`typing.Union[pathlib.Path, str, None]`) – relative or absolute path where the generated model code is to be placed. If `None`, this will default to `amici-{self.model_name}` in the current working directory. will be created if it does not exist.

**Return type** `None`

**class** `amici.de_export.DEModel(verbose=False, simplify=<function _default_simplify>, cache_simplify=False)`

Defines a Differential Equation as set of ModelQuantities. This class provides general purpose interfaces to compute arbitrary symbolic derivatives that are necessary for model simulation or sensitivity computation.

**Variables**

- **\_differential\_states** – list of differential state variables
- **\_algebraic\_states** – list of algebraic state variables
- **\_observables** – list of observables
- **\_event\_observables** – list of event observables
- **\_sigma\_ys** – list of sigmas for observables

- **\_sigma\_zs** – list of sigmas for event observables
- **\_parameters** – list of parameters
- **\_log\_likelihood\_ys** – list of loglikelihoods for observables
- **\_log\_likelihood\_zs** – list of loglikelihoods for event observables
- **\_log\_likelihood\_rzs** – list of loglikelihoods for event observable regularizations
- **\_expressions** – list of expressions instances
- **\_conservation\_laws** – list of conservation laws
- **\_symboldim\_funs** – define functions that compute model dimensions, these are functions as the underlying symbolic expressions have not been populated at compile time
- **\_eqs** – carries symbolic formulas of the symbolic variables of the model
- **\_sparseeqs** – carries linear list of all symbolic formulas for sparsified variables
- **\_vals** – carries numeric values of symbolic identifiers of the symbolic variables of the model
- **\_names** – carries the names of symbolic identifiers of the symbolic variables of the model
- **\_syms** – carries symbolic identifiers of the symbolic variables of the model
- **\_sparsesyms** – carries linear list of all symbolic identifiers for sparsified variables
- **\_colptrs** – carries column pointers for sparsified variables. See `SUNMatrixContent_Sparse` definition in `sunmatrix/sunmatrix_sparse.h`
- **\_rowvals** – carries row values for sparsified variables. See `SUNMatrixContent_Sparse` definition in `sunmatrix/sunmatrix_sparse.h`
- **\_equation\_prototype** – defines the attribute from which an equation should be generated via list comprehension (see `OEModel._generate_equation()`)
- **\_variable\_prototype** – defines the attribute from which a variable should be generated via list comprehension (see `DEModel._generate_symbol()`)
- **\_value\_prototype** – defines the attribute from which a value should be generated via list comprehension (see `DEModel._generate_value()`)
- **\_total\_derivative\_prototypes** – defines how a total derivative equation is computed for an equation, key defines the name and values should be arguments for `DEModel.totalDerivative()`
- **\_lock\_total\_derivative** – add chainvariables to this set when computing total derivative from a partial derivative call to enforce a partial derivative in the next recursion. prevents infinite recursion
- **\_simplify** – If not `None`, this function will be used to simplify symbolic derivative expressions. Receives sympy expressions as only argument. To apply multiple simplifications, wrap them in a lambda expression.
- **\_x0\_fixedParameters\_idx** – Index list of subset of states for which `x0_fixedParameters` was computed
- **\_w\_recursion\_depth** – recursion depth in `w`, quantified as nilpotency of `dwdw`
- **\_has\_quadratic\_nllh** – whether all observables have a gaussian noise model, i.e. whether `res` and `FIM` make sense.
- **\_code\_printer** – Code printer to generate C++ code
- **\_z2event** – list of event indices for each event observable

**\_\_init\_\_**(*verbose=False, simplify=<function \_default\_simplify>, cache\_simplify=False*)

Create a new DEModel instance.

**Parameters**

- **verbose** (`typing.Union[bool, int, None]`) – verbosity level for logging, True/False default to `logging.DEBUG/logging.ERROR`
- **simplify** (`typing.Optional[typing.Callable]`) – see `DEModel._simplify()`
- **cache\_simplify** (`bool`) – Whether to cache calls to the simplify method. Can e.g. decrease import times for models with events.

**add\_component**(*component, insert\_first=False*)

Adds a new ModelQuantity to the model.

**Parameters**

- **component** (`amici.de_model.ModelQuantity`) – model quantity to be added
- **insert\_first** (`typing.Optional[bool]`) – whether to add quantity first or last, relevant when components may refer to other components of the same type.

**Return type** `None`

**add\_conservation\_law**(*state, total\_abundance, coefficients*)

Adds a new conservation law to the model. A conservation law is defined by the conserved quantity  $T = \sum_i (a_i * x_i)$ , where  $a_i$  are coefficients and  $x_i$  are different state variables.

**Parameters**

- **state** (`sympy.core.symbol.Symbol`) – symbolic identifier of the state that should be replaced by the conservation law ( $x_j$ )
- **total\_abundance** (`sympy.core.symbol.Symbol`) – symbolic identifier of the total abundance ( $T/a_j$ )
- **coefficients** (`typing.Dict[sympy.core.symbol.Symbol, sympy.core.expr.Expr]`) – Dictionary of coefficients  $\{x_i: a_i\}$

**Return type** `None`

**algebraic\_states**()

Get all algebraic states.

**Return type** `typing.List[amici.de_model.AlgebraicState]`

**colptrs**(*name*)

Returns (and constructs if necessary) the column pointers for a sparsified symbolic variable.

**Parameters** **name** (`str`) – name of the symbolic variable

**Return type** `typing.Union[typing.List[sympy.core.numbers.Number], typing.List[typing.List[sympy.core.numbers.Number]]]`

**Returns** list containing the column pointers

**conservation\_law\_has\_multispecies**(*tcl*)

Checks whether a conservation law has multiple species or it just defines one constant species

**Parameters** **tcl** (`amici.de_model.ConservationLaw`) – conservation law

**Return type** `bool`

**Returns** boolean indicating if conservation\_law is not None

**conservation\_laws()**

Get all conservation laws.

**Return type** `typing.List[amici.de_model.ConservationLaw]`

**constants()**

Get all constants.

**Return type** `typing.List[amici.de_model.Constant]`

**differential\_states()**

Get all differential states.

**Return type** `typing.List[amici.de_model.DifferentialState]`

**eq(name)**

Returns (and constructs if necessary) the formulas for a symbolic entity.

**Parameters** **name** (`str`) – name of the symbolic variable

**Return type** `sympy.matrices.dense.MutableDenseMatrix`

**Returns** matrix of symbolic formulas

**event\_observables()**

Get all event observables.

**Return type** `typing.List[amici.de_model.EventObservable]`

**events()**

Get all events.

**Return type** `typing.List[amici.de_model.Event]`

**expressions()**

Get all expressions.

**Return type** `typing.List[amici.de_model.Expression]`

**free\_symbols()**

Returns list of free symbols that appear in RHS and initial conditions.

**Return type** `typing.Set[sympy.core.basic.Basic]`

**generate\_basic\_variables()**

Generates the symbolic identifiers for all variables in `DEModel._variable_prototype`

**Return type** `None`

**get\_appearance\_counts(idxs)**

Counts how often a state appears in the time derivative of another state and expressions for a subset of states

**Parameters** **idxs** (`typing.List[int]`) – list of state indices for which counts are to be computed

**Return type** `typing.List[int]`

**Returns** list of counts for the states ordered according to the provided indices

**get\_conservation\_laws()**

Returns a list of states with conservation law set

**Return type** `typing.List[typing.Tuple[sympy.core.symbol.Symbol, sympy.core.expr.Expr]]`

**Returns** list of state identifiers

**get\_observable\_transformations()**

List of observable transformations

**Return type** `typing.List[amici.import_utils.ObservableTransformation]`

**Returns** list of transformations

**get\_solver\_indices()**

Returns a mapping that maps rdata species indices to solver indices

**Return type** `typing.Dict[int, int]`

**Returns** dictionary mapping rdata species indices to solver indices

**import\_from\_sbml\_importer**(*si*, *compute\_cls=True*)

Imports a model specification from a `amici.sbml_import.SbmlImporter` instance.

**Parameters**

- **si** (`amici.sbml_import.SbmlImporter`) – imported SBML model
- **compute\_cls** (`typing.Optional[bool]`) – whether to compute conservation laws

**Return type** `None`

**is\_ode()**

Check if model is ODE model.

**Return type** `bool`

**log\_likelihood\_rzs()**

Get all event observable regularization log likelihoods.

**Return type** `typing.List[amici.de_model.LogLikelihoodRZ]`

**log\_likelihood\_ys()**

Get all observable log likelihoodss.

**Return type** `typing.List[amici.de_model.LogLikelihoodY]`

**log\_likelihood\_zs()**

Get all event observable log likelihoods.

**Return type** `typing.List[amici.de_model.LogLikelihoodZ]`

**name**(*name*)

Returns (and constructs if necessary) the names of a symbolic variable

**Parameters** **name** (`str`) – name of the symbolic variable

**Return type** `typing.List[str]`

**Returns** list of names

**num\_cons\_law()**

Number of conservation laws.

**Return type** `int`

**Returns** number of conservation laws



**num\_const()**

Number of Constants.

**Return type** `int`

**Returns** number of constant symbols

**num\_eventobs()**

Number of Event Observables.

**Return type** `int`

**Returns** number of event observable symbols

**num\_events()**

Number of Events.

**Return type** `int`

**Returns** number of event symbols (length of the root vector in AMICI)

**num\_expr()**

Number of Expressions.

**Return type** `int`

**Returns** number of expression symbols

**num\_obs()**

Number of Observables.

**Return type** `int`

**Returns** number of observable symbols

**num\_par()**

Number of Parameters.

**Return type** `int`

**Returns** number of parameter symbols

**num\_state\_reinits()**

Number of solver states which would be reinitialized after preequilibration

**Return type** `int`

**Returns** number of state variable symbols with reinitialization

**num\_states\_rdata()**

Number of states.

**Return type** `int`

**Returns** number of state variable symbols

**num\_states\_solver()**

Number of states after applying conservation laws.

**Return type** `int`

**Returns** number of state variable symbols

**observables()**

Get all observables.

**Return type** `typing.List[amici.de_model.Observable]`

**parameters()**

Get all parameters.

**Return type** `typing.List[amici.de_model.Parameter]`

**parse\_events()**

This function checks the right-hand side for roots of Heaviside functions or events, collects the roots, removes redundant roots, and replaces the formulae of the found roots by identifiers of AMICI's Heaviside function implementation in the right-hand side

**Return type** `None`

**rowvals(name)**

Returns (and constructs if necessary) the row values for a sparsified symbolic variable.

**Parameters** **name** (`str`) – name of the symbolic variable

**Return type** `typing.Union[typing.List[sympy.core.numbers.Number], typing.List[typing.List[sympy.core.numbers.Number]]]`

**Returns** list containing the row values

**sigma\_ys()**

Get all observable sigmas.

**Return type** `typing.List[amici.de_model.SigmaY]`

**sigma\_zs()**

Get all event observable sigmas.

**Return type** `typing.List[amici.de_model.SigmaZ]`

**sparseeq(name)**

Returns (and constructs if necessary) the sparsified formulas for a sparsified symbolic variable.

**Parameters** **name** – name of the symbolic variable

**Return type** `sympy.matrices.dense.MutableDenseMatrix`

**Returns** linearized matrix containing the symbolic formulas

**sparsesym(name, force\_generate=True)**

Returns (and constructs if necessary) the sparsified identifiers for a sparsified symbolic variable.

**Parameters**

- **name** (`str`) – name of the symbolic variable
- **force\_generate** (`bool`) – whether the symbols should be generated if not available

**Return type** `typing.List[str]`

**Returns** linearized Matrix containing the symbolic identifiers

**state\_has\_conservation\_law(ix)**

Checks whether the state at specified index has a conservation law set

**Parameters** **ix** (`int`) – state index

**Return type** `bool`

**Returns** boolean indicating if `conservation_law` is not `None`

**state\_has\_fixed\_parameter\_initial\_condition**(*ix*)

Checks whether the state at specified index has a fixed parameter initial condition

**Parameters** *ix* (`int`) – state index

**Return type** `bool`

**Returns** boolean indicating if any of the initial condition free variables is contained in the model constants

**state\_is\_constant**(*ix*)

Checks whether the temporal derivative of the state is zero

**Parameters** *ix* (`int`) – state index

**Return type** `bool`

**Returns** boolean indicating if constant over time

**states**()

Get all states.

**Return type** `typing.List[amici.de_model.State]`

**sym**(*name*)

Returns (and constructs if necessary) the identifiers for a symbolic entity.

**Parameters** *name* (`str`) – name of the symbolic variable

**Return type** `sympy.matrices.dense.MutableDenseMatrix`

**Returns** matrix of symbolic identifiers

**sym\_names**()

Returns a list of names of generated symbolic variables

**Return type** `typing.List[str]`

**Returns** list of names

**sym\_or\_eq**(*name*, *varname*)

Returns symbols or equations depending on whether a given variable appears in the function signature or not.

**Parameters**

- **name** (`str`) – name of function for which the signature should be checked
- **varname** (`str`) – name of the variable which should be contained in the function signature

**Return type** `sympy.matrices.dense.MutableDenseMatrix`

**Returns** the variable symbols if the variable is part of the signature and the variable equations otherwise.

**val**(*name*)

Returns (and constructs if necessary) the numeric values of a symbolic entity

**Parameters** *name* (`str`) – name of the symbolic variable

**Return type** `typing.List[sympy.core.numbers.Number]`

**Returns** list containing the numeric values

```
class amici.de_export.TemplateAmici(template)
```

Template format used in AMICI (see `string.Template` for more details).

**Variables** `delimiter` – delimiter that identifies template variables

```
__init__(template)
```

```
braceidpattern = None
```

```
delimiter = 'TPL_'
```

```
flags = 2
```

```
idpattern = '(?a:[_a-z][_a-z0-9]*)'
```

```
pattern = re.compile('\n TPL_(?:\n (?P<escaped>TPL_) | # Escape sequence of two  
delimiters\n (?P<named>(?:a:[_a-z][_a-z0-9]*)) | # delimiter and a Python  
identifier\n , re.IGNORECASE|re.VERBOSE)
```

```
safe_substitute(mapping={},/, **kws)
```

```
substitute(mapping={},/, **kws)
```

```
amici.de_export.apply_template(source_file, target_file, template_data)
```

Load source file, apply template substitution as provided in `templateData` and save as `targetFile`.

**Parameters**

- **source\_file** (`typing.Union[str, pathlib.Path]`) – relative or absolute path to template file
- **target\_file** (`typing.Union[str, pathlib.Path]`) – relative or absolute path to output file
- **template\_data** (`typing.Dict[str, str]`) – template keywords to substitute (key is template variable without `TemplateAmici.delimiter`)

**Return type** `None`

```
amici.de_export.get_function_extern_declaration(fun, name, ode)
```

Constructs the extern function declaration for a given function

**Parameters**

- **fun** (`str`) – function name
- **name** (`str`) – model name
- **ode** (`bool`) – whether to generate declaration for DAE or ODE

**Return type** `str`

**Returns** C++ function definition string

```
amici.de_export.get_model_override_implementation(fun, name, ode, nobody=False)
```

Constructs `amici::Model::*` override implementation for a given function

**Parameters**

- **fun** (`str`) – function name
- **name** (`str`) – model name
- **nobody** (`bool`) – whether the function has a nontrivial implementation

**Return type** `str`

**Returns** C++ function implementation string

`amici.de_export.get_sunindex_extern_declaration(fun, name, indextype)`

Constructs the function declaration for an index function of a given function

**Parameters**

- **fun** (`str`) – function name
- **name** (`str`) – model name
- **indextype** (`str`) – index function { ‘colptrs’, ‘rowvals’ }

**Return type** `str`

**Returns** C++ function declaration string

`amici.de_export.get_sunindex_override_implementation(fun, name, indextype, nobody=False)`

Constructs the `amici::Model` function implementation for an index function of a given function

**Parameters**

- **fun** (`str`) – function name
- **name** (`str`) – model name
- **indextype** (`str`) – index function { ‘colptrs’, ‘rowvals’ }
- **nobody** (`bool`) – whether the corresponding function has a nontrivial implementation

**Return type** `str`

**Returns** C++ function implementation string

`amici.de_export.is_valid_identififer(x)`

Check whether *x* is a valid identifier for conditions, parameters, observables... . Identifiers may only contain upper and lower case letters, digits and underscores, and must not start with a digit.

**Parameters** **x** (`str`) – string to check

**Return type** `bool`

**Returns** True if valid, False otherwise

`amici.de_export.remove_argument_types(signature)`

Strips argument types from a function signature

**Parameters** **signature** (`str`) – function signature

**Return type** `str`

**Returns** string that can be used to construct function calls with the same variable names and ordering as in the function signature

`amici.de_export.smart_is_zero_matrix(x)`

A faster implementation of `sympy`’s `is_zero_matrix`

Avoids repeated indexer type checks and double iteration to distinguish False/None. Found to be about 100x faster for large matrices.

**Parameters** **x** (`typing.Union[sympy.matrices.dense.MutableDenseMatrix, sympy.matrices.sparse.MutableSparseMatrix]`) – Matrix to check

**Return type** `bool`

`amici.de_export.smart_jacobian(eq, sym_var)`

Wrapper around symbolic jacobian with some additional checks that reduce computation time for large matrices

**Parameters**

- `eq` (`sympy.matrices.dense.MutableDenseMatrix`) – equation
- `sym_var` (`sympy.matrices.dense.MutableDenseMatrix`) – differentiation variable

**Return type** `sympy.matrices.sparse.MutableSparseMatrix`

**Returns** jacobian of eq wrt sym\_var

`amici.de_export.smart_multiply(x, y)`

Wrapper around symbolic multiplication with some additional checks that reduce computation time for large matrices

**Parameters**

- `x` (`typing.Union[sympy.matrices.dense.MutableDenseMatrix, sympy.matrices.sparse.MutableSparseMatrix]`) – educt 1
- `y` (`sympy.matrices.dense.MutableDenseMatrix`) – educt 2

**Return type** `typing.Union[sympy.matrices.dense.MutableDenseMatrix, sympy.matrices.sparse.MutableSparseMatrix]`

**Returns** product

`amici.de_export.var_in_function_signature(name, varname, ode)`

Checks if the values for a symbolic variable is passed in the signature of a function

**Parameters**

- `name` (`str`) – name of the function
- `varname` (`str`) – name of the symbolic variable
- `ode` (`bool`) – whether to check the ODE or DAE signature

**Return type** `bool`

**Returns** boolean indicating whether the variable occurs in the function signature

## 10.5.12 amici.de\_model

Objects for AMICI's internal differential equation model representation

## Classes

<i>AlgebraicEquation</i> (identifier, value)	An AlgebraicEquation defines an algebraic equation.
<i>AlgebraicState</i> (identifier, name, init)	An AlgebraicState defines an entity that is algebraically determined
<i>ConservationLaw</i> (identifier, name, value, ...)	A conservation law defines the absolute the total amount of a (weighted) sum of states
<i>Constant</i> (identifier, name, value)	A Constant is a fixed variable in the model with respect to which sensitivities cannot be computed, abbreviated by <b>k</b> .
<i>DifferentialState</i> (identifier, name, init, dt)	A State variable defines an entity that evolves with time according to the provided time derivative, abbreviated by <b>x</b> .
<i>Event</i> (identifier, name, value, state_update)	An Event defines either a SBML event or a root of the argument of a Heaviside function.
<i>EventObservable</i> (identifier, name, value, event)	An Event Observable links model simulations to event related experimental measurements, abbreviated by <b>z</b> .
<i>Expression</i> (identifier, name, value)	An Expression is a recurring elements in symbolic formulas.
<i>LogLikelihood</i> (identifier, name, value)	A LogLikelihood defines the distance between measurements and experiments for a particular observable.
<i>LogLikelihoodRZ</i> (identifier, name, value)	Loglikelihood for event observables regularization
<i>LogLikelihoodY</i> (identifier, name, value)	Loglikelihood for observables
<i>LogLikelihoodZ</i> (identifier, name, value)	Loglikelihood for event observables
<i>ModelQuantity</i> (identifier, name, value)	Base class for model components
<i>Observable</i> (identifier, name, value[, ...])	An Observable links model simulations to experimental measurements, abbreviated by <b>y</b> .
<i>Parameter</i> (identifier, name, value)	A Parameter is a free variable in the model with respect to which sensitivities may be computed, abbreviated by <b>p</b> .
<i>Sigma</i> (identifier, name, value)	A Standard Deviation Sigma rescales the distance between simulations and measurements when computing residuals or objective functions, abbreviated by <b>sigma{y,z}</b> .
<i>SigmaY</i> (identifier, name, value)	Standard deviation for observables
<i>SigmaZ</i> (identifier, name, value)	Standard deviation for event observables
<i>State</i> (identifier, name, value)	Base class for differential and algebraic model states

**class** amici.de\_model.**AlgebraicEquation**(*identifier*, *value*)

An AlgebraicEquation defines an algebraic equation.

**\_\_init\_\_**(*identifier*, *value*)

Create a new AlgebraicEquation instance.

**Parameters** *value* (`sympy.core.expr.Expr`) – formula of the algebraic equation, solution is given by `formula == 0`

**get\_free\_symbols**()

**get\_id**()

ModelQuantity identifier

**Return type** `sympy.core.symbol.Symbol`

**Returns** identifier of the ModelQuantity

**get\_name()**

ModelQuantity name

**Return type** `str`

**Returns** name of the ModelQuantity

**get\_val()**

ModelQuantity value

**Return type** `sympy.core.expr.Expr`

**Returns** value of the ModelQuantity

**set\_val(val)**

Set ModelQuantity value

**Returns** value of the ModelQuantity

**class** `amici.de_model.AlgebraicState(identifier, name, init)`

An AlgebraicState defines an entity that is algebraically determined

**\_\_init\_\_**(*identifier, name, init*)

Create a new AlgebraicState instance.

**Parameters**

- **identifier** (`sympy.core.symbol.Symbol`) – unique identifier of the AlgebraicState
- **name** (`str`) – individual name of the AlgebraicState (does not need to be unique)
- **init** (`sympy.core.expr.Expr`) – initial value of the AlgebraicState

**get\_dx\_rdata\_dx\_solver**(*state\_id*)

Returns the expression that allows computation of `dx_rdata_dx_solver` for this state, accounting for conservation laws.

**Returns** `dx_rdata_dx_solver` expression

**get\_free\_symbols()**

**get\_id()**

ModelQuantity identifier

**Return type** `sympy.core.symbol.Symbol`

**Returns** identifier of the ModelQuantity

**get\_name()**

ModelQuantity name

**Return type** `str`

**Returns** name of the ModelQuantity

**get\_val()**

ModelQuantity value

**Return type** `sympy.core.expr.Expr`

**Returns** value of the ModelQuantity



**get\_x\_rdata()**

Returns the expression that allows computation of x\_rdata for this state, accounting for conservation laws.

**Returns** x\_rdata expression

**has\_conservation\_law()**

Checks whether this state has a conservation law assigned.

**Returns** True if assigned, False otherwise

**set\_val(val)**

Set ModelQuantity value

**Returns** value of the ModelQuantity

**class amici.de\_model.ConservationLaw(identifier, name, value, coefficients, state\_id)**

A conservation law defines the absolute the total amount of a (weighted) sum of states

**\_\_init\_\_(identifier, name, value, coefficients, state\_id)**

Create a new ConservationLaw instance.

**Parameters**

- **identifier** (`sympy.core.symbol.Symbol`) – unique identifier of the ConservationLaw
- **name** (`str`) – individual name of the ConservationLaw (does not need to be unique)
- **value** (`sympy.core.expr.Expr`) – formula (sum of states)
- **coefficients** (`typing.Dict[sympy.core.symbol.Symbol, sympy.core.expr.Expr]`) – coefficients of the states in the sum
- **state\_id** (`sympy.core.symbol.Symbol`) – identifier of the state that this conservation law replaces

**get\_id()**

ModelQuantity identifier

**Return type** `sympy.core.symbol.Symbol`

**Returns** identifier of the ModelQuantity

**get\_name()**

ModelQuantity name

**Return type** `str`

**Returns** name of the ModelQuantity

**get\_ncoeff(state\_id)**

Computes the normalized coefficient  $a_i/a_j$  where  $i$  is the index of the provided state\_id and  $j$  is the index of the state that is replaced by this conservation law. This can be used to compute both  $dtotal\_cl/dx\_rdata$  ( $=ncoeff$ ) and  $dx\_rdata/dx\_solver$  ( $=-ncoeff$ ).

**Parameters** **state\_id** – identifier of the state

**Return type** `typing.Union[sympy.core.expr.Expr, int, float]`

**Returns** normalized coefficient of the state

**get\_val()**

ModelQuantity value

**Return type** `sympy.core.expr.Expr`

**Returns** value of the ModelQuantity

**get\_x\_rdata()**

Returns the expression that allows computation of x\_rdata for the state that this conservation law replaces.

**Returns** x\_rdata expression

**set\_val(val)**

Set ModelQuantity value

**Returns** value of the ModelQuantity

**class** amici.de\_model.Constant(*identifier, name, value*)

A Constant is a fixed variable in the model with respect to which sensitivities cannot be computed, abbreviated by k.

**\_\_init\_\_**(*identifier, name, value*)

Create a new Expression instance.

**Parameters**

- **identifier** (`sympy.core.symbol.Symbol`) – unique identifier of the Constant
- **name** (`str`) – individual name of the Constant (does not need to be unique)
- **value** (`numbers.Number`) – numeric value

**get\_id()**

ModelQuantity identifier

**Return type** `sympy.core.symbol.Symbol`

**Returns** identifier of the ModelQuantity

**get\_name()**

ModelQuantity name

**Return type** `str`

**Returns** name of the ModelQuantity

**get\_val()**

ModelQuantity value

**Return type** `sympy.core.expr.Expr`

**Returns** value of the ModelQuantity

**set\_val(val)**

Set ModelQuantity value

**Returns** value of the ModelQuantity

**class** amici.de\_model.DifferentialState(*identifier, name, init, dt*)

A State variable defines an entity that evolves with time according to the provided time derivative, abbreviated by x.

**Variables**

- **\_conservation\_law** – algebraic formula that allows computation of this state according to a conservation law
- **\_dt** – algebraic formula that defines the temporal derivative of this state

**\_\_init\_\_**(*identifier, name, init, dt*)

Create a new State instance. Extends *ModelQuantity.\_\_init\_\_()* by dt

**Parameters**

- **identifier** (*sympy.core.symbol.Symbol*) – unique identifier of the state
- **name** (*str*) – individual name of the state (does not need to be unique)
- **init** (*sympy.core.expr.Expr*) – initial value
- **dt** (*sympy.core.expr.Expr*) – time derivative

**get\_dt**()

Gets the time derivative

**Return type** *sympy.core.expr.Expr*

**Returns** time derivative

**get\_dx\_rdata\_dx\_solver**(*state\_id*)

Returns the expression that allows computation of *dx\_rdata\_dx\_solver* for this state, accounting for conservation laws.

**Returns** *dx\_rdata\_dx\_solver* expression

**get\_free\_symbols**()

Gets the set of free symbols in time derivative and initial conditions

**Return type** *typing.Set[sympy.core.basic.Basic]*

**Returns** free symbols

**get\_id**()

ModelQuantity identifier

**Return type** *sympy.core.symbol.Symbol*

**Returns** identifier of the ModelQuantity

**get\_name**()

ModelQuantity name

**Return type** *str*

**Returns** name of the ModelQuantity

**get\_val**()

ModelQuantity value

**Return type** *sympy.core.expr.Expr*

**Returns** value of the ModelQuantity

**get\_x\_rdata**()

Returns the expression that allows computation of *x\_rdata* for this state, accounting for conservation laws.

**Returns** *x\_rdata* expression

**has\_conservation\_law**()

Checks whether this state has a conservation law assigned.

**Returns** True if assigned, False otherwise

**set\_conservation\_law**(*law*)

Sets the conservation law of a state.

If a conservation law is set, the respective state will be replaced by an algebraic formula according to the respective conservation law.

**Parameters** *law* (`amici.de_model.ConservationLaw`) – linear sum of states that if added to this state remain constant over time

**Return type** `None`

**set\_dt**(*dt*)

Sets the time derivative

**Parameters** *dt* (`sympy.core.expr.Expr`) – time derivative

**Return type** `None`

**set\_val**(*val*)

Set ModelQuantity value

**Returns** value of the ModelQuantity

**class** `amici.de_model.Event`(*identifier, name, value, state\_update, initial\_value=True*)

An Event defines either a SBML event or a root of the argument of a Heaviside function. The Heaviside functions will be tracked via the vector *h* during simulation and are needed to inform the solver about a discontinuity in either the right-hand side or the states themselves, causing a reinitialization of the solver.

**\_\_init\_\_**(*identifier, name, value, state\_update, initial\_value=True*)

Create a new Event instance.

**Parameters**

- **identifier** (`sympy.core.symbol.Symbol`) – unique identifier of the Event
- **name** (`str`) – individual name of the Event (does not need to be unique)
- **value** (`sympy.core.expr.Expr`) – formula for the root / trigger function
- **state\_update** (`typing.Optional[sympy.core.expr.Expr]`) – formula for the bolus function (None for Heaviside functions, zero vector for events without bolus)
- **initial\_value** (`typing.Optional[bool]`) – initial boolean value of the trigger function at *t0*. If set to *False*, events may trigger at *t==t0*, otherwise not.

**get\_id**()

ModelQuantity identifier

**Return type** `sympy.core.symbol.Symbol`

**Returns** identifier of the ModelQuantity

**get\_initial\_value**()

Return the initial value for the root function.

**Return type** `bool`

**Returns** initial value formula

**get\_name**()

ModelQuantity name

**Return type** `str`

**Returns** name of the ModelQuantity

**get\_val()**

ModelQuantity value

**Return type** `sympy.core.expr.Expr`

**Returns** value of the ModelQuantity

**set\_val(val)**

Set ModelQuantity value

**Returns** value of the ModelQuantity

**class** `amici.de_model.EventObservable(identifier, name, value, event, measurement_symbol=None, transformation='lin')`

An Event Observable links model simulations to event related experimental measurements, abbreviated by z.

**Variables** `_event` – symbolic event identifier

**\_\_init\_\_**(*identifier, name, value, event, measurement\_symbol=None, transformation='lin'*)

Create a new EventObservable instance.

**Parameters**

- **identifier** (`sympy.core.symbol.Symbol`) – See `Observable.__init__()`.
- **name** (`str`) – See `Observable.__init__()`.
- **value** (`sympy.core.expr.Expr`) – See `Observable.__init__()`.
- **transformation** (`typing.Optional[amici.import_utils.ObservableTransformation]`) – See `Observable.__init__()`.
- **event** (`sympy.core.symbol.Symbol`) – Symbolic identifier of the corresponding event.

**get\_event()**

Get the symbolic identifier of the corresponding event.

**Return type** `sympy.core.symbol.Symbol`

**Returns** symbolic identifier

**get\_id()**

ModelQuantity identifier

**Return type** `sympy.core.symbol.Symbol`

**Returns** identifier of the ModelQuantity

**get\_measurement\_symbol()**

**Return type** `sympy.core.symbol.Symbol`

**get\_name()**

ModelQuantity name

**Return type** `str`

**Returns** name of the ModelQuantity

**get\_regularization\_symbol()**

**Return type** `sympy.core.symbol.Symbol`

**get\_val()**

ModelQuantity value

**Return type** `sympy.core.expr.Expr`**Returns** value of the ModelQuantity**set\_val(val)**

Set ModelQuantity value

**Returns** value of the ModelQuantity**class** `amici.de_model.Expression(identifier, name, value)`

An Expression is a recurring elements in symbolic formulas. Specifying this may yield more compact expression which may lead to substantially shorter model compilation times, but may also reduce model simulation time. Abbreviated by `w`.

**\_\_init\_\_(identifier, name, value)**

Create a new Expression instance.

**Parameters**

- **identifier** (`sympy.core.symbol.Symbol`) – unique identifier of the Expression
- **name** (`str`) – individual name of the Expression (does not need to be unique)
- **value** (`sympy.core.expr.Expr`) – formula

**get\_id()**

ModelQuantity identifier

**Return type** `sympy.core.symbol.Symbol`**Returns** identifier of the ModelQuantity**get\_name()**

ModelQuantity name

**Return type** `str`**Returns** name of the ModelQuantity**get\_val()**

ModelQuantity value

**Return type** `sympy.core.expr.Expr`**Returns** value of the ModelQuantity**set\_val(val)**

Set ModelQuantity value

**Returns** value of the ModelQuantity**class** `amici.de_model.LogLikelihoodRZ(identifier, name, value)`

Loglikelihood for event observables regularization

**\_\_init\_\_(identifier, name, value)**

Create a new Expression instance.

**Parameters**

- **identifier** (`sympy.core.symbol.Symbol`) – unique identifier of the LogLikelihood
- **name** (`str`) – individual name of the LogLikelihood (does not need to be unique)

---

- **value** (`sympy.core.expr.Expr`) – formula

**get\_id()**  
ModelQuantity identifier

**Return type** `sympy.core.symbol.Symbol`

**Returns** identifier of the ModelQuantity

**get\_name()**  
ModelQuantity name

**Return type** `str`

**Returns** name of the ModelQuantity

**get\_val()**  
ModelQuantity value

**Return type** `sympy.core.expr.Expr`

**Returns** value of the ModelQuantity

**set\_val(val)**  
Set ModelQuantity value

**Returns** value of the ModelQuantity

**class amici.de\_model.LogLikelihoodY(identifier, name, value)**  
Loglikelihood for observables

**\_\_init\_\_(identifier, name, value)**  
Create a new Expression instance.

**Parameters**

- **identifier** (`sympy.core.symbol.Symbol`) – unique identifier of the LogLikelihood
- **name** (`str`) – individual name of the LogLikelihood (does not need to be unique)
- **value** (`sympy.core.expr.Expr`) – formula

**get\_id()**  
ModelQuantity identifier

**Return type** `sympy.core.symbol.Symbol`

**Returns** identifier of the ModelQuantity

**get\_name()**  
ModelQuantity name

**Return type** `str`

**Returns** name of the ModelQuantity

**get\_val()**  
ModelQuantity value

**Return type** `sympy.core.expr.Expr`

**Returns** value of the ModelQuantity

**set\_val**(*val*)

Set ModelQuantity value

**Returns** value of the ModelQuantity

**class** amici.de\_model.**LogLikelihoodZ**(*identifier, name, value*)

Loglikelihood for event observables

**\_\_init\_\_**(*identifier, name, value*)

Create a new Expression instance.

**Parameters**

- **identifier** (`sympy.core.symbol.Symbol`) – unique identifier of the LogLikelihood
- **name** (`str`) – individual name of the LogLikelihood (does not need to be unique)
- **value** (`sympy.core.expr.Expr`) – formula

**get\_id**()

ModelQuantity identifier

**Return type** `sympy.core.symbol.Symbol`

**Returns** identifier of the ModelQuantity

**get\_name**()

ModelQuantity name

**Return type** `str`

**Returns** name of the ModelQuantity

**get\_val**()

ModelQuantity value

**Return type** `sympy.core.expr.Expr`

**Returns** value of the ModelQuantity

**set\_val**(*val*)

Set ModelQuantity value

**Returns** value of the ModelQuantity

**class** amici.de\_model.**ModelQuantity**(*identifier, name, value*)

Base class for model components

**\_\_init\_\_**(*identifier, name, value*)

Create a new ModelQuantity instance.

**Parameters**

- **identifier** (`sympy.core.symbol.Symbol`) – unique identifier of the quantity
- **name** (`str`) – individual name of the quantity (does not need to be unique)
- **value** (`typing.Union[typing.SupportsFloat, numbers.Number, sympy.core.expr.Expr]`) – either formula, numeric value or initial value

**get\_id**()

ModelQuantity identifier

**Return type** `sympy.core.symbol.Symbol`



**Returns** identifier of the ModelQuantity

**get\_name()**

ModelQuantity name

**Return type** `str`

**Returns** name of the ModelQuantity

**get\_val()**

ModelQuantity value

**Return type** `sympy.core.expr.Expr`

**Returns** value of the ModelQuantity

**set\_val(val)**

Set ModelQuantity value

**Returns** value of the ModelQuantity

**class** `amici.de_model.Observable(identifier, name, value, measurement_symbol=None, transformation=ObservableTransformation.LIN)`

An Observable links model simulations to experimental measurements, abbreviated by `y`.

#### Variables

- **`_measurement_symbol`** – sympy symbol used in the objective function to represent measurements to this observable
- **`trafo`** – observable transformation, only applies when evaluating objective function or residuals

**`__init__`**(`identifier, name, value, measurement_symbol=None, transformation=ObservableTransformation.LIN`)

Create a new Observable instance.

#### Parameters

- **`identifier`** (`sympy.core.symbol.Symbol`) – unique identifier of the Observable
- **`name`** (`str`) – individual name of the Observable (does not need to be unique)
- **`value`** (`sympy.core.expr.Expr`) – formula
- **`transformation`** (`typing.Optional[amici.import_utils.ObservableTransformation]`) – observable transformation, only applies when evaluating objective function or residuals

**get\_id()**

ModelQuantity identifier

**Return type** `sympy.core.symbol.Symbol`

**Returns** identifier of the ModelQuantity

**get\_measurement\_symbol()**

**Return type** `sympy.core.symbol.Symbol`

**get\_name()**

ModelQuantity name

**Return type** `str`

**Returns** name of the ModelQuantity

**get\_regularization\_symbol()**

**Return type** `sympy.core.symbol.Symbol`

**get\_val()**

ModelQuantity value

**Return type** `sympy.core.expr.Expr`

**Returns** value of the ModelQuantity

**set\_val(val)**

Set ModelQuantity value

**Returns** value of the ModelQuantity

**class** `amici.de_model.Parameter(identifier, name, value)`

A Parameter is a free variable in the model with respect to which sensitivities may be computed, abbreviated by p.

**\_\_init\_\_**(*identifier, name, value*)

Create a new Expression instance.

**Parameters**

- **identifier** (`sympy.core.symbol.Symbol`) – unique identifier of the Parameter
- **name** (`str`) – individual name of the Parameter (does not need to be unique)
- **value** (`numbers.Number`) – numeric value

**get\_id()**

ModelQuantity identifier

**Return type** `sympy.core.symbol.Symbol`

**Returns** identifier of the ModelQuantity

**get\_name()**

ModelQuantity name

**Return type** `str`

**Returns** name of the ModelQuantity

**get\_val()**

ModelQuantity value

**Return type** `sympy.core.expr.Expr`

**Returns** value of the ModelQuantity

**set\_val(val)**

Set ModelQuantity value

**Returns** value of the ModelQuantity

**class** `amici.de_model.SigmaY(identifier, name, value)`

Standard deviation for observables

**\_\_init\_\_**(*identifier, name, value*)

Create a new Standard Deviation instance.

**Parameters**

- **identifier** (`sympy.core.symbol.Symbol`) – unique identifier of the Standard Deviation
- **name** (`str`) – individual name of the Standard Deviation (does not need to be unique)
- **value** (`sympy.core.expr.Expr`) – formula

**get\_id**()

ModelQuantity identifier

**Return type** `sympy.core.symbol.Symbol`

**Returns** identifier of the ModelQuantity

**get\_name**()

ModelQuantity name

**Return type** `str`

**Returns** name of the ModelQuantity

**get\_val**()

ModelQuantity value

**Return type** `sympy.core.expr.Expr`

**Returns** value of the ModelQuantity

**set\_val**(*val*)

Set ModelQuantity value

**Returns** value of the ModelQuantity

**class** amici.de\_model.**SigmaZ**(*identifier, name, value*)

Standard deviation for event observables

**\_\_init\_\_**(*identifier, name, value*)

Create a new Standard Deviation instance.

**Parameters**

- **identifier** (`sympy.core.symbol.Symbol`) – unique identifier of the Standard Deviation
- **name** (`str`) – individual name of the Standard Deviation (does not need to be unique)
- **value** (`sympy.core.expr.Expr`) – formula

**get\_id**()

ModelQuantity identifier

**Return type** `sympy.core.symbol.Symbol`

**Returns** identifier of the ModelQuantity

**get\_name**()

ModelQuantity name

**Return type** `str`

**Returns** name of the ModelQuantity

**get\_val()**

ModelQuantity value

**Return type** `sympy.core.expr.Expr`

**Returns** value of the ModelQuantity

**set\_val(val)**

Set ModelQuantity value

**Returns** value of the ModelQuantity

**class** `amici.de_model.State(identifier, name, value)`

Base class for differential and algebraic model states

**\_\_init\_\_(identifier, name, value)**

Create a new ModelQuantity instance.

**Parameters**

- **identifier** (`sympy.core.symbol.Symbol`) – unique identifier of the quantity
- **name** (`str`) – individual name of the quantity (does not need to be unique)
- **value** (`typing.Union[typing.SupportsFloat, numbers.Number, sympy.core.expr.Expr]`) – either formula, numeric value or initial value

**get\_dx\_rdata\_dx\_solver(state\_id)**

Returns the expression that allows computation of `dx_rdata_dx_solver` for this state, accounting for conservation laws.

**Returns** `dx_rdata_dx_solver` expression

**get\_id()**

ModelQuantity identifier

**Return type** `sympy.core.symbol.Symbol`

**Returns** identifier of the ModelQuantity

**get\_name()**

ModelQuantity name

**Return type** `str`

**Returns** name of the ModelQuantity

**get\_val()**

ModelQuantity value

**Return type** `sympy.core.expr.Expr`

**Returns** value of the ModelQuantity

**get\_x\_rdata()**

Returns the expression that allows computation of `x_rdata` for this state, accounting for conservation laws.

**Returns** `x_rdata` expression

**abstract has\_conservation\_law()**

Checks whether this state has a conservation law assigned.

**Returns** True if assigned, False otherwise

**set\_val(val)**

Set ModelQuantity value

**Returns** value of the ModelQuantity

## 10.5.13 amici.plotting

### Plotting

Plotting related functions

### Functions

<code>plotObservableTrajectories(rdata[, ...])</code>	Plot observable trajectories
<code>plotStateTrajectories(rdata[, ...])</code>	Plot state trajectories
<code>plot_expressions(exprs, rdata)</code>	Plot the given expressions evaluated on the given simulation outputs.
<code>plot_jacobian(rdata)</code>	Plot Jacobian as heatmap.
<code>plot_observable_trajectories(rdata[, ...])</code>	Plot observable trajectories
<code>plot_state_trajectories(rdata[, ...])</code>	Plot state trajectories

`amici.plotting.plotObservableTrajectories(rdata, observable_indices=None, ax=None, model=None, prefer_names=True)`

Plot observable trajectories

#### Parameters

- **rdata** (`amici.numpy.ReturnDataView`) – AMICI simulation results as returned by `amici.amici.runAmiciSimulation()`
- **observable\_indices** (`typing.Optional[typing.Iterable[int]]`) – Indices of observables for which trajectories are to be plotted
- **ax** (`typing.Optional[matplotlib.axes._axes.Axes]`) – matplotlib Axes instance to plot into
- **model** (`typing.Optional[amici.amici.Model]`) – amici model instance
- **prefer\_names** (`bool`) – Whether observables names should be preferred over IDs, if available.

**Return type** `None`

`amici.plotting.plotStateTrajectories(rdata, state_indices=None, ax=None, model=None, prefer_names=True)`

Plot state trajectories

#### Parameters

- **rdata** (`amici.numpy.ReturnDataView`) – AMICI simulation results as returned by `amici.amici.runAmiciSimulation()`

- **state\_indices** (`typing.Optional[typing.Iterable[int]]`) – Indices of states for which trajectories are to be plotted
- **ax** (`typing.Optional[matplotlib.axes._axes.Axes]`) – matplotlib Axes instance to plot into
- **model** (`typing.Optional[amici.amici.Model]`) – amici model instance
- **prefer\_names** (`bool`) – Whether state names should be preferred over IDs, if available.

**Return type** `None`

`amici.plotting.plot_expressions(exprs, rdata)`

Plot the given expressions evaluated on the given simulation outputs.

**Parameters**

- **exprs** (`typing.Union[typing.Sequence[typing.Union[str, sympy.core.expr.Expr]], str, sympy.core.expr.Expr]`) – A symbolic expression, e.g. a sympy expression or a string that can be sympified. Can include state variable, expression, and observable IDs, depending on whether the respective data is available in the simulation results. Parameters are not yet supported.
- **rdata** (`amici.numpy.ReturnDataView`) – The simulation results.

**Return type** `None`

`amici.plotting.plot_jacobian(rdata)`

Plot Jacobian as heatmap.

`amici.plotting.plot_observable_trajectories(rdata, observable_indices=None, ax=None, model=None, prefer_names=True)`

Plot observable trajectories

**Parameters**

- **rdata** (`amici.numpy.ReturnDataView`) – AMICI simulation results as returned by `amici.amici.runAmiciSimulation()`
- **observable\_indices** (`typing.Optional[typing.Iterable[int]]`) – Indices of observables for which trajectories are to be plotted
- **ax** (`typing.Optional[matplotlib.axes._axes.Axes]`) – matplotlib Axes instance to plot into
- **model** (`typing.Optional[amici.amici.Model]`) – amici model instance
- **prefer\_names** (`bool`) – Whether observables names should be preferred over IDs, if available.

**Return type** `None`

`amici.plotting.plot_state_trajectories(rdata, state_indices=None, ax=None, model=None, prefer_names=True)`

Plot state trajectories

**Parameters**

- **rdata** (`amici.numpy.ReturnDataView`) – AMICI simulation results as returned by `amici.amici.runAmiciSimulation()`
- **state\_indices** (`typing.Optional[typing.Iterable[int]]`) – Indices of states for which trajectories are to be plotted

- **ax** (`typing.Optional[matplotlib.axes._axes.Axes]`) – matplotlib Axes instance to plot into
- **model** (`typing.Optional[amici.amici.Model]`) – amici model instance
- **prefer\_names** (`bool`) – Whether state names should be preferred over IDs, if available.

**Return type** `None`

## 10.5.14 amici.pandas

### Pandas Wrappers

This module contains convenience wrappers that allow for easy interconversion between C++ objects from *amici* and pandas DataFrames

### Functions

<code>constructEdataFromDataFrame(df, model, condition)</code>	Constructs an ExpData instance according to the provided Model and DataFrame.
<code>getDataObservablesAsDataFrame(model, edata_list)</code>	Write Observables from experimental data as DataFrame.
<code>getEdataFromDataFrame(model, df[, by_id])</code>	Constructs a ExpData instances according to the provided Model and DataFrame.
<code>getResidualsAsDataFrame(model, edata_list, ...)</code>	Convert a list of ReturnData and ExpData to pandas DataFrame with residuals.
<code>getSimulationObservablesAsDataFrame(model, ...)</code>	Write Observables from simulation results as DataFrame.
<code>getSimulationStatesAsDataFrame(model, ..., ...)</code>	Get model state according to lists of ReturnData and ExpData.
<code>get_expressions_as_dataframe(model, ..., by_id)</code>	Get values of model expressions from lists of ReturnData as DataFrame.

`amici.pandas.getDataObservablesAsDataFrame(model, edata_list, by_id=False)`

Write Observables from experimental data as DataFrame.

#### Parameters

- **model** (`typing.Union[amici.amici.ModelPtr, amici.amici.Model]`) – Model instance.
- **edata\_list** (`typing.Union[typing.List[amici.amici.ExpData], typing.List[amici.amici.ExpDataPtr], amici.amici.ExpData, amici.amici.ExpDataPtr]`) – list of ExpData instances with experimental data. May also be a single ExpData instance.
- **by\_id** (`typing.Optional[bool]`) – If True, uses observable ids as column names in the generated DataFrame, otherwise the possibly more descriptive observable names are used.

**Return type** `pandas.core.frame.DataFrame`

**Returns** pandas DataFrame with conditions/timepoints as rows and observables as columns.

`amici.pandas.getEdataFromDataFrame(model, df, by_id=False)`

Constructs a `ExpData` instances according to the provided `Model` and `DataFrame`.

**Parameters**

- **df** (`pandas.core.frame.DataFrame`) – dataframe with Observable Names/Ids, FixedParameter Names/Ids and time as columns. Standard deviations may be specified by appending ‘\_std’ as suffix. Preequilibration fixedParameters may be specified by appending ‘\_preeq’ as suffix. Presimulation fixedParameters may be specified by appending ‘\_presim’ as suffix. Presimulation time may be specified as ‘t\_presim’ column.
- **model** (`typing.Union[amici.amici.ModelPtr, amici.amici.Model]`) – Model instance.
- **by\_id** (`typing.Optional[bool]`) – Whether the column names in `df` are based on ids or names, corresponding to how the dataframe was created in the first place.

**Return type** `typing.List[amici.amici.ExpData]`

**Returns** list of `ExpData` instances.

`amici.pandas.getResidualsAsDataFrame(model, edata_list, rdata_list, by_id=False)`

Convert a list of `ReturnData` and `ExpData` to pandas `DataFrame` with residuals.

**Parameters**

- **model** (`amici.amici.Model`) – Model instance.
- **edata\_list** (`typing.Union[typing.List[amici.amici.ExpData], typing.List[amici.amici.ExpDataPtr], amici.amici.ExpData, amici.amici.ExpDataPtr]`) – list of `ExpData` instances with experimental data. May also be a single `ExpData` instance.
- **rdata\_list** (`typing.Union[typing.List[amici.numpy.ReturnDataView], amici.numpy.ReturnDataView]`) – list of `ReturnData` instances corresponding to `ExpData`. May also be a single `ReturnData` instance.
- **by\_id** (`typing.Optional[bool]`) – bool, optional (default = False) If True, ids are used as identifiers, otherwise the possibly more descriptive names.

**Return type** `pandas.core.frame.DataFrame`

**Returns** pandas `DataFrame` with conditions and residuals.

`amici.pandas.getSimulationObservablesAsDataFrame(model, edata_list, rdata_list, by_id=False)`

Write Observables from simulation results as `DataFrame`.

**Parameters**

- **model** (`amici.amici.Model`) – Model instance.
- **edata\_list** (`typing.Union[typing.List[amici.amici.ExpData], typing.List[amici.amici.ExpDataPtr], amici.amici.ExpData, amici.amici.ExpDataPtr]`) – list of `ExpData` instances with experimental data. May also be a single `ExpData` instance.
- **rdata\_list** (`typing.Union[typing.List[amici.numpy.ReturnDataView], amici.numpy.ReturnDataView]`) – list of `ReturnData` instances corresponding to `ExpData`. May also be a single `ReturnData` instance.
- **by\_id** (`typing.Optional[bool]`) – If True, ids are used as identifiers, otherwise the possibly more descriptive names.

**Return type** `pandas.core.frame.DataFrame`



**Returns** pandas DataFrame with conditions/timepoints as rows and observables as columns.

`amici.pandas.getSimulationStatesAsDataFrame(model, edata_list, rdata_list, by_id=False)`

Get model state according to lists of ReturnData and ExpData.

#### Parameters

- **model** (`amici.amici.Model`) – Model instance.
- **edata\_list** (`typing.Union[typing.List[amici.amici.ExpData], typing.List[amici.amici.ExpDataPtr], amici.amici.ExpData, amici.amici.ExpDataPtr]`) – list of ExpData instances with experimental data. May also be a single ExpData instance.
- **rdata\_list** (`typing.Union[typing.List[amici.numpy.ReturnDataView], amici.numpy.ReturnDataView]`) – list of ReturnData instances corresponding to ExpData. May also be a single ReturnData instance.
- **by\_id** (`typing.Optional[bool]`) – If True, ids are used as identifiers, otherwise the possibly more descriptive names.

**Return type** `pandas.core.frame.DataFrame`

**Returns** pandas DataFrame with conditions/timepoints as rows and state variables as columns.

`amici.pandas.get_expressions_as_dataframe(model, edata_list, rdata_list, by_id=False)`

Get values of model expressions from lists of ReturnData as DataFrame.

#### Parameters

- **model** (`amici.amici.Model`) – Model instance.
- **edata\_list** (`typing.Union[typing.List[amici.amici.ExpData], typing.List[amici.amici.ExpDataPtr], amici.amici.ExpData, amici.amici.ExpDataPtr]`) – list of ExpData instances with experimental data. May also be a single ExpData instance.
- **rdata\_list** (`typing.Union[typing.List[amici.numpy.ReturnDataView], amici.numpy.ReturnDataView]`) – list of ReturnData instances corresponding to ExpData. May also be a single ReturnData instance.
- **by\_id** (`typing.Optional[bool]`) – If True, ids are used as identifiers, otherwise the possibly more descriptive names.

**Return type** `pandas.core.frame.DataFrame`

**Returns** pandas DataFrame with conditions/timepoints as rows and model expressions as columns.

## 10.5.15 amici.logging

### Logging

This module provides custom logging functionality for other amici modules

## Functions

---

<code>get_logger([logger_name, log_level])</code>	Returns (if extistant) or creates an AMICI logger
<code>log_execution_time(description, logger)</code>	Parameterized function decorator that enables automatic execution time tracking
<code>set_log_level(logger, log_level)</code>	

---

**rtype** `None`

---

`amici.logging.get_logger(logger_name='amici', log_level=None, **kwargs)`

Returns (if extistant) or creates an AMICI logger

If the AMICI base logger has already been set up, this method will return it or any of its descendant loggers without overriding the settings - i.e. any values supplied as kwargs will be ignored.

### Parameters

- **logger\_name** (`typing.Optional[str]`) – Get a logger for a specific namespace, typically `__name__` for code outside of classes or `self.__module__` inside a class
- **log\_level** (`typing.Optional[int]`) – Override the default or preset log level for the requested logger. `None` or `False` uses the default or preset value. `True` evaluates to `logging.DEBUG`. Any integer is used directly.
- **console\_output** – Set up a default console log handler if `True` (default). Only used when the AMICI logger hasn't been set up yet.
- **file\_output** – Supply a filename to copy all log output to that file, or set to `False` to disable (default). Only used when the AMICI logger hasn't been set up yet.
- **capture\_warnings** – Capture warnings from Python's warnings module if `True` (default). Only used when the AMICI logger hasn't been set up yet..

**Return type** `logging.Logger`

**Returns** A `logging.Logger` object with the requested name

`amici.logging.log_execution_time(description, logger)`

Parameterized function decorator that enables automatic execution time tracking

### Parameters

- **description** (`str`) – Description of what the decorated function does
- **logger** (`logging.Logger`) – Logger to which execution timing will be printed

**Return type** `typing.Callable`

`amici.logging.set_log_level(logger, log_level)`

**Return type** `None`

## 10.5.16 amici.gradient\_check

### Finite Difference Check

This module provides functions to automatically check correctness of amici computed sensitivities using finite difference approximations

### Functions

<code>check_derivatives(model, solver[, edata, ...])</code>	Finite differences check for likelihood gradient.
<code>check_finite_difference(x0, model, solver, ...)</code>	Checks the computed sensitivity based derivatives against a finite difference approximation.

`amici.gradient_check.check_derivatives(model, solver, edata=None, atol=0.0001, rtol=0.0001, epsilon=0.001, check_least_squares=True, skip_zero_pars=False)`

Finite differences check for likelihood gradient.

#### Parameters

- **model** (`amici.amici.Model`) – amici model
- **solver** (`amici.amici.Solver`) – amici solver
- **edata** (`typing.Optional[amici.swig_wrappers.ExpData]`) – exp data
- **atol** (`typing.Optional[float]`) – absolute tolerance for comparison
- **rtol** (`typing.Optional[float]`) – relative tolerance for comparison
- **epsilon** (`typing.Optional[float]`) – finite difference step-size
- **check\_least\_squares** (`bool`) – whether to check least squares related values.
- **skip\_zero\_pars** (`bool`) – whether to perform FD checks for parameters that are zero

#### Return type `None`

`amici.gradient_check.check_finite_difference(x0, model, solver, edata, ip, fields, atol=0.0001, rtol=0.0001, epsilon=0.001)`

Checks the computed sensitivity based derivatives against a finite difference approximation.

#### Parameters

- **x0** (`typing.Sequence[float]`) – parameter value at which to check finite difference approximation
- **model** (`amici.amici.Model`) – amici model
- **solver** (`amici.amici.Solver`) – amici solver
- **edata** (`amici.swig_wrappers.ExpData`) – exp data
- **ip** (`int`) – parameter index
- **fields** (`typing.List[str]`) – rdata fields for which to check the gradient
- **atol** (`typing.Optional[float]`) – absolute tolerance for comparison
- **rtol** (`typing.Optional[float]`) – relative tolerance for comparison
- **epsilon** (`typing.Optional[float]`) – finite difference step-size

Return type `None`

## 10.5.17 amici.parameter\_mapping

### Parameter mapping

When performing parameter inference, often parameters need to be mapped from simulation to estimation parameters, and parameters can differ between conditions. This can be handled using the *ParameterMapping*.

**Note:** While the parameter mapping can be used directly with AMICI, it was developed for usage together with PEtab, for which the whole workflow of generating the mapping is automatized.

### Functions

<code>amici_to_petab_scale(amici_scale)</code>	Convert amici scale id to petab scale id.
<code>fill_in_parameters(edatas, ...)</code>	Fill fixed and dynamic parameters into the edatas (in-place).
<code>fill_in_parameters_for_condition(edata, ...)</code>	Fill fixed and dynamic parameters into the edata for condition (in-place).
<code>petab_to_amici_scale(petab_scale)</code>	Convert petab scale id to amici scale id.
<code>scale_parameter(value, petab_scale)</code>	Bring parameter from linear scale to target scale.
<code>scale_parameters_dict(value_dict, ...)</code>	Bring parameters from linear scale to target scale.
<code>unscale_parameter(value, petab_scale)</code>	Bring parameter from scale to linear scale.
<code>unscale_parameters_dict(value_dict, ...)</code>	Bring parameters from target scale to linear scale.

### Classes

<code>ParameterMapping([parameter_mappings])</code>	Parameter mapping for multiple conditions.
<code>ParameterMappingForCondition([map_sim_var, ...])</code>	Parameter mapping for condition.

**class** `amici.parameter_mapping.ParameterMapping(parameter_mappings=None)`

Parameter mapping for multiple conditions.

This can be used like a list of *ParameterMappingForConditions*.

**Parameters** `parameter_mappings` (`typing.Optional[typing.List[amici.parameter_mapping.ParameterMappingForCondition]]`) – List of parameter mappings for specific conditions.

**\_\_init\_\_**(`parameter_mappings=None`)

**append**(`parameter_mapping_for_condition`)

Append a condition specific parameter mapping.

**count**(`value`) → integer -- return number of occurrences of value

**property free\_symbols:** `Set[str]`

Get IDs of all (symbolic) parameters present in this mapping

**index**(*value*[, *start*[, *stop* ]]) → integer -- return first index of value.

Raises `ValueError` if the value is not present.

Supporting start and stop arguments is optional, but recommended.

```
class amici.parameter_mapping.ParameterMappingForCondition(map_sim_var=None,
                                                         scale_map_sim_var=None,
                                                         map_preeq_fix=None,
                                                         scale_map_preeq_fix=None,
                                                         map_sim_fix=None,
                                                         scale_map_sim_fix=None)
```

Parameter mapping for condition.

Contains mappings for free parameters, fixed parameters, and fixed preequilibration parameters, both for parameters and scales.

In the scale mappings, for each simulation parameter the scale on which the value is passed (and potentially gradients are to be returned) is given. In the parameter mappings, for each simulation parameter a corresponding optimization parameter (or a numeric value) is given.

If a mapping is not passed, the parameter mappings are assumed to be empty, and if a scale mapping is not passed, all scales are set to linear.

#### Parameters

- **map\_sim\_var** (`typing.Optional[typing.Dict[str, typing.Union[str, numbers.Number]]]`) – Mapping for free simulation parameters.
- **scale\_map\_sim\_var** (`typing.Optional[typing.Dict[str, str]]`) – Scales for free simulation parameters.
- **map\_preeq\_fix** (`typing.Optional[typing.Dict[str, typing.Union[str, numbers.Number]]]`) – Mapping for fixed preequilibration parameters.
- **scale\_map\_preeq\_fix** (`typing.Optional[typing.Dict[str, str]]`) – Scales for fixed preequilibration parameters.
- **map\_sim\_fix** (`typing.Optional[typing.Dict[str, typing.Union[str, numbers.Number]]]`) – Mapping for fixed simulation parameters.
- **scale\_map\_sim\_fix** (`typing.Optional[typing.Dict[str, str]]`) – Scales for fixed simulation parameters.

```
__init__(map_sim_var=None, scale_map_sim_var=None, map_preeq_fix=None,
        scale_map_preeq_fix=None, map_sim_fix=None, scale_map_sim_fix=None)
```

**property free\_symbols:** `Set[str]`

Get IDs of all (symbolic) parameters present in this mapping

```
amici.parameter_mapping.amici_to_petab_scale(amici_scale)
```

Convert amici scale id to petab scale id.

**Return type** `str`

```
amici.parameter_mapping.fill_in_parameters(edatas, problem_parameters, scaled_parameters,
                                          parameter_mapping, amici_model)
```

Fill fixed and dynamic parameters into the edatas (in-place).

#### Parameters

- **edatas** (`typing.List[amici.swig_wrappers.ExpData]`) – List of experimental datas `amici.amici.ExpData` with everything except parameters filled.

- **problem\_parameters** (`typing.Dict[str, numbers.Number]`) – Problem parameters as `parameterId=>value` dict. Only parameters included here will be set. Remaining parameters will be used as currently set in *amici\_model*.
- **scaled\_parameters** (`bool`) – If True, problem\_parameters are assumed to be on the scale provided in the parameter mapping. If False, they are assumed to be in linear scale.
- **parameter\_mapping** (`amici.parameter_mapping.ParameterMapping`) – Parameter mapping for all conditions.
- **amici\_model** (`typing.Union[amici.amici.Model, amici.amici.ModelPtr]`) – AMICI model.

**Return type** `None`

`amici.parameter_mapping.fill_in_parameters_for_condition(edata, problem_parameters, scaled_parameters, parameter_mapping, amici_model)`

Fill fixed and dynamic parameters into the edata for condition (in-place).

#### Parameters

- **edata** (`amici.swig_wrappers.ExpData`) – Experimental data object to fill parameters into.
- **problem\_parameters** (`typing.Dict[str, numbers.Number]`) – Problem parameters as `parameterId=>value` dict. Only parameters included here will be set. Remaining parameters will be used as already set in *amici\_model* and *edata*.
- **scaled\_parameters** (`bool`) – If True, problem\_parameters are assumed to be on the scale provided in the parameter mapping. If False, they are assumed to be in linear scale.
- **parameter\_mapping** (`amici.parameter_mapping.ParameterMappingForCondition`) – Parameter mapping for current condition.
- **amici\_model** (`typing.Union[amici.amici.Model, amici.amici.ModelPtr]`) – AMICI model

**Return type** `None`

`amici.parameter_mapping.petab_to_amici_scale(petab_scale)`

Convert petab scale id to amici scale id.

**Return type** `int`

`amici.parameter_mapping.scale_parameter(value, petab_scale)`

Bring parameter from linear scale to target scale.

#### Parameters

- **value** (`numbers.Number`) – Value to scale
- **petab\_scale** (`str`) – Target scale of value

**Return type** `numbers.Number`

**Returns** value on target scale

`amici.parameter_mapping.scale_parameters_dict(value_dict, petab_scale_dict)`

Bring parameters from linear scale to target scale.

Bring values in *value\_dict* from linear scale to the scale provided in *petab\_scale\_dict* (in-place). Both arguments are expected to have the same length and matching keys.

**Parameters**

- **value\_dict** (`typing.Dict[typing.Any, numbers.Number]`) – Values to scale
- **petab\_scale\_dict** (`typing.Dict[typing.Any, str]`) – Target scales of values

**Return type** `None``amici.parameter_mapping.unscale_parameter(value, petab_scale)`

Bring parameter from scale to linear scale.

**Parameters**

- **value** (`numbers.Number`) – Value to scale
- **petab\_scale** (`str`) – Target scale of value

**Return type** `numbers.Number`**Returns** value on linear scale`amici.parameter_mapping.unscale_parameters_dict(value_dict, petab_scale_dict)`

Bring parameters from target scale to linear scale.

Bring values in `value_dict` from linear scale to the scale provided in `petab_scale_dict` (in-place). Both arguments are expected to have the same length and matching keys.**Parameters**

- **value\_dict** (`typing.Dict[typing.Any, numbers.Number]`) – Values to scale
- **petab\_scale\_dict** (`typing.Dict[typing.Any, str]`) – Target scales of values

**Return type** `None`

## 10.5.18 amici.conservated\_quantities\_demartino

### Functions

---

<code>compute_moiety_conservation_laws(...[, ...])</code>	Compute moiety conservation laws.
---	-----------------------------------

---

```
amici.conservated_quantities_demartino.compute_moiety_conservation_laws(stoichiometric_list,
                                                                    num_species,
                                                                    num_reactions,
                                                                    max_num_monte_carlo=20,
                                                                    rng_seed=False,
                                                                    species_names=None)
```

Compute moiety conservation laws.

According to the algorithm proposed by De Martino et al. (2014) <https://doi.org/10.1371/journal.pone.0100750>**Parameters**

- **stoichiometric\_list** (`typing.Sequence[float]`) – the stoichiometric matrix as a list (species x reactions, column-major ordering)
- **num\_species** (`int`) – total number of species in the reaction network
- **num\_reactions** (`int`) – total number of reactions in the reaction network
- **max\_num\_monte\_carlo** (`int`) – maximum number of MonteCarlo steps before changing to relaxation

- **rng\_seed** (`typing.Union[None, bool, int]`) – Seed for the random number generator. If *False*, the RNG will not be re-initialized. Other values will be passed to `random.seed()`.
- **species\_names** (`typing.Optional[typing.Sequence[str]]`) – Species names. Optional and only used for logging.

**Return type** `typing.Tuple[typing.List[typing.List[int]], typing.List[typing.List[float]]]`

**Returns** Integer MCLs as list of lists of indices of involved species and list of lists of corresponding coefficients.

### 10.5.19 amici.conserverved\_quantities\_rref

Find conserved quantities deterministically

#### Functions

<code>nullspace_by_rref(mat)</code>	Compute basis of the nullspace of <code>mat</code> based on the reduced row echelon form
<code>pivots(mat)</code>	Get indices of pivot columns in <code>mat</code> , assumed to be in reduced row echelon form
<code>rref(mat[, round_ndigits])</code>	Bring matrix <code>mat</code> to reduced row echelon form

`amici.conserverved_quantities_rref.nullspace_by_rref(mat)`

Compute basis of the nullspace of `mat` based on the reduced row echelon form

**Return type** `numpy.array`

`amici.conserverved_quantities_rref.pivots(mat)`

Get indices of pivot columns in `mat`, assumed to be in reduced row echelon form

**Return type** `typing.List[int]`

`amici.conserverved_quantities_rref.rref(mat, round_ndigits=None)`

Bring matrix `mat` to reduced row echelon form

see [https://en.wikipedia.org/wiki/Row\\_echelon\\_form](https://en.wikipedia.org/wiki/Row_echelon_form)

#### Parameters

- **mat** (`numpy.array`) – Numpy float matrix to operate on (will be copied)
- **round\_ndigits** (`typing.Union[typing.Literal[False], int, None]`) – Number of digits to round intermediary results to, or *False* to disable rounding completely. Helps to avoid numerical artifacts.

**Return type** `numpy.array`

**Returns** `mat` in rref form.



## 10.5.20 amici.numpy

### C++ object views

This module provides views on C++ objects for efficient access.

### Functions

<code>evaluate(expr, rdata)</code>	Evaluate a symbolic expression based on the given simulation outputs.
------------------------------------	---

### Classes

<code>ExpDataView(edata)</code>	Interface class for C++ Exp Data objects that avoids possibly costly copies of member data.
<code>ReturnDataView(rdata)</code>	Interface class for C++ amici.ReturnData objects that avoids possibly costly copies of member data.
<code>SwigPtrView(swigptr)</code>	Interface class to expose <code>std::vector&lt;double&gt;</code> and scalar members of swig wrapped C++ objects as numpy array attributes and fields.

**class** `amici.numpy.ExpDataView(edata)`

Interface class for C++ Exp Data objects that avoids possibly costly copies of member data.

NOTE: This currently assumes that the underlying ExpData does not change after instantiating an `ExpDataView`.

`__init__(edata)`

Constructor

**Parameters** `edata` (`typing.Union[amici.amici.ExpDataPtr, amici.amici.ExpData]`)  
– pointer to the ExpData instance

`get(k[, d])` → `D[k]` if `k` in `D`, else `d`. `d` defaults to `None`.

`items()` → a set-like object providing a view on `D`'s items

`keys()` → a set-like object providing a view on `D`'s keys

`values()` → an object providing a view on `D`'s values

**class** `amici.numpy.ReturnDataView(rdata)`

Interface class for C++ amici.ReturnData objects that avoids possibly costly copies of member data.

`__init__(rdata)`

Constructor

**Parameters** `rdata` (`typing.Union[amici.amici.ReturnDataPtr, amici.amici.ReturnData]`) – pointer to the ReturnData instance

`by_id(entity_id, field=None, model=None)`

Get the value of a given field for a named entity.

**Parameters**

- **entity\_id** (`str`) – The ID of the model entity that is to be extracted from `field` (e.g. a state ID).
- **field** (`typing.Optional[str]`) – The requested field, e.g. 'x' for model states. This is optional if field would be one of {'x', 'y', 'w'}
- **model** (`typing.Optional[amici.amici.Model]`) – The model from which this ReturnDataView was generated. This is optional if this ReturnData was generated with `solver.getReturnDataReportingMode() == amici.RDataReporting.full`.

**Return type** `numpy.array`

**get**(`k[, d]`) → `D[k]` if `k` in `D`, else `d`. `d` defaults to `None`.

**items**() → a set-like object providing a view on `D`'s items

**keys**() → a set-like object providing a view on `D`'s keys

**values**() → an object providing a view on `D`'s values

**class** `amici.numpy.SwigPtrView`(`swigptr`)

Interface class to expose `std::vector<double>` and scalar members of swig wrapped C++ objects as numpy array attributes and fields. This class is memory efficient as copies of the underlying C++ objects is only created when respective fields are accessed for the first time. Cached copies are used for all subsequent calls.

#### Variables

- **\_swigptr** – pointer to the C++ object
- **\_field\_names** – names of members that will be exposed as numpy arrays
- **\_field\_dimensions** – dimensions of numpy arrays
- **\_cache** – dictionary with cached values

**\_\_init\_\_**(`swigptr`)

Constructor

**Parameters** `swigptr` – pointer to the C++ object

**get**(`k[, d]`) → `D[k]` if `k` in `D`, else `d`. `d` defaults to `None`.

**items**() → a set-like object providing a view on `D`'s items

**keys**() → a set-like object providing a view on `D`'s keys

**values**() → an object providing a view on `D`'s values

`amici.numpy.evaluate`(`expr, rdata`)

Evaluate a symbolic expression based on the given simulation outputs.

#### Parameters

- **expr** (`typing.Union[str, sympy.core.expr.Expr]`) – A symbolic expression, e.g. a sympy expression or a string that can be sympified. Can include state variable, expression, and observable IDs, depending on whether the respective data is available in the simulation results. Parameters are not yet supported.
- **rdata** (`amici.numpy.ReturnDataView`) – The simulation results.

**Return type** `numpy.array`

**Returns** The evaluated expression for the simulation output timepoints.

## 10.5.21 amici.sbml\_utils

### SBML Utilities

This module provides helper functions for working with SBML.

### Functions

<i>add_assignment_rule</i> (model, variable_id, formula)	Helper for adding an assignment rule to a SBML model.
<i>add_compartment</i> (model, compartment_id, *[, size])	Helper for adding a compartment to a SBML model.
<i>add_inflow</i> (model, species_id, rate, *[, ...])	
<b>rtype</b> libsbml.Reaction	
<i>add_parameter</i> (model, parameter_id, *[, ...])	Helper for adding a parameter to a SBML model.
<i>add_rate_rule</i> (model, variable_id, formula[, ...])	Helper for adding a rate rule to a SBML model.
<i>add_species</i> (model, species_id, *[, ...])	Helper for adding a species to a SBML model.
<i>create_sbml_model</i> (model_id[, level, version])	Helper for creating an empty SBML model.
<i>get_sbml_units</i> (model, x)	Try to get the units for expression x.
<i>mathml2sympy</i> (mathml, *[, evaluate, locals, ...])	
<b>rtype</b> sympy.core.basic.Basic	
<i>pretty_xml</i> (ugly_xml)	Prettifies an XML document (given as a string).
<i>sbml_math_ast</i> (expr, **kwargs)	Convert a SymPy expression to SBML math AST.
<i>sbml_mathml</i> (expr, *[, replace_time, pretty])	Prints a SymPy expression to a MathML expression parsable by libSBML.
<i>set_sbml_math</i> (obj, expr, **kwargs)	Set the math attribute of a SBML node using a SymPy expression.

### Classes

<i>MathMLSbmlPrinter</i> ([settings])	Prints a SymPy expression to a MathML expression parsable by libSBML.
---------------------------------------	---

### Exceptions

<i>SbmlAnnotationError</i>	
<i>SbmlDuplicateComponentIdError</i>	
<i>SbmlInvalidIdSyntax</i>	
<i>SbmlMathError</i>	
<i>SbmlMissingComponentIdError</i>	

```
class amici.sbml_utils.MathMLSbmlPrinter(settings=None)
    Prints a SymPy expression to a MathML expression parsable by libSBML.

    Differences from sympy.MathMLContentPrinter: 1. underscores in symbol names are not converted to subscripts
    2. symbols with name 'time' are converted to the SBML time symbol

    __init__(settings=None)

    apply_patch()

    doprint(expr, *, pretty=False)
        Prints the expression as MathML.

        Return type str

    emptyPrinter(expr)

    mathml_tag(e)
        Returns the MathML tag for an expression.

    property order

    printmethod: str = '_mathml_content'

    restore_patch()

    classmethod set_global_settings(**settings)
        Set system-wide printing settings.

exception amici.sbml_utils.SbmlAnnotationError

    __init__(*args, **kwargs)

    args

    with_traceback()
        Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception amici.sbml_utils.SbmlDuplicateComponentIdError

    __init__(*args, **kwargs)

    args

    with_traceback()
        Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception amici.sbml_utils.SbmlInvalidIdSyntax

    __init__(*args, **kwargs)

    args

    with_traceback()
        Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception amici.sbml_utils.SbmlMathError

    __init__(*args, **kwargs)
```

**args**

**with\_traceback()**

Exception.with\_traceback(tb) – set self.\_\_traceback\_\_ to tb and return self.

**exception amici.sbml\_utils.SbmlMissingComponentIdError**

**\_\_init\_\_**(\*args, \*\*kwargs)

**args**

**with\_traceback()**

Exception.with\_traceback(tb) – set self.\_\_traceback\_\_ to tb and return self.

**amici.sbml\_utils.add\_assignment\_rule**(model, variable\_id, formula, rule\_id=None)

Helper for adding an assignment rule to a SBML model.

**Parameters**

- **model** (`libsbml.Model`) – SBML model to which the assignment rule is to be added.
- **variable\_id** (`typing.Union[str, sympy.core.symbol.Symbol]`) – SBML ID of the quantity for which the assignment rule is to be added.
- **formula** – Formula for the assignment rule (it will be sympified).
- **rule\_id** (`typing.Optional[str]`) – SBML ID of the new assignment rule. Defaults to `'assignment_' + variableId`.

**Return type** `libsbml.AssignmentRule`

**Returns** The assignment rule as a `libsbml.AssignmentRule` object.

**amici.sbml\_utils.add\_compartment**(model, compartment\_id, \*, size=1.0)

Helper for adding a compartment to a SBML model.

**Parameters**

- **model** (`libsbml.Model`) – SBML model to which the compartment is to be added.
- **compartment\_id** (`typing.Union[str, sympy.core.symbol.Symbol]`) – SBML ID of the new compartment.
- **size** (`float`) – Size of the new compartment. Defaults to `1.0`.

**Return type** `libsbml.Species`

**Returns** The new compartment as a `libsbml.Compartment` object.

**amici.sbml\_utils.add\_inflow**(model, species\_id, rate, \*, reaction\_id=None, reversible=False)

**Return type** `libsbml.Reaction`

**amici.sbml\_utils.add\_parameter**(model, parameter\_id, \*, name=False, value=None, units=None, constant=None)

Helper for adding a parameter to a SBML model.

**Parameters**

- **model** (`libsbml.Model`) – SBML model to which the parameter is to be added.
- **parameter\_id** (`typing.Union[str, sympy.core.symbol.Symbol]`) – SBML ID of the new parameter.
- **name** (`typing.Union[bool, str]`) – SBML name of the new parameter.

- **value** (`typing.Optional[float]`) – Value attribute for the new parameter.
- **units** (`typing.Optional[str]`) – Units attribute for the new parameter.
- **constant** (`typing.Optional[bool]`) – Constant attribute for the new parameter.

**Return type** `libsbml.Parameter`

**Returns** The new parameter as a `libsbml.Parameter` object.

`amici.sbml_utils.add_rate_rule(model, variable_id, formula, rule_id=None)`

Helper for adding a rate rule to a SBML model.

**Parameters**

- **model** (`libsbml.Model`) – SBML model to which the rate rule is to be added.
- **variable\_id** (`typing.Union[str, sympy.core.symbol.Symbol]`) – SBML ID of the quantity for which the rate rule is to be added.
- **formula** – Formula for the rate rule (it will be sympified).
- **rule\_id** (`typing.Optional[str]`) – SBML ID of the new rate rule. Defaults to `'rate_' + variableId`.

**Return type** `libsbml.RateRule`

**Returns** The new rate rule as a `libsbml.RateRule` object.

`amici.sbml_utils.add_species(model, species_id, *, compartment_id=None, name=False, initial_amount=0.0, units=None)`

Helper for adding a species to a SBML model.

**Parameters**

- **model** (`libsbml.Model`) – SBML model to which the species is to be added.
- **species\_id** (`typing.Union[str, sympy.core.symbol.Symbol]`) – SBML ID of the new species.
- **compartment\_id** (`typing.Optional[str]`) – Compartment ID for the new species. If there is only one compartment it can be auto-selected.
- **initial\_amount** (`float`) – Initial amount of the new species.
- **units** (`typing.Optional[str]`) – Units attribute for the new species.

**Return type** `libsbml.Species`

**Returns** The new species as a `libsbml.Species` object.

`amici.sbml_utils.create_sbml_model(model_id, level=2, version=5)`

Helper for creating an empty SBML model.

**Parameters**

- **model\_id** (`str`) – SBML ID of the new model.
- **level** (`int`) – Level of the new SBML document.
- **version** (`int`) – Version of the new SBML document.

**Return type** `typing.Tuple[libsbml.SBMLDocument, libsbml.Model]`

**Returns** A tuple containing the newly created `libsbml.SBMLDocument` and `libsbml.Model`.

`amici.sbml_utils.get_sbml_units(model, x)`

Try to get the units for expression  $x$ .

**Parameters**

- **model** (`libsbml.Model`) – SBML model.
- **x** (`typing.Union[str, sympy.core.symbol.Symbol, sympy.core.basic.Basic]`) – Expression to get the units of.

**Return type** `typing.Optional[str]`

**Returns** A string if the units could be determined, otherwise *None*.

`amici.sbml_utils.mathml2sympy(mathml, *, evaluate=False, locals=None, expression_type='mathml2sympy')`

**Return type** `sympy.core.basic.Basic`

`amici.sbml_utils.pretty_xml(ugly_xml)`

Prettifies an XML document (given as a string).

**Return type** `str`

`amici.sbml_utils.sbml_math_ast(expr, **kwargs)`

Convert a SymPy expression to SBML math AST.

**Parameters**

- **expr** – expression to be converted (will be sympified).
- **kwargs** – extra options for MathML conversion.

**Return type** `libsbml.ASTNode`

`amici.sbml_utils.sbml_mathml(expr, *, replace_time=False, pretty=False, **settings)`

Prints a SymPy expression to a MathML expression parsable by libSBML.

**Parameters**

- **expr** – expression to be converted to MathML (will be sympified).
- **replace\_time** (`bool`) – replace the AMICI time symbol with the SBML time symbol.
- **pretty** (`bool`) – prettify the resulting MathML.

**Return type** `str`

`amici.sbml_utils.set_sbml_math(obj, expr, **kwargs)`

Set the math attribute of a SBML node using a SymPy expression.

**Parameters**

- **obj** (`libsbml.SBase`) – SBML node supporting *setMath* method.
- **expr** – expression to which the math attribute of *obj* should be set to (will be sympified).
- **kwargs** – extra options for MathML conversion.

**Return type** `None`

## 10.5.22 amici.splines

### Splines

This module provides helper functions for reading/writing splines with AMICI annotations from/to SBML files and for adding such splines to the AMICI C++ code.

### Functions

<code>spline_user_functions(splines, p_index)</code>	Custom user functions to be used in <i>ODEExporter</i> for linking spline expressions to C++ code.
<code>sympify_noeval(x)</code>	

### Classes

<code>AbstractSpline(sbml_id, nodes, ...[, ...])</code>	Base class for spline functions which can be computed efficiently thanks to tailored C++ implementations in AMICI.
<code>CubicHermiteSpline(sbml_id, nodes, ...[, ...])</code>	
<code>UniformGrid(start, stop[, step, ...])</code>	A grid of uniformly-spaced real points, computed with rational arithmetic.

```
class amici.splines.AbstractSpline(sbml_id, nodes, values_at_nodes, *, evaluate_at=None, bc=None,
                                   extrapolate=None, logarithmic_parametrization=False)
```

Base class for spline functions which can be computed efficiently thanks to tailored C++ implementations in AMICI. Inside an SBML file, such splines are implemented with an assignment rule containing both a symbolic piecewise formula for the spline (allowing compatibility with any SBML-aware software) and annotations which encode the necessary information for AMICI to recreate the spline object (allowing for fast computations when the SBML file is used together with AMICI).

```
__init__(sbml_id, nodes, values_at_nodes, *, evaluate_at=None, bc=None, extrapolate=None,
          logarithmic_parametrization=False)
```

Base constructor for `AbstractSpline` objects.

#### Parameters

- **sbml\_id** (`typing.Union[str, sympy.core.symbol.Symbol]`) – The SBML ID of the parameter associated to the spline as a string or a SymPy symbol.
- **nodes** (`typing.Sequence`) – The points at which the spline values are known. Currently, they must be numeric or only depend on constant parameters. These points should be strictly increasing. This argument will be sympified.
- **values\_at\_nodes** (`typing.Sequence`) – The spline values at each of the points in nodes. They must not depend on model species. This argument will be sympified.
- **evaluate\_at** (`typing.Union[str, sympy.core.basic.Basic, None]`) – The point at which the spline is evaluated. It will be sympified. Defaults to model time.
- **bc** (`typing.Union[None, str, typing.Tuple[typing.Optional[str], typing.Optional[str]]]`) – Tuple of applied boundary conditions, one for each side of the spline



domain. If a single boundary condition is given it will be applied to both sides. Possible boundary conditions (allowed values depend on the `AbstractSpline` subclass):

***None* or *'no\_bc'***: Boundary conditions are not needed for this spline object;

***'zeroderivative'***: first derivative set to zero;

***'natural'***: second derivative set to zero;

***'zeroderivative+natural'***: first and second derivatives set to zero;

***'periodic'***: periodic bc.

- **`extrapolate`** (`typing.Union[None, str, typing.Tuple[typing.Optional[str], typing.Optional[str]]]`) – Whether to extrapolate the spline outside the base interval defined by (`nodes[0]`, `nodes[-1]`). It is a tuple of extrapolation methods, one for each side of the base interval. If it is not a tuple, then the same extrapolation will be applied on both sides. Extrapolation methods supported:

***None* or *'no\_extrapolation'***: no extrapolation should be performed. An exception will be raised in the C++ code if the spline is evaluated outside the base interval. In the fallback SBML symbolic expression *'polynomial'* extrapolation will be used.

***'polynomial'***: the cubic polynomial used in the nearest spline segment will be used.

***'constant'***: constant extrapolation will be used. Requires *'zeroderivative'* boundary condition. For splines which are continuous up to the second derivative, it requires the stricter *'zeroderivative+natural'* boundary condition.

***'linear'***: linear extrapolation will be used. For splines which are continuous up to the second derivative, this requires the *'natural'* boundary condition.

***'periodic'***: Periodic extrapolation. Requires *'periodic'* boundary conditions.

- **`logarithmic_parametrization`** (`bool`) – Whether interpolation should be done in log-scale.

**`add_to_sbml_model`** (`model`, `*`, `auto_add=False`, `x_nominal=None`, `y_nominal=None`, `x_units=None`, `y_units=None`, `y_constant=None`)

Function to add the spline to an SBML model using an assignment rule with AMICI-specific annotations.

#### Parameters

- **`model`** (`libsbml.Model`) – A `libsbml.Model` to which the spline is to be added.
- **`auto_add`** (`typing.Union[bool, str]`) – Automatically add missing parameters to the SBML model (defaults to *False*). Only used for expressions consisting in a single symbol. If equal to *'spline'*, only the parameter representing the spline will be added.
- **`x_nominal`** (`typing.Optional[typing.Sequence[float]]`) – Nominal values used when auto-adding parameters for *nodes*.
- **`y_nominal`** (`typing.Union[typing.Sequence[float], float, None]`) – Nominal values used when auto-adding parameters for *values\_at\_nodes*.
- **`x_units`** (`typing.Optional[str]`) – Units used when auto-adding parameters for *nodes*.
- **`y_units`** (`typing.Optional[str]`) – Units used when auto-adding parameters for *values\_at\_nodes*.
- **`y_constant`** (`typing.Union[typing.Sequence[bool], bool, None]`) – Constant flags used when auto-adding parameters for *values\_at\_nodes*.

**Return type** `None`

**property amici\_annotation:** `str`

An SBML annotation describing the spline.

**property bc:** `Tuple[Union[None, str], Union[None, str]]`

Boundary conditions applied to this spline.

**check\_if\_valid**(*importer*)

Check if the spline described by this object can be correctly be implemented by AMICI. E.g., check whether the formulas for spline grid points, values, ... contain species symbols.

**Return type** `None`

**derivative**(*x*, *\*\*kwargs*)

Evaluate the spline derivative at the point *x*.

**Return type** `sympy.core.expr.Expr`

**evaluate**(*x*)

Evaluate the spline at the point *x*.

**Return type** `sympy.core.basic.Basic`

**property evaluate\_at:** `sympy.core.basic.Basic`

The symbolic argument at which the spline is evaluated.

**property extrapolate:** `Tuple[Union[None, str], Union[None, str]]`

Whether to extrapolate the spline outside the base interval.

**property extrapolation\_formulas:** `Tuple[Union[None, sympy.core.basic.Basic], Union[None, sympy.core.basic.Basic]]`

Returns the extrapolation formulas on the left and right side of the interval (`nodes[0]`, `nodes[-1]`). A value of `None` means that no extrapolation is required.

**property formula:** `sympy.functions.elementary.piecewise.Piecewise`

Compute a symbolic piecewise formula for the spline.

**static from\_annotation**(*sbml\_id*, *annotation*, *\**, *locals\_*)

Create a spline object from a SBML annotation.

This function extracts annotation and children from the XML annotation and gives them to the `_fromAnnotation` function for parsing. Subclass behaviour should be implemented by extending `_fromAnnotation`. However, the mapping between method strings and subclasses must be hard-coded into this function here (at the moment).

**Return type** `amici.splines.AbstractSpline`

**static get\_annotation**(*rule*)

Extract AMICI spline annotation from an SBML assignment rule (given as a `libsbml.AssignmentRule` object). Return `None` if any such annotation could not be found.

**Return type** `typing.Optional[xml.etree.ElementTree.Element]`

**integrate**(*x0*, *x1*)

Integrate the spline between the points *x0* and *x1*.

**Return type** `sympy.core.basic.Basic`

**static is\_spline**(*rule*)

Determine if an SBML assignment rule (given as a `libsbml.AssignmentRule` object) is an AMICI-annotated spline formula.

**Return type** `bool`

**property** `logarithmic_parametrization`: `bool`

Whether interpolation is done in log-scale.

**property** `mathml_formula`: `sympy.functions.elementary.piecewise.Piecewise`

Compute a symbolic piecewise formula for the spline for use inside a SBML assignment rule: SBML symbol naming will be used and operations not supported by SBML MathML will be avoided.

**abstract property method**: `str`

Spline method.

**property** `nodes`: `numpy.ndarray`

The points at which the spline values are known.

**ode\_model\_symbol**(*importer*)

Returns the *sympy* object to be used by `amici.de_export.ODEModel`. This expression can be differentiated and easily mapped to the C++ code.

**Return type** `sympy.core.function.Function`

**parameters**(*importer*)

Returns the SBML parameters used by this spline

**Return type** `typing.Set[sympy.core.symbol.Symbol]`

**property** `period`: `Optional[sympy.core.basic.Basic]`

Period of a periodic spline. *None* if the spline is not periodic.

**plot**(*parameters=None*, \*, *xlim=None*, *npoints=100*, *xlabel=None*, *ylabel='spline value'*, *ax=None*)

Plots the spline, highlighting the nodes positions.

**poly**(*i*, \*, *x=None*)

Get the polynomial interpolant on the `(nodes[i], nodes[i+1])` interval. The polynomial is written in Horner form with respect to the scaled variable `poly_variable(x, i)`. If no variable *x* is provided, it will default to the one given at initialization time.

**Return type** `sympy.core.basic.Basic`

**poly\_variable**(*x*, *i*)

Given an evaluation point, return the value of the variable in which the polynomial on the *i*-th interval is expressed.

**Return type** `sympy.core.basic.Basic`

**property** `sbml_formula`: `sympy.functions.elementary.piecewise.Piecewise`

Compute a symbolic piecewise formula for the spline, using SBML symbol naming (the AMICI time symbol will be replaced with its SBML counterpart).

**property** `sbml_id`: `sympy.core.symbol.Symbol`

SBML ID of the spline parameter.

**second\_derivative**(*x*)

Evaluate the spline second derivative at the point *x*.

**Return type** `sympy.core.basic.Basic`

**segment\_formula**(*i*, \*, *x=None*)

Return the formula for the actual value of the spline expression on the `(nodes[i], nodes[i+1])` interval. Unless logarithmic parametrization is used, this is equal to the interpolating polynomial.

Return type `sympy.core.basic.Basic`

**abstract property smoothness:** `int`

Smoothness of this spline.

**squared\_L2\_norm\_of\_curvature()**

Return the squared L2 norm of the spline's curvature (commonly used as a regularizer). This is always computed in the spline native scale (i.e., in log-scale for positivity enforcing splines).

Return type `sympy.core.basic.Basic`

**property values\_at\_nodes:** `numpy.ndarray`

The spline values at each of the points in `nodes`.

**y\_scaled(*i*)**

Return the values which should be interpolated by a polynomial. Unless logarithmic parametrization is used, they are equal to the values given at initialization time.

```
class amici.splines.CubicHermiteSpline(sbml_id, nodes, values_at_nodes, derivatives_at_nodes=None, *,  
                                     evaluate_at=None, bc='auto', extrapolate=None,  
                                     logarithmic_parametrization=False)
```

```
__init__(sbml_id, nodes, values_at_nodes, derivatives_at_nodes=None, *, evaluate_at=None, bc='auto',  
         extrapolate=None, logarithmic_parametrization=False)
```

Constructor for *CubicHermiteSpline* objects.

#### Parameters

- **sbml\_id** (`typing.Union[str, sympy.core.symbol.Symbol]`) – The SBML ID of the parameter associated to the spline as a string or a SymPy symbol.
- **x** – The point at which the spline is evaluated. It will be sympified.
- **nodes** (`typing.Sequence`) – The points at which the spline values are known. Currently, they must be numeric or only depend on constant parameters. These points should be strictly increasing. This argument will be sympified.
- **values\_at\_nodes** (`typing.Sequence`) – The spline values at each of the points in *nodes*. They must not depend on model species. This argument will be sympified.
- **derivatives\_at\_nodes** (`typing.Optional[typing.Sequence]`) – The spline derivatives at each of the points in *nodes*. They must not depend on model species. This argument will be sympified. If not specified, it will be computed by finite differences.
- **evaluate\_at** (`typing.Union[str, sympy.core.basic.Basic, None]`) – The point at which the spline is evaluated. It will be sympified. Defaults to model time.
- **bc** (`typing.Union[None, str, typing.Tuple[typing.Optional[str], typing.Optional[str]]]`) – Applied boundary conditions (see *AbstractSpline* documentation). If 'auto' (the default), the boundary conditions will be automatically set depending on the extrapolation methods.
- **extrapolate** (`typing.Union[None, str, typing.Tuple[typing.Optional[str], typing.Optional[str]]]`) – Extrapolation method (see *AbstractSpline* documentation).
- **logarithmic\_parametrization** (`bool`) – Whether interpolation should be done in log-scale.

```
add_to_sbml_model(model, *, auto_add=False, x_nominal=None, y_nominal=None, x_units=None,  
                 y_units=None, y_constant=None)
```

Function to add the spline to an SBML model using an assignment rule with AMICI-specific annotations.

**Parameters**

- **model** (`libsbml.Model`) – A `libsbml.Model` to which the spline is to be added.
- **auto\_add** (`typing.Union[bool, str]`) – Automatically add missing parameters to the SBML model (defaults to `False`). Only used for expressions consisting in a single symbol. If equal to `'spline'`, only the parameter representing the spline will be added.
- **x\_nominal** (`typing.Optional[typing.Sequence[float]]`) – Nominal values used when auto-adding parameters for *nodes*.
- **y\_nominal** (`typing.Union[typing.Sequence[float], float, None]`) – Nominal values used when auto-adding parameters for *values\_at\_nodes*.
- **x\_units** (`typing.Optional[str]`) – Units used when auto-adding parameters for *nodes*.
- **y\_units** (`typing.Optional[str]`) – Units used when auto-adding parameters for *values\_at\_nodes*.
- **y\_constant** (`typing.Union[typing.Sequence[bool], bool, None]`) – Constant flags used when auto-adding parameters for *values\_at\_nodes*.

**Return type** `None`**property amici\_annotation:** `str`

An SBML annotation describing the spline.

**property bc:** `Tuple[Union[None, str], Union[None, str]]`

Boundary conditions applied to this spline.

**check\_if\_valid**(*importer*)

Check if the spline described by this object can be correctly be implemented by AMICI. E.g., check whether the formulas for spline grid points, values, ... contain species symbols.

**Return type** `None`**d\_scaled**(*i*)Return the derivative of the polynomial interpolant at the *i*-th point. Unless logarithmic parametrization is used, it is equal to the derivative of the spline expression.**Return type** `sympy.core.expr.Expr`**derivative**(*x*, *\*\*kwargs*)Evaluate the spline derivative at the point *x*.**Return type** `sympy.core.expr.Expr`**property derivatives\_at\_nodes:** `numpy.ndarray`The spline derivatives at each of the points in *nodes*.**property derivatives\_by\_fd:** `bool`**evaluate**(*x*)Evaluate the spline at the point *x*.**Return type** `sympy.core.basic.Basic`**property evaluate\_at:** `sympy.core.basic.Basic`

The symbolic argument at which the spline is evaluated.

**property extrapolate:** `Tuple[Union[None, str], Union[None, str]]`

Whether to extrapolate the spline outside the base interval.

**property** `extrapolation_formulas`: `Tuple[Union[None, sympy.core.basic.Basic], Union[None, sympy.core.basic.Basic]]`

Returns the extrapolation formulas on the left and right side of the interval (`nodes[0]`, `nodes[-1]`). A value of `None` means that no extrapolation is required.

**property** `formula`: `sympy.functions.elementary.piecewise.Piecewise`

Compute a symbolic piecewise formula for the spline.

**static** `from_annotation(sbml_id, annotation, *, locals_)`

Create a spline object from a SBML annotation.

This function extracts annotation and children from the XML annotation and gives them to the `_fromAnnotation` function for parsing. Subclass behaviour should be implemented by extending `_fromAnnotation`. However, the mapping between method strings and subclasses must be hard-coded into this function here (at the moment).

**Return type** `amici.splines.AbstractSpline`

**static** `get_annotation(rule)`

Extract AMICI spline annotation from an SBML assignment rule (given as a `libsbml.AssignmentRule` object). Return `None` if any such annotation could not be found.

**Return type** `typing.Optional[xml.etree.ElementTree.Element]`

**integrate**(`x0`, `x1`)

Integrate the spline between the points `x0` and `x1`.

**Return type** `sympy.core.basic.Basic`

**static** `is_spline(rule)`

Determine if an SBML assignment rule (given as a `libsbml.AssignmentRule` object) is an AMICI-annotated spline formula.

**Return type** `bool`

**property** `logarithmic_parametrization`: `bool`

Whether interpolation is done in log-scale.

**property** `mathml_formula`: `sympy.functions.elementary.piecewise.Piecewise`

Compute a symbolic piecewise formula for the spline for use inside a SBML assignment rule: SBML symbol naming will be used and operations not supported by SBML MathML will be avoided.

**property** `method`: `str`

Spline method (cubic Hermite spline)

**property** `nodes`: `numpy.ndarray`

The points at which the spline values are known.

**ode\_model\_symbol**(`importer`)

Returns the `sympy` object to be used by `amici.de_export.ODEModel`. This expression can be differentiated and easily mapped to the C++ code.

**Return type** `sympy.core.function.Function`

**parameters**(`importer`)

Returns the SBML parameters used by this spline

**Return type** `typing.Set[sympy.core.symbol.Symbol]`

**property period:** `Optional[sympy.core.basic.Basic]`

Period of a periodic spline. *None* if the spline is not periodic.

**plot**(*parameters=None, \*, xlim=None, npoints=100, xlabel=None, ylabel='spline value', ax=None*)

Plots the spline, highlighting the nodes positions.

**poly**(*i, \*, x=None*)

Get the polynomial interpolant on the `(nodes[i], nodes[i+1])` interval. The polynomial is written in Horner form with respect to the scaled variable `poly_variable(x, i)`. If no variable `x` is provided, it will default to the one given at initialization time.

**Return type** `sympy.core.basic.Basic`

**poly\_variable**(*x, i*)

Given an evaluation point, return the value of the variable in which the polynomial on the `i`-th interval is expressed.

**Return type** `sympy.core.basic.Basic`

**property sbml\_formula:** `sympy.functions.elementary.piecewise.Piecewise`

Compute a symbolic piecewise formula for the spline, using SBML symbol naming (the AMICI time symbol will be replaced with its SBML counterpart).

**property sbml\_id:** `sympy.core.symbol.Symbol`

SBML ID of the spline parameter.

**second\_derivative**(*x*)

Evaluate the spline second derivative at the point `x`.

**Return type** `sympy.core.basic.Basic`

**segment\_formula**(*i, \*, x=None*)

Return the formula for the actual value of the spline expression on the `(nodes[i], nodes[i+1])` interval. Unless logarithmic parametrization is used, this is equal to the interpolating polynomial.

**Return type** `sympy.core.basic.Basic`

**property smoothness:** `int`

Smoothness of this spline (equal to 1 for cubic Hermite splines since they are continuous up to the first derivative).

**squared\_L2\_norm\_of\_curvature**()

Return the squared L2 norm of the spline's curvature (commonly used as a regularizer). This is always computed in the spline native scale (i.e., in log-scale for positivity enforcing splines).

**Return type** `sympy.core.basic.Basic`

**property values\_at\_nodes:** `numpy.ndarray`

The spline values at each of the points in `nodes`.

**y\_scaled**(*i*)

Return the values which should be interpolated by a polynomial. Unless logarithmic parametrization is used, they are equal to the values given at initialization time.

**class** `amici.splines.UniformGrid`(*start, stop, step=None, \*, number\_of\_nodes=None, always\_include\_stop=True*)

A grid of uniformly-spaced real points, computed with rational arithmetic.

Implements the `collections.abc.Sequence` interface and can be converted to a `numpy.ndarray` (conversion to float can be specified with `dtype=float`).

### Variables

- **start** – first point.
- **stop** – last point.
- **step** – distance between consecutive points.
- **number\_of\_nodes** – number of grid nodes.

`__init__(start, stop, step=None, *, number_of_nodes=None, always_include_stop=True)`

Create a new `UniformGrid`.

Note: A `UniformGrid` with a single node cannot be created.

### Parameters

- **start** (`typing.Union[numbers.Real, sympy.core.basic.Basic]`) – First point in the grid
- **stop** (`typing.Union[numbers.Real, sympy.core.basic.Basic]`) – Last point in the grid (some caveats apply, see `always_include_stop`)
- **step** (`typing.Union[numbers.Real, sympy.core.basic.Basic, None]`) – Desired step size of the grid. Mutually exclusive with `number_of_nodes`.
- **number\_of\_nodes** (`typing.Optional[numbers.Integral]`) – Number of grid nodes, i.e., the length of the grid. It must be greater than or equal to 2. Mutually exclusive with `step`.
- **always\_include\_stop** (`bool`) – Controls the behaviour when `step` is not `None`. If `True` (default), the endpoint is the smallest `start + k * step`, with `k` integer, which is greater than or equal to `stop`. Otherwise, the endpoint is the largest `start + k * step`, with `k` integer, which is smaller than or equal to `stop`.

`count(value)` → integer -- return number of occurrences of value

`index(value[, start[, stop]])` → integer -- return first index of value.

Raises `ValueError` if the value is not present.

Supporting start and stop arguments is optional, but recommended.

**property number\_of\_nodes:** `sympy.core.basic.Basic`

Number of grid nodes.

**property start:** `sympy.core.basic.Basic`

First point.

**property step:** `sympy.core.basic.Basic`

Distance between consecutive points.

**property stop:** `sympy.core.basic.Basic`

Last point.

`amici.splines.spline_user_functions(splines, p_index)`

Custom user functions to be used in `ODEExporter` for linking spline expressions to C++ code.

**Return type** `typing.Dict[str, typing.List[typing.Tuple[typing.Callable[... , bool], typing.Callable[... , str]]]]`

`amici.splines.sympify_noeval(x)`



## C++ INTERFACE

### 11.1 Building the C++ library

The following section describes building the AMICI C++ library:

---

**Note:** The AMICI C++ interface only supports simulation of models imported using the *Python interface* and *Matlab interface*. It cannot be used for model import itself.

---

Prerequisites:

- CBLAS compatible BLAS library
- a C++17 compatible compiler
- a C compiler
- Optional: \* HDF5 libraries \* boost for serialization

To use AMICI from C++, run the

```
./scripts/buildSuiteSparse.sh  
./scripts/buildSundials.sh  
./scripts/buildAmici.sh
```

script to build the AMICI library.

---

**Note:** On some systems, the CMake executable may be named something other than `cmake`. In this case, set the `CMAKE` environment variable to the correct name (e.g. `export CMAKE=cmake3`, in case you have CMake available as `cmake3`).

---

The static library can then be linked from

```
./build/libamici.a
```

In CMake-based packages, amici can be linked via

```
find_package(Amici)
```

For further usage, consult the AMICI *C++ interface documentation*.

### 11.1.1 Supported CBLAS libraries

The C++ interfaces require a system installation of a CBLAS-compatible *Basic Linear Algebra Subprograms* (BLAS) library. AMICI has been tested with various implementations such as Accelerate, Intel MKL, cblas, openblas and atlas.

### 11.1.2 Optional SuperLU\_MT support

To build AMICI with SuperLU\_MT support, run

```
./scripts/buildSuperLUMT.sh
./scripts/buildSundials.sh
cd build/
cmake -DSUNDIALS_SUPERLUMT_ENABLE=ON ..
make
```

## 11.2 Using AMICI's C++ interface

The various import functions in of the *Python interface* and *Matlab interface* translate models defined in different formats into C++ code. These generated model libraries, together with the AMICI base library can be used in any C++ application for model simulation and sensitivity analysis. This section will give a short overview over the generated files and provide a brief introduction of how this code can be included in other applications. Further details are available in the *C++ API reference*.

### 11.2.1 AMICI-generated C++ model files

After importing a model using either the *Python interface* or the *Matlab interface*, the specified output directory contains (among others) C++ code for the various model functions.

The content of a model source directory looks something like this (given *MODEL\_NAME=model\_steadystate*):

```
CMakeLists.txt
main.cpp
deltaqB.cpp
deltaqB.h
[... many more files *.cpp|h|md5|o ]
wrapfunctions.cpp
wrapfunctions.h
model_steadystate.h
model_steadystate.cpp
```

These files provide the implementation of a model-specific subclass of `amici::Model`. The `CMakeLists.txt` file can be used to build the model library using `CMake`. `main.cpp` contains a simple scaffold for running a model simulation from C++. See next section for more details on these files.

### 11.2.2 Running a model simulation

AMICI's public API is mostly available through `amici/amici.h`. This is the only header file that needs to be included for basic usage. All functions there are declared within the *amici namespace*. Additionally, `amici/hdf5.h` and `amici/serialization.h` may be handy for specific use cases. The former provides some functions for reading and writing *HDF5* files, latter for serialization (requires *Boost*). All model-specific functions are defined in the namespace `model_$modelname`.

The main function for running an AMICI simulation is `amici::runAmiciSimulation()`. This function requires

- an instance of a `amici::Model` subclass as generated during model import. For the example `model_steadystate` the respective class is provided as `Model_model_steadystate` in `model_steadystate.h` in output directory for the given model.
- a `amici::Solver` instance. This solver instance needs to match the requirements of the model and can be obtained from `amici::AbstractModel::getSolver()`.
- optionally an `amici::ExpData` instance, which contains any experimental data (e.g. measurements, noise model parameters or model inputs) to evaluate residuals or an objective function.

This function returns a `amici::ReturnData` object, which contains all simulation results.

For running simulations for multiple experimental conditions (multiple `amici::ExpData` instances), `amici::runAmiciSimulations()` provides an alternative entry point. If AMICI (and your application) have been compiled with OpenMP support (see installation guide), this allows for running those simulations in parallel.

A scaffold for a standalone simulation program is automatically generated during model import in `main.cpp` in the model output directory. This program shows how to use the above-mentioned classes, how to obtain the simulation results, and may provide a starting point for your own simulation code.

#### Working with multiple or anonymous models

AMICI model import generates a `amici::Model` subclass for the specific model, based on the name used during import. On the one hand, this allows you to use multiple models with different names within a single application. On the other hand, this requires you to know the name of the model, which can be inconvenient in some cases.

When working with a single model, the `wrapfunctions.h` file generated during model import can be used to avoid specifying model names explicitly. It defines a function `amici::generic_model::getModel()`, that returns an instance of the model class by a generic name.

---

**Note:** Including multiple `wrapfunctions.h` files from different models in a single application is not possible. When using multiple models, explicit names have to be used or the different model libraries need to be loaded dynamically at runtime.

---

### 11.2.3 Compiling and linking

To run AMICI simulations from within your C++ application, you need to compile and link the following libraries:

- model library
- AMICI base library
- SUNDIALS libraries
- SuiteSparse libraries
- CBLAS-compatible BLAS

- optionally HDF5 (C, HL, and CXX components) set CMake option `ENABLE_HDF5` to OFF to build without HDF5-support
- optionally OpenMP (for parallel simulation of multiple conditions, see `amici::runAmiciSimulations()`)
- optionally boost (only when using serialization of AMICI object)

The simplest and recommended way is using the provide CMake files which take care of all these dependencies.

Considering the simple case, that you want to simulate one specific model in your CMake-based C++ application, you can copy or move the generated model directory containing the `CMakeLists.txt` file to your application directory, add `add_subdirectory(yourModelDirectory)` to your project's `CMakeLists.txt` file and build your project using CMake as usual.

### 11.2.4 Parameter estimation for AMICI models in high-performance computing environments

To perform parameter estimation for large or otherwise computationally demanding AMICI models from C++ in a high-performance computing environment, you may find the [parPE library](#) helpful. parPE allows for the private or shared memory parallel evaluation of a cost function requiring multiple simulations of the same model with different inputs. It provides interfaces to different optimizers, such as Ipopt.

## 11.3 AMICI C++ API

AMICI C++ library functions

### 11.3.1 Class Hierarchy

### 11.3.2 File Hierarchy

### 11.3.3 Full API

#### Namespaces

#### Namespace amici

##### Contents

- *Namespaces*
- *Classes*
- *Enums*
- *Functions*
- *Typedefs*
- *Variables*

## Namespaces

- Namespace *amici::hdf5*

## Classes

- Struct *LogItem*
- Struct *ModelDimensions*
- Struct *ModelState*
- Struct *ModelStateDerived*
- Struct *SimulationState*
- Class *AbstractModel*
- Class *AbstractSpline*
- Class *AmiException*
- Class *AmiVector*
- Class *AmiVectorArray*
- Class *BackwardProblem*
- Class *ConditionContext*
- Class *ContextManager*
- Class *CpuTimer*
- Class *CvodeException*
- Class *CVodeSolver*
- Class *ExpData*
- Class *FinalStateStorer*
- Class *ForwardProblem*
- Class *HermiteSpline*
- Class *IDAException*
- Class *IDASolver*
- Class *IntegrationFailure*
- Class *IntegrationFailureB*
- Class *Logger*
- Class *Model*
- Class *Model\_DAE*
- Class *Model\_ODE*
- Class *ModelContext*
- Class *NewtonFailure*
- Class *NewtonSolver*

- *Class NewtonSolverDense*
- *Class NewtonSolverSparse*
- *Class ReturnData*
- *Class SetupFailure*
- *Class SimulationParameters*
- *Class Solver*
- *Class SteadystateProblem*
- *Class SUNLinSolBand*
- *Class SUNLinSolDense*
- *Class SUNLinSolKLU*
- *Class SUNLinSolPCG*
- *Class SUNLinSolSPBCGS*
- *Class SUNLinSolSPFGMR*
- *Class SUNLinSolSPGMR*
- *Class SUNLinSolSPTFQMR*
- *Class SUNLinSolWrapper*
- *Class SUNMatrixWrapper*
- *Class SUNNonLinSolFixedPoint*
- *Class SUNNonLinSolNewton*
- *Class SUNNonLinSolWrapper*

## **Enums**

- *Enum BLASLayout*
- *Enum BLASTranspose*
- *Enum FixedParameterContext*
- *Enum InternalSensitivityMethod*
- *Enum InterpolationType*
- *Enum LinearMultistepMethod*
- *Enum LinearSolver*
- *Enum LogSeverity*
- *Enum ModelQuantity*
- *Enum NewtonDampingFactorMode*
- *Enum NonlinearSolverIteration*
- *Enum ObservableScaling*
- *Enum ParameterScaling*
- *Enum RDataReporting*

- *Enum SecondOrderMode*
- *Enum SensitivityMethod*
- *Enum SensitivityOrder*
- *Enum SplineBoundaryCondition*
- *Enum SplineExtrapolation*
- *Enum SteadyStateComputationMode*
- *Enum SteadyStateContext*
- *Enum SteadyStateSensitivityMode*
- *Enum SteadyStateStatus*

## Functions

- *Template Function amici::addSlice(const gsl::span<T const>, gsl::span<T>)*
- *Template Function amici::addSlice(std::vector<T> const&, gsl::span<T>)*
- *Function amici::amici\_daxpy*
- *Function amici::amici\_dgemm*
- *Function amici::amici\_dgemv*
- *Function amici::backtraceString*
- *Template Function amici::checkBufferSize*
- *Function amici::checkSigmaPositivity(std::vector<realtype> const&, char const \*)*
- *Function amici::checkSigmaPositivity(realtype, char const \*)*
- *Template Function amici::deserializeFromChar*
- *Template Function amici::deserializeFromString*
- *Function amici::dotProd*
- *Function amici::getScaledParameter*
- *Function amici::getUnscaledParameter*
- *Template Function amici::is\_equal*
- *Function amici::linearSum*
- *Function amici::N\_VGetArrayPointerConst*
- *Function amici::operator==(ExpData const&, ExpData const&)*
- *Function amici::operator==(Model const&, Model const&)*
- *Function amici::operator==(ModelDimensions const&, ModelDimensions const&)*
- *Function amici::operator==(ModelState const&, ModelState const&)*
- *Function amici::operator==(SimulationParameters const&, SimulationParameters const&)*
- *Function amici::operator==(Solver const&, Solver const&)*
- *Function amici::printfToString*
- *Function amici::regexErrorToString*

- *Function amici::runAmiciSimulation*
- *Function amici::runAmiciSimulations*
- *Function amici::scaleParameters*
- *Template Function amici::serializeToChar*
- *Template Function amici::serializeToStdVec*
- *Template Function amici::serializeToString*
- *Function amici::simulation\_status\_to\_str*
- *Template Function amici::slice(std::vector<T>&, int, unsigned)*
- *Template Function amici::slice(std::vector<T> const&, int, unsigned)*
- *Function amici::unravel\_index(size\_t, size\_t)*
- *Function amici::unravel\_index(sunindextype, SUNMatrix)*
- *Function amici::unscaleParameters*
- *Function amici::wrapErrHandlerFn*
- *Template Function amici::writeSlice(const gsl::span<T const>, gsl::span<T>)*
- *Template Function amici::writeSlice(std::vector<T> const&, std::vector<T>&)*
- *Template Function amici::writeSlice(std::vector<T> const&, gsl::span<T>)*
- *Function amici::writeSlice(AmiVector const&, gsl::span<realtype>)*

## Typedefs

- *Typedef amici::const\_N\_Vector*
- *Typedef amici::realtype*

## Variables

- *Variable amici::AMICI\_CONV\_FAILURE*
- *Variable amici::AMICI\_DAMPING\_FACTOR\_ERROR*
- *Variable amici::AMICI\_DATA\_RETURN*
- *Variable amici::AMICI\_ERR\_FAILURE*
- *Variable amici::AMICI\_ERROR*
- *Variable amici::AMICI\_FIRST\_RHSFUNC\_ERR*
- *Variable amici::AMICI\_ILL\_INPUT*
- *Variable amici::AMICI\_MAX\_TIME\_EXCEEDED*
- *Variable amici::AMICI\_NO\_STEADY\_STATE*
- *Variable amici::AMICI\_NORMAL*
- *Variable amici::AMICI\_NOT\_IMPLEMENTED*
- *Variable amici::AMICI\_NOT\_RUN*



- Variable `amici::AMICI_ONE_STEP`
- Variable `amici::AMICI_ONEOUTPUT`
- Variable `amici::AMICI_PREEQUILIBRATE`
- Variable `amici::AMICI_RECOVERABLE_ERROR`
- Variable `amici::AMICI_RHSFUNC_FAIL`
- Variable `amici::AMICI_ROOT_RETURN`
- Variable `amici::AMICI_SINGULAR_JACOBIAN`
- Variable `amici::AMICI_SUCCESS`
- Variable `amici::AMICI_TOO_MUCH_ACC`
- Variable `amici::AMICI_TOO_MUCH_WORK`
- Variable `amici::AMICI_UNRECOVERABLE_ERROR`
- Variable `amici::model_quantity_to_str`
- Variable `amici::pi`

## Namespace `amici::hdf5`

### Contents

- *Functions*

## Functions

- Function `amici::hdf5::attributeExists(H5::H5File const&, std::string const&, std::string const&)`
- Function `amici::hdf5::attributeExists(H5::H5Object const&, std::string const&)`
- Function `amici::hdf5::createAndWriteDouble1DDataset`
- Function `amici::hdf5::createAndWriteDouble2DDataset`
- Function `amici::hdf5::createAndWriteDouble3DDataset`
- Function `amici::hdf5::createAndWriteInt1DDataset`
- Function `amici::hdf5::createAndWriteInt2DDataset`
- Function `amici::hdf5::createGroup`
- Function `amici::hdf5::createOrOpenForWriting`
- Function `amici::hdf5::getDoubleDataset1D`
- Function `amici::hdf5::getDoubleDataset2D`
- Function `amici::hdf5::getDoubleDataset3D`
- Function `amici::hdf5::getDoubleScalarAttribute`
- Function `amici::hdf5::getIntDataset1D`
- Function `amici::hdf5::getIntScalarAttribute`

- *Function amici::hdf5::getStringAttribute*
- *Function amici::hdf5::locationExists(std::string const&, std::string const&)*
- *Function amici::hdf5::locationExists(H5::H5File const&, std::string const&)*
- *Function amici::hdf5::readModelDataFromHDF5(H5::H5File const&, Model&, std::string const&)*
- *Function amici::hdf5::readModelDataFromHDF5(std::string const&, Model&, std::string const&)*
- *Function amici::hdf5::readSimulationExpData*
- *Function amici::hdf5::readSolverSettingsFromHDF5(const H5::H5File&, Solver&, std::string const&)*
- *Function amici::hdf5::readSolverSettingsFromHDF5(std::string const&, Solver&, std::string const&)*
- *Function amici::hdf5::writeReturnData(ReturnData const&, H5::H5File const&, std::string const&)*
- *Function amici::hdf5::writeReturnData(ReturnData const&, std::string const&, std::string const&)*
- *Function amici::hdf5::writeReturnDataDiagnosis*
- *Function amici::hdf5::writeSimulationExpData*
- *Function amici::hdf5::writeSolverSettingsToHDF5(Solver const&, std::string const&, std::string const&)*
- *Function amici::hdf5::writeSolverSettingsToHDF5(Solver const&, H5::H5File const&, std::string const&)*

## Namespace boost

### Contents

- *Namespaces*

## Namespaces

- *Namespace boost::serialization*

## Namespace boost::serialization

### Contents

- *Functions*

## Functions

- *Template Function* `boost::serialization::archiveVector`
- *Template Function* `boost::serialization::serialize(Archive&, amici::IDASolver&, unsigned int)`
- *Template Function* `boost::serialization::serialize(Archive&, amici::ReturnData&, unsigned int)`
- *Template Function* `boost::serialization::serialize(Archive&, amici::Model&, unsigned int)`
- *Template Function* `boost::serialization::serialize(Archive&, amici::Solver&, unsigned int)`
- *Template Function* `boost::serialization::serialize(Archive&, amici::CvodeSolver&, unsigned int)`

## Namespace gsl

### Contents

- *Functions*

## Functions

- *Function* `gsl::make_span(SUNMatrix)`
- *Function* `gsl::make_span(N_Vector)`

## Namespace std

STL namespace.

## Classes and Structs

### Struct LogItem

- Defined in `file_include_amici_logging.h`

### Struct Documentation

struct **LogItem**

A log item.

## Public Functions

**LogItem()** = default

Default ctor.

inline **LogItem**(*LogSeverity* severity, std::string const &identifier, std::string const &message)

Construct a *LogItem*.

### Parameters

- **severity** –
- **identifier** –
- **message** –

## Public Members

*LogSeverity* **severity**

Severity level

std::string **identifier**

Short identifier for the logged event

std::string **message**

A more detailed and readable message

## Struct ModelDimensions

- Defined in file\_include\_amici\_model\_dimensions.h

## Inheritance Relationships

### Derived Types

- public amici::Model (*Class Model*)
- public amici::ReturnData (*Class ReturnData*)

## Struct Documentation

struct **ModelDimensions**

Container for model dimensions.

Holds number of states, observables, etc.

Subclassed by *amici::Model*, *amici::ReturnData*

## Public Functions

**ModelDimensions()** = default

Default ctor

```
inline ModelDimensions(int const nx_rdata, int const nxtrue_rdata, int const nx_solver, int const
    nxtrue_solver, int const nx_solver_reinit, int const np, int const nk, int const ny, int
    const nytrue, int const nz, int const nztrue, int const ne, int const nspl, int const nJ,
    int const nw, int const ndwdx, int const ndwdp, int const ndwdw, int const ndxdotdw,
    std::vector<int> ndJydy, int const ndxrdatadxsolver, int const ndxrdatadtcl, int const
    ndtotal_cldx_rdata, int const nnz, int const ubw, int const lbw)
```

Constructor with model dimensions.

### Parameters

- **nx\_rdata** – Number of state variables
- **nxtrue\_rdata** – Number of state variables of the non-augmented model
- **nx\_solver** – Number of state variables with conservation laws applied
- **nxtrue\_solver** – Number of state variables of the non-augmented model with conservation laws applied
- **nx\_solver\_reinit** – Number of state variables with conservation laws subject to reinitialization
- **np** – Number of parameters
- **nk** – Number of constants
- **ny** – Number of observables
- **nytrue** – Number of observables of the non-augmented model
- **nz** – Number of event observables
- **nztrue** – Number of event observables of the non-augmented model
- **ne** – Number of events
- **nspl** – Number of splines
- **nJ** – Number of objective functions
- **nw** – Number of repeating elements
- **ndwdx** – Number of nonzero elements in the  $x$  derivative of the repeating elements
- **ndwdp** – Number of nonzero elements in the  $p$  derivative of the repeating elements
- **ndwdw** – Number of nonzero elements in the  $w$  derivative of the repeating elements
- **ndxdotdw** – Number of nonzero elements in the  $w$  derivative of  $x_{dot}$
- **ndJydy** – Number of nonzero elements in the  $y$  derivative of  $dJy$  (shape **nytrue**)
- **ndxrdatadxsolver** – Number of nonzero elements in the  $x$  derivative of  $x_rdata$
- **ndxrdatadtcl** – Number of nonzero elements in the  $tcl$  derivative of  $x_rdata$
- **ndtotal\_cldx\_rdata** – Number of nonzero elements in the  $x_rdata$  derivative of  $total_{cl}$
- **nnz** – Number of nonzero elements in Jacobian
- **ubw** – Upper matrix bandwidth in the Jacobian

- **lbw** – Lower matrix bandwidth in the Jacobian

## Public Members

int **nx\_rdata** = {0}

Number of states

int **nxtrue\_rdata** = {0}

Number of states in the unaugmented system

int **nx\_solver** = {0}

Number of states with conservation laws applied

int **nxtrue\_solver** = {0}

Number of states in the unaugmented system with conservation laws applied

int **nx\_solver\_reinit** = {0}

Number of solver states subject to reinitialization

int **np** = {0}

Number of parameters

int **nk** = {0}

Number of constants

int **ny** = {0}

Number of observables

int **nytrue** = {0}

Number of observables in the unaugmented system

int **nz** = {0}

Number of event outputs

int **nztrue** = {0}

Number of event outputs in the unaugmented system

int **ne** = {0}

Number of events

int **nspl** = {0}

number of spline functions in the model

int **nw** = {0}

Number of common expressions

int **ndwdx** = {0}

Number of nonzero elements in the  $x$  derivative of the repeating elements

int **ndwdp** = {0}

Number of nonzero elements in the  $p$  derivative of the repeating elements

int **ndwdw** = {0}

Number of nonzero elements in the  $w$  derivative of the repeating elements

int **ndxdotdw** = {0}

Number of nonzero elements in the  $w$  derivative of  $x_{dot}$

std::vector<int> **ndJydy**

Number of nonzero elements in the  $y$  derivative of  $dJy$  (dimension `nytrue`)

int **ndxrdatadxsolver** = {0}

Number of nonzero elements in the  $x$  derivative of  $x_rdata$

int **ndxrdatadtcl** = {0}

Number of nonzero elements in the  $tcl$  derivative of  $x_rdata$

int **ndtotal\_cldx\_rdata** = {0}

Number of nonzero elements in the  $x_rdata$  derivative of  $total_{c_l}$

int **nnz** = {0}

Number of nonzero entries in Jacobian

int **nJ** = {0}

Dimension of the augmented objective function for 2nd order ASA

int **ubw** = {0}

Upper bandwidth of the Jacobian

int **lbw** = {0}

Lower bandwidth of the Jacobian

## Struct ModelState

- Defined in `file_include_amici_model_state.h`

## Struct Documentation

### struct **ModelState**

Exchange format to store and transfer the state of the model at a specific timepoint.

This is designed to only encompass the minimal number of attributes that need to be transferred.

#### Public Members

`std::vector<realtype> h`

Flag indicating whether a certain Heaviside function should be active or not (dimension: ne)

`std::vector<realtype> total_cl`

Total abundances for conservation laws (dimension: `nx_rdata` - `nx_solver`)

`std::vector<realtype> stotal_cl`

Sensitivities of total abundances for conservation laws (dimension: `(nx_rdata-nx_solver)` x `np`, row-major)

`std::vector<realtype> unscaledParameters`

Unscaled parameters (dimension: `np`)

`std::vector<realtype> fixedParameters`

Constants (dimension: `nk`)

`std::vector<int> plist`

Indexes of parameters wrt to which sensitivities are computed (dimension: `nplist`)

`std::vector<realtype> spl_`

temporary storage for spline values

### Struct **ModelStateDerived**

- Defined in file `_include_amici_model_state.h`

## Struct Documentation

### struct **ModelStateDerived**

Storage for `amici::Model` quantities computed based on `amici::ModelState` for a specific timepoint.

Serves as workspace for a model simulation to avoid repeated reallocation.



## Public Functions

**ModelStateDerived()** = default

explicit **ModelStateDerived**(*ModelDimensions* const &dim)

Constructor from model dimensions.

**Parameters** **dim** – *Model* dimensions

## Public Members

*SUNMatrixWrapper* **J\_**

Sparse Jacobian (dimension: `nx_solver` x `nx_solver`, nnz: `amici::Model::nnz`)

*SUNMatrixWrapper* **JB\_**

Sparse Backwards Jacobian (dimension: `nx_solver` x `nx_solver`, nnz: `amici::Model::nnz`)

*SUNMatrixWrapper* **dxdotdw\_**

Sparse dxdotdw temporary storage (dimension: `nx_solver` x `nw`, nnz: `ndxdotdw`)

*SUNMatrixWrapper* **dwdx\_**

Sparse dwdx temporary storage (dimension: `nw` x `nx_solver`, nnz: `ndwdx`)

*SUNMatrixWrapper* **dwdp\_**

Sparse dwdp temporary storage (dimension: `nw` x `np`, nnz: `ndwdp`)

*SUNMatrixWrapper* **M\_**

Dense Mass matrix (dimension: `nx_solver` x `nx_solver`)

*SUNMatrixWrapper* **MSparse\_**

Sparse Mass matrix (dimension: `nx_solver` x `nx_solver`, nnz: `sum(amici::Model::idlist)`)

*SUNMatrixWrapper* **dfdx\_**

JSparse intermediate matrix (dimension: `nx_solver` x `nx_solver`, nnz: dynamic)

*SUNMatrixWrapper* **dxdotdp\_full**

Temporary storage of `dxdotdp_full` data across functions (Python only) (dimension: `nplist` x `nx_solver`, nnz: dynamic, type `CSC_MAT`)

*SUNMatrixWrapper* **dxdotdp\_explicit**

Temporary storage of `dxdotdp_explicit` data across functions (Python only) (dimension: `nplist` x `nx_solver`, nnz: `ndxdotdp_explicit`, type `CSC_MAT`)

*SUNMatrixWrapper* **dxdotdp\_implicit**

Temporary storage of `dxdotdp_implicit` data across functions, Python-only (dimension: `nplist` x `nx_solver`, nnz: dynamic, type `CSC_MAT`)

***SUNMatrixWrapper* dxdotdx\_explicit**

Temporary storage of dxdotdx\_explicit data across functions (Python only) (dimension: nplist x nx\_solver, nnz: nxdotdotdx\_explicit, type CSC\_MAT)

***SUNMatrixWrapper* dxdotdx\_implicit**

Temporary storage of dxdotdx\_implicit data across functions, Python-only (dimension: nplist x nx\_solver, nnz: dynamic, type CSC\_MAT)

***SUNMatrixWrapper* dx\_rdatadx\_solver**

Temporary storage for dx\_rdatadx\_solver (dimension: nx\_rdata x nx\_solver, nnz: ndxrdatadx\_solver, type: CSC\_MAT)

***SUNMatrixWrapper* dx\_rdatadtcl**

Temporary storage for dx\_rdatadtcl (dimension: nx\_rdata x ncl, nnz: ndxrdatadtclr, type: CSC\_MAT)

***SUNMatrixWrapper* dtotal\_cldx\_rdata**

Temporary storage for dtotal\_cldx\_rdata (dimension: ncl x nx\_rdata, nnz: ndtotal\_cldx\_rdata, type: CSC\_MAT)

***AmiVectorArray* dxdotdp = {0, 0}**

Temporary storage of dxdotdp data across functions, Matlab only (dimension: nplist x nx\_solver , row-major)

**std::vector<*SUNMatrixWrapper*> dJydy\_**

Sparse observable derivative of data likelihood, only used if pythonGenerated == true (dimension nytrue, nJ x ny, row-major)

**std::vector<*realtype*> dJydy\_matlab\_**

Observable derivative of data likelihood, only used if pythonGenerated == false (dimension nJ x ny x nytrue , row-major)

**std::vector<*realtype*> dJydsigma\_**

Observable sigma derivative of data likelihood (dimension nJ x ny x nytrue, row-major)

**std::vector<*realtype*> dJydx\_**

State derivative of data likelihood (dimension nJ x nx\_solver, row-major)

**std::vector<*realtype*> dJydp\_**

Parameter derivative of data likelihood for current timepoint (dimension: nJ x nplist, row-major)

**std::vector<*realtype*> dJzdz\_**

event output derivative of event likelihood (dimension nJ x nz x nztrue, row-major)

**std::vector<*realtype*> dJzdsigma\_**

event sigma derivative of event likelihood (dimension nJ x nz x nztrue, row-major)

`std::vector<realtype> dJrzdz_`  
 event output derivative of event likelihood at final timepoint (dimension nJ x nz x nztrue, row-major)

`std::vector<realtype> dJrzdsigma_`  
 event sigma derivative of event likelihood at final timepoint (dimension nJ x nz x nztrue, row-major)

`std::vector<realtype> dJzdx_`  
 state derivative of event likelihood (dimension nJ x nx\_solver, row-major)

`std::vector<realtype> dJzdp_`  
 parameter derivative of event likelihood for current timepoint (dimension: nJ x nplist x, row-major)

`std::vector<realtype> dzdx_`  
 state derivative of event output (dimension: nz x nx\_solver, row-major)

`std::vector<realtype> dzdp_`  
 parameter derivative of event output (dimension: nz x nplist, row-major)

`std::vector<realtype> drzdx_`  
 state derivative of event regularization variable (dimension: nz x nx\_solver, row-major)

`std::vector<realtype> drzdp_`  
 parameter derivative of event regularization variable (dimension: nz x nplist, row-major)

`std::vector<realtype> dydp_`  
 parameter derivative of observable (dimension: ny x nplist, row-major)

`std::vector<realtype> dydx_`  
 state derivative of time-resolved observable (dimension: nx\_solver x ny, row-major)

`std::vector<realtype> w_`  
 temporary storage of w data across functions (dimension: nw)

`std::vector<realtype> sx_`  
 temporary storage for flattened sx, (dimension: nx\_solver x nplist, row-major)

`std::vector<realtype> sy_`  
 temporary storage for sy, (dimension: ny x nplist, row-major)

`std::vector<realtype> x_rdata_`  
 temporary storage for x\_rdata (dimension: nx\_rdata)

`std::vector<realtype> sx_rdata_`  
 temporary storage for sx\_rdata slice (dimension: nx\_rdata)

`std::vector<realtype> y_`  
temporary storage for time-resolved observable (dimension: ny)

`std::vector<realtype> sigmay_`  
data standard deviation for current timepoint (dimension: ny)

`std::vector<realtype> dsigmaydp_`  
temporary storage for parameter derivative of data standard deviation, (dimension: ny x nplist, row-major)

`std::vector<realtype> dsigmaydy_`  
temporary storage for observable derivative of data standard deviation, (dimension: ny x ny, row-major)

`std::vector<realtype> z_`  
temporary storage for event-resolved observable (dimension: nz)

`std::vector<realtype> rz_`  
temporary storage for event regularization (dimension: nz)

`std::vector<realtype> sigmaz_`  
temporary storage for event standard deviation (dimension: nz)

`std::vector<realtype> dsigmazdp_`  
temporary storage for parameter derivative of event standard deviation, (dimension: nz x nplist, row-major)

`std::vector<realtype> deltax_`  
temporary storage for change in x after event (dimension: nx\_solver)

`std::vector<realtype> deltasx_`  
temporary storage for change in sx after event (dimension: nx\_solver x nplist, row-major)

`std::vector<realtype> deltaxB_`  
temporary storage for change in xB after event (dimension: nx\_solver)

`std::vector<realtype> deltaqB_`  
temporary storage for change in qB after event (dimension: nJ x nplist, row-major)

*SUNMatrixWrapper* `sspl_`  
temporary storage for sensitivity values of splines

*AmiVector* `x_pos_tmp_ = {0}`  
temporary storage of positified state variables according to stateIsNonNegative (dimension: nx\_solver)

## Struct `SimulationState`

- Defined in `file_include_amici_model_state.h`

## Struct Documentation

struct **SimulationState**

implements an exchange format to store and transfer the state of a simulation at a specific timepoint.

### Public Members

*realtype* **t**

timepoint

*AmiVector* **x**

state variables

*AmiVector* **dx**

state variables

*AmiVectorArray* **sx**

state variable sensitivity

*ModelState* **state**

state of the model that was used for simulation

## Class `AbstractModel`

- Defined in `file_include_amici_abstract_model.h`

## Inheritance Relationships

### Derived Type

- `public amici::Model` (*Class Model*)

## Class Documentation

### class **AbstractModel**

Abstract base class of *amici::Model* defining functions that need to be implemented in an AMICI model.

Some functions have empty default implementations or throw. This class shall not have any data members.

Subclassed by *amici::Model*

### Public Functions

virtual **~AbstractModel**() = default

virtual std::unique\_ptr<*Solver*> **getSolver**() = 0

Retrieves the solver object.

**Returns** The *Solver* instance

virtual void **froot**(const *realtype* t, *AmiVector* const &x, *AmiVector* const &dx, gsl::span<*realtype*> root) = 0

Root function.

#### Parameters

- **t** – time
- **x** – state
- **dx** – time derivative of state (DAE only)
- **root** – array to which values of the root function will be written

virtual void **fxdot**(const *realtype* t, *AmiVector* const &x, *AmiVector* const &dx, *AmiVector* &xdot) = 0

Residual function.

#### Parameters

- **t** – time
- **x** – state
- **dx** – time derivative of state (DAE only)
- **xdot** – array to which values of the residual function will be written

virtual void **fsxdot**(const *realtype* t, *AmiVector* const &x, *AmiVector* const &dx, int ip, *AmiVector* const &sx, *AmiVector* const &sdx, *AmiVector* &sxdot) = 0

Sensitivity Residual function.

#### Parameters

- **t** – time
- **x** – state
- **dx** – time derivative of state (DAE only)
- **ip** – parameter index
- **sx** – sensitivity state
- **sdx** – time derivative of sensitivity state (DAE only)
- **sxdot** – array to which values of the sensitivity residual function will be written

```
virtual void fxBdot_ss(const realtype t, AmiVector const &xB, AmiVector const &dxB, AmiVector &xBdot)
    = 0
```

Residual function backward when running in steady state mode.

#### Parameters

- **t** – time
- **xB** – adjoint state
- **dxB** – time derivative of state (DAE only)
- **xBdot** – array to which values of the residual function will be written

```
virtual void fJSparseB_ss(SUNMatrix JB) = 0
```

Sparse Jacobian function backward, steady state case.

**Parameters** **JB** – sparse matrix to which values of the Jacobian will be written

```
virtual void writeSteadystateJB(const realtype t, realtype cj, AmiVector const &x, AmiVector const &dx,
    AmiVector const &xB, AmiVector const &dxB, AmiVector const &xBdot)
    = 0
```

Computes the sparse backward Jacobian for steadystate integration and writes it to the model member.

#### Parameters

- **t** – timepoint
- **cj** – scalar in Jacobian
- **x** – Vector with the states
- **dx** – Vector with the derivative states
- **xB** – Vector with the adjoint states
- **dxB** – Vector with the adjoint derivative states
- **xBdot** – Vector with the adjoint state right hand side

```
virtual void fJ(const realtype t, realtype cj, AmiVector const &x, AmiVector const &dx, AmiVector const
    &xdot, SUNMatrix J) = 0
```

Dense Jacobian function.

#### Parameters

- **t** – time
- **cj** – scaling factor (inverse of timestep, DAE only)
- **x** – state
- **dx** – time derivative of state (DAE only)
- **xdot** – values of residual function (unused)
- **J** – dense matrix to which values of the jacobian will be written

```
virtual void fJB(const realtype t, realtype cj, AmiVector const &x, AmiVector const &dx, AmiVector const
    &xB, AmiVector const &dxB, AmiVector const &xBdot, SUNMatrix JB) = 0
```

Dense Jacobian function.

#### Parameters

- **t** – time

- **cj** – scaling factor (inverse of timestep, DAE only)
- **x** – state
- **dx** – time derivative of state (DAE only)
- **xB** – Vector with the adjoint states
- **dxB** – Vector with the adjoint derivative states
- **xBdot** – Vector with the adjoint right hand side (unused)
- **JB** – dense matrix to which values of the jacobian will be written

virtual void **fJSparse**(const *realtype* t, *realtype* cj, *AmiVector* const &x, *AmiVector* const &dx, *AmiVector* const &xdot, SUNMatrix J) = 0

Sparse Jacobian function.

#### Parameters

- **t** – time
- **cj** – scaling factor (inverse of timestep, DAE only)
- **x** – state
- **dx** – time derivative of state (DAE only)
- **xdot** – values of residual function (unused)
- **J** – sparse matrix to which values of the Jacobian will be written

virtual void **fJSparseB**(const *realtype* t, *realtype* cj, *AmiVector* const &x, *AmiVector* const &dx, *AmiVector* const &xB, *AmiVector* const &dxB, *AmiVector* const &xBdot, SUNMatrix JB) = 0

Sparse Jacobian function.

#### Parameters

- **t** – time
- **cj** – scaling factor (inverse of timestep, DAE only)
- **x** – state
- **dx** – time derivative of state (DAE only)
- **xB** – Vector with the adjoint states
- **dxB** – Vector with the adjoint derivative states
- **xBdot** – Vector with the adjoint right hand side (unused)
- **JB** – dense matrix to which values of the jacobian will be written

virtual void **fJDiag**(const *realtype* t, *AmiVector* &Jdiag, *realtype* cj, *AmiVector* const &x, *AmiVector* const &dx) = 0

Diagonal Jacobian function.

#### Parameters

- **t** – time
- **Jdiag** – array to which the diagonal of the Jacobian will be written
- **cj** – scaling factor (inverse of timestep, DAE only)
- **x** – state



- **dx** – time derivative of state (DAE only)

virtual void **fdxdotdp**(const *realtype* t, *AmiVector* const &x, *AmiVector* const &dx) = 0

Model-specific sparse implementation of explicit parameter derivative of right hand side.

#### Parameters

- **t** – time
- **x** – state
- **dx** – time derivative of state (DAE only)

virtual void **fJv**(const *realtype* t, *AmiVector* const &x, *AmiVector* const &dx, *AmiVector* const &xdot, *AmiVector* const &v, *AmiVector* &nJv, *realtype* cj) = 0

Jacobian multiply function.

#### Parameters

- **t** – time
- **x** – state
- **dx** – time derivative of state (DAE only)
- **xdot** – values of residual function (unused)
- **v** – multiplication vector (unused)
- **nJv** – array to which result of multiplication will be written
- **cj** – scaling factor (inverse of timestep, DAE only)

virtual std::string **getAmiciVersion**() const

Returns the AMICI version that was used to generate the model.

**Returns** AMICI version string

virtual std::string **getAmiciCommit**() const

Returns the AMICI commit that was used to generate the model.

**Returns** AMICI commit string

virtual void **fx0**(*realtype* \*x0, const *realtype* t, *realtype* const \*p, *realtype* const \*k)

Model-specific implementation of fx0.

#### Parameters

- **x0** – initial state
- **t** – initial time
- **p** – parameter vector
- **k** – constant vector

virtual bool **isFixedParameterStateReinitializationAllowed**() const

Function indicating whether reinitialization of states depending on fixed parameters is permissible.

**Returns** flag indicating whether reinitialization of states depending on fixed parameters is permissible

virtual void **fx0\_fixedParameters**(*realtype* \*x0, const *realtype* t, *realtype* const \*p, *realtype* const \*k, gsl::span<int const> reinitialization\_state\_idx)

Model-specific implementation of fx0\_fixedParameters.

**Parameters**

- **x0** – initial state
- **t** – initial time
- **p** – parameter vector
- **k** – constant vector
- **reinitialization\_state\_idx**s – Indices of states to be reinitialized based on provided constants / fixed parameters.

```
virtual void fsx0_fixedParameters(realtype *sx0, const realtype t, realtype const *x0, realtype const *p,  
                                realtype const *k, int ip, gsl::span<int const>  
                                reinitialization_state_idx)
```

Model-specific implementation of fsx0\_fixedParameters.

**Parameters**

- **sx0** – initial state sensitivities
- **t** – initial time
- **x0** – initial state
- **p** – parameter vector
- **k** – constant vector
- **ip** – sensitivity index
- **reinitialization\_state\_idx**s – Indices of states to be reinitialized based on provided constants / fixed parameters.

```
virtual void fsx0(realtype *sx0, const realtype t, realtype const *x0, realtype const *p, realtype const *k, int  
                ip)
```

Model-specific implementation of fsx0.

**Parameters**

- **sx0** – initial state sensitivities
- **t** – initial time
- **x0** – initial state
- **p** – parameter vector
- **k** – constant vector
- **ip** – sensitivity index

```
virtual void fdx0(AmiVector &x0, AmiVector &dx0)
```

Initial value for time derivative of states (only necessary for DAEs)

**Parameters**

- **x0** – Vector with the initial states
- **dx0** – Vector to which the initial derivative states will be written (only DAE)

```
virtual void fstau(realtype *stau, const realtype t, realtype const *x, realtype const *p, realtype const *k,  
                 realtype const *h, realtype const *tcl, realtype const *sx, int ip, int ie)
```

Model-specific implementation of fstau.

**Parameters**

- **stau** – total derivative of event timepoint
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **tcl** – total abundances for conservation laws
- **sx** – current state sensitivity
- **ip** – sensitivity index
- **ie** – event index

virtual void **fy**(*realtype* \*y, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w)

Model-specific implementation of fy.

#### Parameters

- **y** – model output at current timepoint
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **w** – repeating elements vector

virtual void **fdydp**(*realtype* \*dydp, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, int ip, *realtype* const \*w, *realtype* const \*dwdp)

Model-specific implementation of fdydp (MATLAB-only)

#### Parameters

- **dydp** – partial derivative of observables y w.r.t. model parameters p
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **ip** – parameter index w.r.t. which the derivative is requested
- **w** – repeating elements vector
- **dwdp** – Recurring terms in xdot, parameter derivative

virtual void **fdydp**(*realtype* \*dydp, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, int ip, *realtype* const \*w, *realtype* const \*tcl, *realtype* const \*dtcl, *realtype* const \*spl, *realtype* const \*sspl)

Model-specific implementation of fdydp (Python)

**Parameters**

- **dydp** – partial derivative of observables  $y$  w.r.t. model parameters  $p$
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **ip** – parameter index w.r.t. which the derivative is requested
- **w** – repeating elements vector
- **tcl** – total abundances for conservation laws
- **dtcldp** – Sensitivities of total abundances for conservation laws
- **spl** – spline value vector
- **sspl** – sensitivities of spline values vector w.r.t. parameters  $p$

virtual void **fdydx**(*realtype* \*dydx, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w, *realtype* const \*dwdx)

Model-specific implementation of fdydx.

**Parameters**

- **dydx** – partial derivative of observables  $y$  w.r.t. model states  $x$
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **w** – repeating elements vector
- **dwdx** – Recurring terms in  $\dot{x}$ , state derivative

virtual void **fz**(*realtype* \*z, int ie, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h)

Model-specific implementation of fz.

**Parameters**

- **z** – value of event output
- **ie** – event index
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector

virtual void **fsz**(*realtype* \*sz, int ie, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*sx, int ip)

Model-specific implementation of fsz.

#### Parameters

- **sz** – Sensitivity of rz, total derivative
- **ie** – event index
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **sx** – current state sensitivity
- **ip** – sensitivity index

virtual void **frz**(*realtype* \*rz, int ie, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h)

Model-specific implementation of frz.

#### Parameters

- **rz** – value of root function at current timepoint (non-output events not included)
- **ie** – event index
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector

virtual void **fsrz**(*realtype* \*srz, int ie, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*sx, int ip)

Model-specific implementation of fsrz.

#### Parameters

- **srz** – Sensitivity of rz, total derivative
- **ie** – event index
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **sx** – current state sensitivity
- **h** – Heaviside vector
- **ip** – sensitivity index

```
virtual void fdzdp(realtype *dzdp, int ie, const realtype t, realtype const *x, realtype const *p, realtype const *k, realtype const *h, int ip)
```

Model-specific implementation of fdzdp.

#### Parameters

- **dzdp** – partial derivative of event-resolved output z w.r.t. model parameters p
- **ie** – event index
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **ip** – parameter index w.r.t. which the derivative is requested

```
virtual void fdzdx(realtype *dzdx, int ie, const realtype t, realtype const *x, realtype const *p, realtype const *k, realtype const *h)
```

Model-specific implementation of fdzdx.

#### Parameters

- **dzdx** – partial derivative of event-resolved output z w.r.t. model states x
- **ie** – event index
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector

```
virtual void fdrzdp(realtype *drzdp, int ie, const realtype t, realtype const *x, realtype const *p, realtype const *k, realtype const *h, int ip)
```

Model-specific implementation of fdrzdp.

#### Parameters

- **drzdp** – partial derivative of root output rz w.r.t. model parameters p
- **ie** – event index
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **ip** – parameter index w.r.t. which the derivative is requested

virtual void **fdrzdx**(*realtype* \*drzdx, int ie, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h)

Model-specific implementation of fdrzdx.

#### Parameters

- **drzdx** – partial derivative of root output rz w.r.t. model states x
- **ie** – event index
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector

virtual void **fdeltax**(*realtype* \*deltax, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, int ie, *realtype* const \*xdot, *realtype* const \*xdot\_old)

Model-specific implementation of fdeltax.

#### Parameters

- **deltax** – state update
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **ie** – event index
- **xdot** – new model right hand side
- **xdot\_old** – previous model right hand side

virtual void **fdeltasx**(*realtype* \*deltasx, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w, int ip, int ie, *realtype* const \*xdot, *realtype* const \*xdot\_old, *realtype* const \*sx, *realtype* const \*stau, *realtype* const \*tcl)

Model-specific implementation of fdeltasx.

#### Parameters

- **deltasx** – sensitivity update
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **w** – repeating elements vector
- **ip** – sensitivity index
- **ie** – event index

- **xdot** – new model right hand side
- **xdot\_old** – previous model right hand side
- **sx** – state sensitivity
- **stau** – event-time sensitivity
- **tcl** – total abundances for conservation laws

```
virtual void fdeltaxB(realtype *deltaxB, const realtype t, realtype const *x, realtype const *p, realtype const *k, realtype const *h, int ie, realtype const *xdot, realtype const *xdot_old, realtype const *xB)
```

Model-specific implementation of fdeltaxB.

#### Parameters

- **deltaxB** – adjoint state update
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **ie** – event index
- **xdot** – new model right hand side
- **xdot\_old** – previous model right hand side
- **xB** – current adjoint state

```
virtual void fdeltaqB(realtype *deltaqB, const realtype t, realtype const *x, realtype const *p, realtype const *k, realtype const *h, int ip, int ie, realtype const *xdot, realtype const *xdot_old, realtype const *xB)
```

Model-specific implementation of fdeltaqB.

#### Parameters

- **deltaqB** – sensitivity update
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **ip** – sensitivity index
- **ie** – event index
- **xdot** – new model right hand side
- **xdot\_old** – previous model right hand side
- **xB** – adjoint state



virtual void **fsgimay**(*realtype* \*sigmay, const *realtype* t, *realtype* const \*p, *realtype* const \*k, *realtype* const \*y)

Model-specific implementation of fsgimay.

#### Parameters

- **sigmay** – standard deviation of measurements
- **t** – current time
- **p** – parameter vector
- **k** – constant vector
- **y** – model output at timepoint t

virtual void **fdsigmaydp**(*realtype* \*dsigmaydp, const *realtype* t, *realtype* const \*p, *realtype* const \*k, *realtype* const \*y, int ip)

Model-specific implementation of fdsigmaydp.

#### Parameters

- **dsigmaydp** – partial derivative of standard deviation of measurements
- **t** – current time
- **p** – parameter vector
- **k** – constant vector
- **y** – model output at timepoint t
- **ip** – sensitivity index

virtual void **fdsigmaydy**(*realtype* \*dsigmaydy, const *realtype* t, *realtype* const \*p, *realtype* const \*k, *realtype* const \*y)

Model-specific implementation of fsgimay.

#### Parameters

- **dsigmaydy** – partial derivative of standard deviation of measurements w.r.t. model outputs
- **t** – current time
- **p** – parameter vector
- **k** – constant vector
- **y** – model output at timepoint t

virtual void **fsgimaz**(*realtype* \*sigmaz, const *realtype* t, *realtype* const \*p, *realtype* const \*k)

Model-specific implementation of fsgimaz.

#### Parameters

- **sigmaz** – standard deviation of event measurements
- **t** – current time
- **p** – parameter vector
- **k** – constant vector

virtual void **fdsigmazdp**(*realtype* \*dsigmazdp, const *realtype* t, *realtype* const \*p, *realtype* const \*k, int ip)

Model-specific implementation of fsgimaz.

#### Parameters

- **dsigmazdp** – partial derivative of standard deviation of event measurements
- **t** – current time
- **p** – parameter vector
- **k** – constant vector
- **ip** – sensitivity index

virtual void **fJy**(*realtype* \*nllh, int iy, *realtype* const \*p, *realtype* const \*k, *realtype* const \*y, *realtype* const \*sigmay, *realtype* const \*my)

Model-specific implementation of fJy.

#### Parameters

- **nllh** – negative log-likelihood for measurements y
- **iy** – output index
- **p** – parameter vector
- **k** – constant vector
- **y** – model output at timepoint
- **sigmay** – measurement standard deviation at timepoint
- **my** – measurements at timepoint

virtual void **fJz**(*realtype* \*nllh, int iz, *realtype* const \*p, *realtype* const \*k, *realtype* const \*z, *realtype* const \*sigmaz, *realtype* const \*mz)

Model-specific implementation of fJz.

#### Parameters

- **nllh** – negative log-likelihood for event measurements z
- **iz** – event output index
- **p** – parameter vector
- **k** – constant vector
- **z** – model event output at timepoint
- **sigmaz** – event measurement standard deviation at timepoint
- **mz** – event measurements at timepoint

virtual void **fJrz**(*realtype* \*nllh, int iz, *realtype* const \*p, *realtype* const \*k, *realtype* const \*z, *realtype* const \*sigmaz)

Model-specific implementation of fJrz.

#### Parameters

- **nllh** – regularization for event measurements z
- **iz** – event output index
- **p** – parameter vector
- **k** – constant vector
- **z** – model event output at timepoint
- **sigmaz** – event measurement standard deviation at timepoint

virtual void **fdJydy**(*realtype* \*dJydy, int iy, *realtype* const \*p, *realtype* const \*k, *realtype* const \*y, *realtype* const \*sigmay, *realtype* const \*my)

Model-specific implementation of fdJydy.

#### Parameters

- **dJydy** – partial derivative of time-resolved measurement negative log-likelihood Jy
- **iy** – output index
- **p** – parameter vector
- **k** – constant vector
- **y** – model output at timepoint
- **sigmay** – measurement standard deviation at timepoint
- **my** – measurement at timepoint

virtual void **fdJydy\_colptrs**(*SUNMatrixWrapper* &dJydy, int index)

Model-specific implementation of fdJydy colptrs.

#### Parameters

- **dJydy** – sparse matrix to which colptrs will be written
- **index** – ytrue index

virtual void **fdJydy\_rowvals**(*SUNMatrixWrapper* &dJydy, int index)

Model-specific implementation of fdJydy rowvals.

#### Parameters

- **dJydy** – sparse matrix to which rowvals will be written
- **index** – ytrue index

virtual void **fdJydsigma**(*realtype* \*dJydsigma, int iy, *realtype* const \*p, *realtype* const \*k, *realtype* const \*y, *realtype* const \*sigmay, *realtype* const \*my)

Model-specific implementation of fdJydsigma.

#### Parameters

- **dJydsigma** – Sensitivity of time-resolved measurement negative log-likelihood Jy w.r.t. standard deviation sigmay
- **iy** – output index
- **p** – parameter vector
- **k** – constant vector
- **y** – model output at timepoint
- **sigmay** – measurement standard deviation at timepoint
- **my** – measurement at timepoint

virtual void **fdJzdz**(*realtype* \*dJzdz, int iz, *realtype* const \*p, *realtype* const \*k, *realtype* const \*z, *realtype* const \*sigmaz, *realtype* const \*mz)

Model-specific implementation of fdJzdz.

#### Parameters

- **dJzdz** – partial derivative of event measurement negative log-likelihood Jz

- **iz** – event output index
- **p** – parameter vector
- **k** – constant vector
- **z** – model event output at timepoint
- **sigmaz** – event measurement standard deviation at timepoint
- **mz** – event measurement at timepoint

virtual void **fdJzdsigma**(*realtype* \*dJzdsigma, int iz, *realtype* const \*p, *realtype* const \*k, *realtype* const \*z, *realtype* const \*sigmaz, *realtype* const \*mz)

Model-specific implementation of fdJzdsigma.

#### Parameters

- **dJzdsigma** – Sensitivity of event measurement negative log-likelihood Jz w.r.t. standard deviation sigma<sub>z</sub>
- **iz** – event output index
- **p** – parameter vector
- **k** – constant vector
- **z** – model event output at timepoint
- **sigmaz** – event measurement standard deviation at timepoint
- **mz** – event measurement at timepoint

virtual void **fdJrzdz**(*realtype* \*dJrzdz, int iz, *realtype* const \*p, *realtype* const \*k, *realtype* const \*rz, *realtype* const \*sigmaz)

Model-specific implementation of fdJrzdz.

#### Parameters

- **dJrzdz** – partial derivative of event penalization Jrz
- **iz** – event output index
- **p** – parameter vector
- **k** – constant vector
- **rz** – model root output at timepoint
- **sigmaz** – event measurement standard deviation at timepoint

virtual void **fdJrzdsigma**(*realtype* \*dJrzdsigma, int iz, *realtype* const \*p, *realtype* const \*k, *realtype* const \*rz, *realtype* const \*sigmaz)

Model-specific implementation of fdJrzdsigma.

#### Parameters

- **dJrzdsigma** – Sensitivity of event penalization Jrz w.r.t. standard deviation sigma<sub>z</sub>
- **iz** – event output index
- **p** – parameter vector
- **k** – constant vector
- **rz** – model root output at timepoint
- **sigmaz** – event measurement standard deviation at timepoint

virtual void **fw**(*realtype* \*w, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*tcl, *realtype* const \*spl)

Model-specific implementation of fw.

#### Parameters

- **w** – Recurring terms in xdot
- **t** – timepoint
- **x** – vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **tcl** – total abundances for conservation laws
- **spl** – spline value vector

virtual void **fdwdp**(*realtype* \*dwdp, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w, *realtype* const \*tcl, *realtype* const \*stcl, *realtype* const \*spl, *realtype* const \*sspl)

Model-specific sparse implementation of dwdp.

#### Parameters

- **dwdp** – Recurring terms in xdot, parameter derivative
- **t** – timepoint
- **x** – vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **w** – vector with helper variables
- **tcl** – total abundances for conservation laws
- **stcl** – sensitivities of total abundances for conservation laws
- **spl** – spline value vector
- **sspl** – sensitivities of spline values vector w.r.t. parameters *p*

virtual void **fdwdp\_colptrs**(*SUNMatrixWrapper* &dwdp)

Model-specific implementation for dwdp, column pointers.

**Parameters** **dwdp** – sparse matrix to which colptrs will be written

virtual void **fdwdp\_rowvals**(*SUNMatrixWrapper* &dwdp)

Model-specific implementation for dwdp, row values.

**Parameters** **dwdp** – sparse matrix to which rowvals will be written

virtual void **fdwdp**(*realtype* \*dwdp, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w, *realtype* const \*tcl, *realtype* const \*stcl, *realtype* const \*spl, *realtype* const \*sspl, int ip)

Model-specific sensitivity implementation of dwdp.

**Parameters**

- **dwdp** – Recurring terms in xdot, parameter derivative
- **t** – timepoint
- **x** – vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **w** – vector with helper variables
- **tcl** – total abundances for conservation laws
- **stcl** – sensitivities of total abundances for conservation laws
- **spl** – spline value vector
- **sspl** – sensitivities of spline values vector
- **ip** – sensitivity parameter index

virtual void **fdwdx**(*realtype* \*dwdx, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w, *realtype* const \*tcl, *realtype* const \*spl)

Model-specific implementation of dwdx, data part.

**Parameters**

- **dwdx** – Recurring terms in xdot, state derivative
- **t** – timepoint
- **x** – vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **w** – vector with helper variables
- **tcl** – total abundances for conservation laws
- **spl** – spline value vector

virtual void **fdwdx\_colptrs**(*SUNMatrixWrapper* &dwdx)

Model-specific implementation for dwdx, column pointers.

**Parameters** **dwdx** – sparse matrix to which colptrs will be written

virtual void **fdwdx\_rowvals**(*SUNMatrixWrapper* &dwdx)

Model-specific implementation for dwdx, row values.

**Parameters** **dwdx** – sparse matrix to which rowvals will be written

virtual void **fdwdw**(*realtype* \*dwdw, *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w, *realtype* const \*tcl)

Model-specific implementation of fdwdw, no w chainrule (Py)

**Parameters**

- **dwdw** – partial derivative w wrt w

- **t** – timepoint
- **x** – Vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **w** – vector with helper variables
- **tcl** – Total abundances for conservation laws

virtual void **fdwdw\_colptrs**(*SUNMatrixWrapper* &dwdw)

Model-specific implementation of fdwdw, colptrs part.

**Parameters** **dwdw** – sparse matrix to which colptrs will be written

virtual void **fdwdw\_rowvals**(*SUNMatrixWrapper* &dwdw)

Model-specific implementation of fdwdw, rowvals part.

**Parameters** **dwdw** – sparse matrix to which rowvals will be written

virtual void **fdx\_rdatadx\_solver**(*realtype* \*dx\_rdatadx\_solver, *realtype* const \*x, *realtype* const \*tcl, *realtype* const \*p, *realtype* const \*k)

Compute dx\_rdata / dx\_solver.

**Parameters**

- **dx\_rdatadx\_solver** – dx\_rdata / dx\_solver
- **p** – parameter vector
- **k** – constant vector
- **x** – State variables with conservation laws applied
- **tcl** – Total abundances for conservation laws

virtual void **fdx\_rdatadx\_solver\_colptrs**(*SUNMatrixWrapper* &dxrdatadx\_solver)

Model-specific implementation of fdx\_rdatadx\_solver, colptrs part.

**Parameters** **dxrdatadx\_solver** – sparse matrix to which colptrs will be written

virtual void **fdx\_rdatadx\_solver\_rowvals**(*SUNMatrixWrapper* &dxrdatadx\_solver)

Model-specific implementation of fdx\_rdatadx\_solver, rowvals part.

**Parameters** **dxrdatadx\_solver** – sparse matrix to which rowvals will be written

virtual void **fdx\_rdatadp**(*realtype* \*dx\_rdatadp, *realtype* const \*x, *realtype* const \*tcl, *realtype* const \*p, *realtype* const \*k, int const ip)

Compute dx\_rdata / dp.

**Parameters**

- **dx\_rdatadp** – dx\_rdata / dp
- **p** – parameter vector
- **k** – constant vector
- **x** – State variables with conservation laws applied
- **tcl** – Total abundances for conservation laws
- **ip** – Sensitivity index

virtual void **fdx\_rdatadtcl**(*realtype* \*dx\_rdatadtcl, *realtype* const \*x, *realtype* const \*tcl, *realtype* const \*p, *realtype* const \*k)

Compute dx\_rdata / dtcl.

**Parameters**

- **dx\_rdatadtcl** – dx\_rdata / dtcl
- **p** – parameter vector
- **k** – constant vector
- **x** – State variables with conservation laws applied
- **tcl** – Total abundances for conservation laws

virtual void **fdx\_rdatadtcl\_colptrs**(*SUNMatrixWrapper* &dx\_rdatadtcl)

Model-specific implementation of fdx\_rdatadtcl, colptrs part.

**Parameters** **dx\_rdatadtcl** – sparse matrix to which colptrs will be written

virtual void **fdx\_rdatadtcl\_rowvals**(*SUNMatrixWrapper* &dx\_rdatadtcl)

Model-specific implementation of fdx\_rdatadtcl, rowvals part.

**Parameters** **dx\_rdatadtcl** – sparse matrix to which rowvals will be written

virtual void **fdtotal\_cldp**(*realtype* \*dtotal\_cldp, *realtype* const \*x\_rdata, *realtype* const \*p, *realtype* const \*k, int const ip)

Compute dtotal\_cl / dp.

**Parameters**

- **dtotal\_cldp** – dtotal\_cl / dp
- **x\_rdata** – State variables with conservation laws applied
- **p** – parameter vector
- **k** – constant vector
- **ip** – Sensitivity index

virtual void **fdtotal\_cldx\_rdata**(*realtype* \*dtotal\_cldx\_rdata, *realtype* const \*x\_rdata, *realtype* const \*p, *realtype* const \*k, *realtype* const \*tcl)

Compute dtotal\_cl / dx\_rdata.

**Parameters**

- **dtotal\_cldx\_rdata** – dtotal\_cl / dx\_rdata
- **x\_rdata** – State variables with conservation laws applied
- **p** – parameter vector
- **k** – constant vector
- **tcl** – Total abundances for conservation laws

virtual void **fdtotal\_cldx\_rdata\_colptrs**(*SUNMatrixWrapper* &dtotal\_cldx\_rdata)

Model-specific implementation of fdtotal\_cldx\_rdata, colptrs part.

**Parameters** **dtotal\_cldx\_rdata** – sparse matrix to which colptrs will be written



virtual void **fdtotal\_cldx\_rdata\_rowvals**(*SUNMatrixWrapper* &dtotal\_cldx\_rdata)

Model-specific implementation of fdtotal\_cldx\_rdata, rowvals part.

**Parameters** **dtotal\_cldx\_rdata** – sparse matrix to which rowvals will be written

virtual std::vector<*HermiteSpline*> **fcreate\_splines**(*realtype* const \*p, *realtype* const \*k)

Model-specific implementation of spline creation.

**Parameters**

- **p** – parameter vector
- **k** – constants vector

**Returns** Vector of splines used in the model

virtual void **fdspline\_valuesdp**(*realtype* \*dspline\_valuesdp, *realtype* const \*p, *realtype* const \*k, int const ip)

Model-specific implementation the parametric derivatives of spline node values.

**Parameters**

- **dspline\_valuesdp** – vector to which derivatives will be written
- **p** – parameter vector
- **k** – constants vector
- **ip** – Sensitivity index

virtual void **fdspline\_slopesdp**(*realtype* \*dspline\_slopesdp, *realtype* const \*p, *realtype* const \*k, int const ip)

Model-specific implementation the parametric derivatives of slopevalues at spline nodes.

**Parameters**

- **dspline\_slopesdp** – vector to which derivatives will be written
- **p** – parameter vector
- **k** – constants vector
- **ip** – Sensitivity index

## Class AbstractSpline

- Defined in file\_include\_amici\_splinefunctions.h

## Inheritance Relationships

## Derived Type

- public amici::HermiteSpline (*Class HermiteSpline*)

## Class Documentation

### class **AbstractSpline**

AMICI spline base class.

Instances of this class are created upon solver setup and the needed splines are set up (e.g., interpolation of the nodes is performed). Upon call to a spline function, only the evaluation of the spline polynomial is carried out.

Subclassed by *amici::HermiteSpline*

### Public Functions

**AbstractSpline()** = default

default constructor

**AbstractSpline**(std::vector<realtype> nodes, std::vector<realtype> node\_values, bool equidistant\_spacing, bool logarithmic\_parametrization)

Common constructor for *AbstractSpline* instances.

#### Parameters

- **nodes** – the nodes defining the position at which the value of the spline is known (if `equidistant_spacing` is true, it must contain only the first and the last node; the other nodes will be automatically inserted, assuming they are uniformly spaced)
- **node\_values** – the values assumed by the spline at the nodes
- **equidistant\_spacing** – whether equidistant nodes are to be computed
- **logarithmic\_parametrization** – if true, the spline interpolation will occur in log-space in order to ensure positivity of the interpolant (which strictly speaking will no longer be a spline)

virtual **~AbstractSpline()** = default

virtual void **compute\_coefficients()** = 0

Compute the coefficients for all polynomial segments of this spline.

virtual void **compute\_coefficients\_sensi**(int nplist, int spline\_offset, gsl::span<realtype> dvaluesdp, gsl::span<realtype> dslopesdp) = 0

Compute the coefficients for all polynomial segments of the derivatives of this spline with respect to the parameters.

---

#### Remark

The contents of `dvaluesdp` and `dslopesdp` may be modified by this function.

---

#### Parameters

- **nplist** – number of parameters
- **spline\_offset** – offset of this spline inside `dvaluesdp` and `dslopesdp`
- **dvaluesdp** – derivatives of the spline values with respect to the parameters (for all splines in the model, not just this one)

- **dslopesdp** – derivatives of the spline derivatives with respect to the parameters (for all splines in the model, not just this one)

*realtype* **get\_value**(const *realtype* t) const

Get the value of this spline at a given point.

**Parameters** **t** – point at which the spline is to be evaluated

**Returns** value of the spline at **t**

virtual *realtype* **get\_value\_scaled**(const *realtype* t) const = 0

Get the value of this spline at a given point in the scale in which interpolation is carried out (e.g., log-scale)

**Parameters** **t** – point at which the spline is to be evaluated

**Returns** scaled value of the spline at **t**

*realtype* **get\_node\_value**(int const i) const

Get the value of this spline at a given node.

**Parameters** **i** – index of the node at which the spline is to be evaluated

**Returns** value of the spline at the **i**-th node

*realtype* **get\_node\_value\_scaled**(int const i) const

Get the value of this spline at a given node in the scale in which interpolation is carried out (e.g., log-scale)

**Parameters** **i** – index of the node at which the spline is to be evaluated

**Returns** scaled value of the spline at the **i**-th node

*realtype* **get\_sensitivity**(const *realtype* t, int const ip) const

Get the derivative of this spline with respect to a given parameter at a given point.

**Parameters**

- **t** – point at which the sensitivity is to be evaluated
- **ip** – index of the parameter

**Returns** sensitivity of the spline with respect to the **ip**th parameter at **t**

*realtype* **get\_sensitivity**(const *realtype* t, int const ip, const *realtype* value) const

Get the derivative of this spline with respect to a given parameter at a given point.

**Parameters**

- **t** – point at which the sensitivity is to be evaluated
- **ip** – index of the parameter
- **value** – value of the spline at the given time point. It is used e.g. when interpolation is carried out in log-space. If omitted it will be computed.

**Returns** sensitivity of the spline with respect to the **ip**th parameter at **t**

virtual *realtype* **get\_sensitivity\_scaled**(const *realtype* t, int const ip) const = 0

Get the derivative of this spline with respect to a given parameter at a given point in the scale in which interpolation is carried out (e.g., log-scale)

**Parameters**

- **t** – point at which the sensitivity is to be evaluated
- **ip** – index of the parameter

**Returns** scaled sensitivity of the spline with respect to the `ip`th parameter at `t`

virtual void **compute\_final\_value**() = 0

Compute the limit value of the spline as the evaluation point tends to positive infinity.

virtual void **compute\_final\_sensitivity**(int nplist, int spline\_offset, gsl::span<*realtype*>  
dspline\_valuesdp, gsl::span<*realtype*> dspline\_slopesdp) = 0

Compute the limit of the value of the sensitivity as the evaluation point tends to positive infinity.

**Parameters**

- **nplist** – number of parameters
- **spline\_offset** – offset of this spline inside `dspline_valuesdp` and `dspline_slopesdp`
- **dspline\_valuesdp** – derivatives of the spline values with respect to the parameters (for all splines in the model, not just this one)
- **dspline\_slopesdp** – derivatives of the spline derivatives with respect to the parameters (for all splines in the model, not just this one)

*realtype* **get\_final\_value**() const

Get the limit value of the spline as the evaluation point tends to positive infinity.

**Returns** limit value

*realtype* **get\_final\_value\_scaled**() const

Get the limit value of the spline (in the scale in which interpolation is carried out) as the evaluation point tends to positive infinity.

**Returns** limit value

*realtype* **get\_final\_sensitivity**(int const ip) const

Get the limit value of the sensitivity with respect to the given parameter as the evaluation point tends to positive infinity.

**Parameters** **ip** – parameter index

**Returns** limit value

*realtype* **get\_final\_sensitivity\_scaled**(int const ip) const

Get the limit value of the sensitivity with respect to the given parameter (in the scale in which interpolation is carried out) as the evaluation point tends to positive infinity.

**Parameters** **ip** – parameter index

**Returns** limit value

bool **get\_equidistant\_spacing**() const

Whether nodes are uniformly spaced.

**Returns** boolean flag

bool **get\_logarithmic\_parametrization**() const

Whether spline interpolation is carried out in log-space.

**Returns** boolean flag

inline int **n\_nodes**() const

The number of interpolation nodes for this spline.

**Returns** number of nodes

## Protected Functions

void **set\_final\_value\_scaled**(*realtype* finalValue)

Set the limit value of the spline (in the scale in which interpolation is carried out) as the evaluation point tends to positive infinity.

**Parameters** **finalValue** – final value

void **set\_final\_sensitivity\_scaled**(std::vector<*realtype*> finalSensitivity)

Set the limit value of the sensitivity (in the scale in which interpolation is carried out) as the evaluation point tends to positive infinity.

**Parameters** **finalSensitivity** – final value of the sensitivity for each parameter

## Protected Attributes

std::vector<*realtype*> **nodes\_**

The nodes at which this spline is interpolated.

std::vector<*realtype*> **node\_values\_**

The values the spline assumes at the nodes.

std::vector<*realtype*> **coefficients**

Coefficients for each polynomial segment of the spline.

std::vector<*realtype*> **coefficients\_extrapolate**

Polynomial coefficients for the extrapolating the spline values.

std::vector<*realtype*> **coefficients\_sensi**

Coefficients for each polynomial segment of the sensitivities with respect to the parameters.

std::vector<*realtype*> **coefficients\_extrapolate\_sensi**

Polynomial coefficients for the extrapolating the sensitivities.

## Class AmiException

- Defined in file\_include\_amici\_exception.h

## Inheritance Relationships

### Base Type

- public std::exception

## Derived Types

- `public amici::CvodeException (Class CvodeException)`
- `public amici::IDAException (Class IDAException)`
- `public amici::IntegrationFailure (Class IntegrationFailure)`
- `public amici::IntegrationFailureB (Class IntegrationFailureB)`
- `public amici::NewtonFailure (Class NewtonFailure)`
- `public amici::SetupFailure (Class SetupFailure)`

## Class Documentation

class **AmiException** : public std::exception

AMICI exception class.

Has a printf style interface to allow easy generation of error messages

Subclassed by *amici::CvodeException*, *amici::IDAException*, *amici::IntegrationFailure*, *amici::IntegrationFailureB*, *amici::NewtonFailure*, *amici::SetupFailure*

### Public Functions

**AmiException**(int const first\_frame = 3)

Default ctor.

**Parameters** **first\_frame** – Index of first frame to include

explicit **AmiException**(char const \*fmt, ...)

Constructor with printf style interface.

**Parameters**

- **fmt** – error message with printf format
- **...** – printf formatting variables

char const \***what**() const noexcept override

Override of default error message function.

**Returns** msg error message

char const \***getBacktrace**() const

Returns the stored backtrace.

**Returns** trace backtrace

void **storeBacktrace**(int nMaxFrames, int const first\_frame)

Stores the current backtrace.

**Parameters**

- **nMaxFrames** – number of frames to go back in stacktrace
- **first\_frame** – Index of first frame to include

## Protected Functions

void **storeMessage**(char const \*fmt, va\_list argptr)

Store the provided message.

### Parameters

- **fmt** – error message with printf format
- **argptr** – pointer to variadic argument list

## Class AmiVector

- Defined in file\_include\_amici\_vector.h

## Class Documentation

class **AmiVector**

*AmiVector* class provides a generic interface to the NVector\_Serial struct

## Public Functions

**AmiVector**() = default

Default constructor.

inline explicit **AmiVector**(long int const length)

empty constructor

Creates an std::vector<realtype> and attaches the data pointer to a newly created N\_Vector\_Serial. Using N\_VMake\_Serial ensures that the N\_Vector module does not try to deallocate the data vector when calling N\_VDestroy\_Serial

**Parameters** **length** – number of elements in vector

inline explicit **AmiVector**(std::vector<realtype> rvec)

constructor from std::vector,

Moves data from std::vector and constructs an nvec that points to the data

**Parameters** **rvec** – vector from which the data will be moved

inline explicit **AmiVector**(gsl::span<realtype> rvec)

constructor from gsl::span,

Copy data from gsl::span and constructs a vector

**Parameters** **rvec** – vector from which the data will be copied

inline **AmiVector**(*AmiVector* const &vold)

copy constructor

**Parameters** **vold** – vector from which the data will be copied

inline **AmiVector**(*AmiVector* &&other) noexcept  
 move constructor

**Parameters** **other** – vector from which the data will be moved

**~AmiVector**()  
 destructor

*AmiVector* &**operator**=(*AmiVector* const &other)  
 copy assignment operator

**Parameters** **other** – right hand side

**Returns** left hand side

inline *AmiVector* &**operator**\*=(*AmiVector* const &multiplier)  
 operator \*= (element-wise multiplication)

**Parameters** **multiplier** – multiplier

**Returns** result

inline *AmiVector* &**operator**/=(*AmiVector* const &divisor)  
 operator /= (element-wise division)

**Parameters** **divisor** – divisor

**Returns** result

inline auto **begin**()  
 Returns an iterator that points to the first element of the vector.

**Returns** iterator that points to the first element

inline auto **end**()  
 Returns an iterator that points to one element after the last element of the vector.

**Returns** iterator that points to one element after the last element

*realtype* \***data**()  
 data accessor

**Returns** pointer to data array

*realtype* const \***data**() const  
 const data accessor

**Returns** const pointer to data array

N\_Vector **getNVector**()  
 N\_Vector accessor.

**Returns** N\_Vector

*const\_N\_Vector* **getNVector**() const  
 N\_Vector accessor.

**Returns** N\_Vector

std::vector<*realtype*> const &**getVector**() const  
 Vector accessor.

**Returns** Vector



int **getLength**() const  
 returns the length of the vector  
**Returns** length

void **zero**()  
 fills vector with zero values

void **minus**()  
 changes the sign of data elements

void **set**(*realtype* val)  
 sets all data elements to a specific value  
**Parameters** **val** – value for data elements

*realtype* &**operator**[](int pos)  
 accessor to data elements of the vector  
**Parameters** **pos** – index of element  
**Returns** element

*realtype* &**at**(int pos)  
 accessor to data elements of the vector  
**Parameters** **pos** – index of element  
**Returns** element

*realtype* const &**at**(int pos) const  
 accessor to data elements of the vector  
**Parameters** **pos** – index of element  
**Returns** element

void **copy**(*AmiVector* const &other)  
 copies data from another *AmiVector*  
**Parameters** **other** – data source

inline void **abs**()  
 Take absolute value (in-place)

## Class AmiVectorArray

- Defined in file\_include\_amici\_vector.h

## Class Documentation

class **AmiVectorArray**

*AmiVectorArray* class.

Provides a generic interface to arrays of NVector\_Serial structs

## Public Functions

**AmiVectorArray()** = default

Default constructor.

**AmiVectorArray**(long int length\_inner, long int length\_outer)

empty constructor

Creates an `std::vector<realtype>` and attaches the data pointer to a newly created `N_VectorArray` using `CloneVectorArrayEmpty` ensures that the `N_Vector` module does not try to deallocate the data vector when calling `N_VDestroyVectorArray_Serial`

### Parameters

- **length\_inner** – length of vectors
- **length\_outer** – number of vectors

**AmiVectorArray**(*AmiVectorArray* const &vaold)

copy constructor

**Parameters** **vaold** – object to copy from

**~AmiVectorArray()** = default

*AmiVectorArray* &**operator=**(*AmiVectorArray* const &other)

copy assignment operator

**Parameters** **other** – right hand side

**Returns** left hand side

*realtype* \***data**(int pos)

accessor to data of *AmiVector* elements

**Parameters** **pos** – index of *AmiVector*

**Returns** pointer to data array

*realtype* const \***data**(int pos) const

const accessor to data of *AmiVector* elements

**Parameters** **pos** – index of *AmiVector*

**Returns** const pointer to data array

*realtype* &**at**(int ipos, int jpos)

accessor to elements of *AmiVector* elements

### Parameters

- **ipos** – inner index in *AmiVector*
- **jpos** – outer index in *AmiVectorArray*

**Returns** element

*realtype* const &**at**(int ipos, int jpos) const

const accessor to elements of *AmiVector* elements

### Parameters

- **ipos** – inner index in *AmiVector*
- **jpos** – outer index in *AmiVectorArray*

**Returns** element

`N_Vector *getNVectorArray()`

accessor to `NVectorArray`

**Returns** `N_VectorArray`

`N_Vector getNVector(int pos)`

accessor to `NVector` element

**Parameters** `pos` – index of corresponding *AmiVector*

**Returns** `N_Vector`

`const N_Vector getNVector(int pos) const`

const accessor to `NVector` element

**Parameters** `pos` – index of corresponding *AmiVector*

**Returns** `N_Vector`

`AmiVector &operator[] (int pos)`

accessor to *AmiVector* elements

**Parameters** `pos` – index of *AmiVector*

**Returns** *AmiVector*

`AmiVector const &operator[] (int pos) const`

const accessor to *AmiVector* elements

**Parameters** `pos` – index of *AmiVector*

**Returns** const *AmiVector*

`int getLength() const`

length of *AmiVectorArray*

**Returns** length

`void zero()`

set every *AmiVector* in *AmiVectorArray* to zero

`void flatten_to_vector(std::vector<realtype> &vec) const`

flattens the *AmiVectorArray* to a vector in row-major format

**Parameters** `vec` – vector into which the *AmiVectorArray* will be flattened. Must have length equal to number of elements.

`void copy(AmiVectorArray const &other)`

copies data from another *AmiVectorArray*

**Parameters** `other` – data source

## Class BackwardProblem

- Defined in file\_include\_amici\_backwardproblem.h

## Class Documentation

### class **BackwardProblem**

class to solve backwards problems.

solves the backwards problem for adjoint sensitivity analysis and handles events and data-points

### Public Functions

explicit **BackwardProblem**(*ForwardProblem* const &fwd, *SteadystateProblem* const \*posteq)

Construct backward problem from forward problem.

#### Parameters

- **fwd** – pointer to corresponding forward problem
- **posteq** – pointer to postequilibration problem, can be nullptr

void **workBackwardProblem**()

Solve the backward problem.

If adjoint sensitivities are enabled this will also compute sensitivities. `workForwardProblem` must be called before this is function is called.

inline *realtype* **gett**() const

Accessor for current time t.

#### Returns t

inline int **getwhich**() const

Accessor for which.

#### Returns which

inline int \***getwhichptr**()

Accessor for pointer to which.

#### Returns which

inline std::vector<*realtype*> const &**getdJydx**() const

Accessor for dJydx.

#### Returns dJydx

inline *AmiVector* const &**getAdjointState**() const

Accessor for xB.

#### Returns xB

inline *AmiVector* const &**getAdjointQuadrature**() const

Accessor for xQB.

#### Returns xQB

## Class ConditionContext

- Defined in file\_include\_amici\_edata.h

## Inheritance Relationships

### Base Type

- public amici::ContextManager (*Class ContextManager*)

## Class Documentation

class **ConditionContext** : public amici::ContextManager

The *ConditionContext* class applies condition-specific *amici::Model* settings and restores them when going out of scope.

### Public Functions

explicit **ConditionContext**(*Model* \*model, *ExpData* const \*edata = nullptr, *FixedParameterContext* fpc = *FixedParameterContext::simulation*)

Apply condition-specific settings from edata to model while keeping a backup of the original values.

#### Parameters

- **model** –
- **edata** –
- **fpc** – flag indicating which fixedParameter from edata to apply

*ConditionContext* &**operator**=(*ConditionContext* const &other) = delete

**~ConditionContext**()

void **applyCondition**(*ExpData* const \*edata, *FixedParameterContext* fpc)

Apply condition-specific settings from edata to the constructor-supplied model, not changing the settings which were backed-up in the constructor call.

#### Parameters

- **edata** –
- **fpc** – flag indicating which fixedParameter from edata to apply

void **restore**()

Restore original settings on constructor-supplied *amici::Model*. Will be called during destruction. Explicit call is generally not necessary.

## Class ContextManager

- Defined in file\_include\_amici\_misc.h

## Inheritance Relationships

### Derived Types

- public amici::ConditionContext (*Class ConditionContext*)
- public amici::FinalStateStorer (*Class FinalStateStorer*)
- public amici::ModelContext (*Class ModelContext*)

## Class Documentation

### class ContextManager

Generic implementation for a context manager, explicitly deletes copy and move operators for derived classes.

Subclassed by *amici::ConditionContext*, *amici::FinalStateStorer*, *amici::ModelContext*

### Public Functions

**ContextManager**() = default

**ContextManager**(*ContextManager* &other) = delete

**ContextManager**(*ContextManager* &&other) = delete

## Class CpuTimer

- Defined in file\_include\_amici\_misc.h

## Class Documentation

### class CpuTimer

Tracks elapsed CPU time using std::clock.

### Public Functions

inline **CpuTimer**()

Constructor.

inline void **reset**()

Reset the timer.

inline double **elapsed\_seconds()** const

Get elapsed CPU time in seconds since initialization or last reset.

**Returns** CPU time in seconds

inline double **elapsed\_milliseconds()** const

Get elapsed CPU time in milliseconds since initialization or last reset.

**Returns** CPU time in milliseconds

### Public Static Attributes

static const bool **uses\_thread\_clock** = false

## Class CvodeException

- Defined in file\_include\_amici\_exception.h

### Inheritance Relationships

#### Base Type

- public amici::AmiException (*Class AmiException*)

### Class Documentation

class **CvodeException** : public amici::AmiException

cvode exception handler class

### Public Functions

**CvodeException**(int error\_code, char const \*function)

Constructor.

#### Parameters

- **error\_code** – error code returned by cvode function
- **function** – cvode function name

## Class CNodeSolver

- Defined in file\_include\_amici\_solver\_cvodes.h

## Inheritance Relationships

### Base Type

- public amici::Solver (*Class Solver*)

## Class Documentation

class **CNodeSolver** : public amici::*Solver*

The *CNodeSolver* class is a wrapper around the SUNDIALS CVODES solver.

### Public Functions

**~CNodeSolver()** override = default

virtual *Solver* \***clone()** const override

Clone this instance.

**Returns** The clone

virtual void **reInit**(*realtype* t0, *AmiVector* const &yy0, *AmiVector* const &yp0) const override

Reinitializes the states in the solver after an event occurrence.

#### Parameters

- **t0** – reinitialization timepoint
- **yy0** – initial state variables
- **yp0** – initial derivative state variables (DAE only)

virtual void **sensReInit**(*AmiVectorArray* const &yyS0, *AmiVectorArray* const &ypS0) const override

Reinitializes the state sensitivities in the solver after an event occurrence.

#### Parameters

- **yyS0** – new state sensitivity
- **ypS0** – new derivative state sensitivities (DAE only)

virtual void **sensToggleOff**() const override

Switches off computation of state sensitivities without deallocating the memory for sensitivities.

virtual void **reInitB**(int which, *realtype* tB0, *AmiVector* const &yyB0, *AmiVector* const &ypB0) const override

Reinitializes the adjoint states after an event occurrence.

#### Parameters

- **which** – identifier of the backwards problem
- **tB0** – reinitialization timepoint



- **yyB0** – new adjoint state
- **ypB0** – new adjoint derivative state

virtual void **quadReInitB**(int which, *AmiVector* const &yQB0) const override

Reinitialize the adjoint states after an event occurrence.

#### Parameters

- **which** – identifier of the backwards problem
- **yQB0** – new adjoint quadrature state

virtual int **solve**(*realtype* tout, int itask) const override

Solves the forward problem until a predefined timepoint.

#### Parameters

- **tout** – timepoint until which simulation should be performed
- **itask** – task identifier, can be CV\_NORMAL or CV\_ONE\_STEP

**Returns** status flag indicating success of execution

virtual int **solveF**(*realtype* tout, int itask, int \*ncheckPtr) const override

Solves the forward problem until a predefined timepoint (adjoint only)

#### Parameters

- **tout** – timepoint until which simulation should be performed
- **itask** – task identifier, can be CV\_NORMAL or CV\_ONE\_STEP
- **ncheckPtr** – pointer to a number that counts the internal checkpoints

**Returns** status flag indicating success of execution

virtual void **solveB**(*realtype* tBout, int itaskB) const override

Solves the backward problem until a predefined timepoint (adjoint only)

#### Parameters

- **tBout** – timepoint until which simulation should be performed
- **itaskB** – task identifier, can be CV\_NORMAL or CV\_ONE\_STEP

virtual void **getDky**(*realtype* t, int k) const override

interpolates the (derivative of the) solution at the requested timepoint

#### Parameters

- **t** – timepoint
- **k** – derivative order

virtual void **getSensDky**(*realtype* t, int k) const override

interpolates the (derivative of the) solution at the requested timepoint

#### Parameters

- **t** – timepoint
- **k** – derivative order

virtual void **getQuadDkyB**(*realtype* t, int k, int which) const override  
interpolates the (derivative of the) solution at the requested timepoint

**Parameters**

- **t** – timepoint
- **k** – derivative order
- **which** – index of backward problem

virtual void **getDkyB**(*realtype* t, int k, int which) const override  
interpolates the (derivative of the) solution at the requested timepoint

**Parameters**

- **t** – timepoint
- **k** – derivative order
- **which** – index of backward problem

virtual void **getRootInfo**(int \*rootsfound) const override  
getRootInfo extracts information which event occurred

**Parameters** **rootsfound** – array with flags indicating whether the respective event occurred

virtual void **setStopTime**(*realtype* tstop) const override  
Sets a timepoint at which the simulation will be stopped.

**Parameters** **tstop** – timepoint until which simulation should be performed

virtual void **turnOffRootFinding**() const override  
Disable rootfinding.

virtual *Model* const \***getModel**() const override  
Accessor function to the model stored in the user data

**Returns** user data model

virtual void **setLinearSolver**() const override  
Sets the linear solver for the forward problem.

virtual void **setLinearSolverB**(int which) const override  
Sets the linear solver for the backward problem.

**Parameters** **which** – index of the backward problem

virtual void **setNonLinearSolver**() const override  
Set the non-linear solver for the forward problem.

virtual void **setNonLinearSolverSens**() const override  
Set the non-linear solver for sensitivities.

virtual void **setNonLinearSolverB**(int which) const override  
Set the non-linear solver for the backward problem.

**Parameters** **which** – index of the backward problem

**Solver**() = default  
Default constructor.

**Solver**(*Solver* const &other)

*Solver* copy constructor.

**Parameters** *other* –

## Protected Functions

virtual void **calcIC**(*realtype* tout1) const override

Calculates consistent initial conditions, assumes initial states to be correct (DAE only)

**Parameters** *tout1* – next timepoint to be computed (sets timescale)

virtual void **calcICB**(int which, *realtype* tout1) const override

Calculates consistent initial conditions for the backwards problem, assumes initial states to be correct (DAE only)

**Parameters**

- **which** – identifier of the backwards problem
- **tout1** – next timepoint to be computed (sets timescale)

virtual void **getB**(int which) const override

extracts the adjoint state at the current timepoint from solver memory and writes it to the xB member variable

**Parameters** *which* – index of the backwards problem

virtual void **getSens**() const override

extracts the state sensitivity at the current timepoint from solver memory and writes it to the sx member variable

virtual void **getQuadB**(int which) const override

extracts the adjoint quadrature state at the current timepoint from solver memory and writes it to the xQB member variable

**Parameters** *which* – index of the backwards problem

virtual void **getQuad**(*realtype* &t) const override

extracts the quadrature at the current timepoint from solver memory and writes it to the xQ member variable

**Parameters** *t* – timepoint for quadrature extraction

virtual void **getQuadDky**(*realtype* t, int k) const override

interpolates the (derivative of the) solution at the requested timepoint

**Parameters**

- **t** – timepoint
- **k** – derivative order

virtual void **reInitPostProcessF**(*realtype* tnext) const override

reInitPostProcessF postprocessing of the solver memory after a discontinuity in the forward problem

**Parameters** *tnext* – next timepoint (defines integration direction)

virtual void **reInitPostProcessB**(*realtype* tnext) const override

reInitPostProcessB postprocessing of the solver memory after a discontinuity in the backward problem

**Parameters** *tnext* – next timepoint (defines integration direction)

void **reInitPostProcess**(void \*cv\_mem, *realtype* \*t, *AmiVector* \*yout, *realtype* tout) const

Postprocessing of the solver memory after a discontinuity.

**Parameters**

- **cv\_mem** – pointer to CVODES solver memory object
- **t** – pointer to integration time
- **yout** – new state vector
- **tout** – anticipated next integration timepoint.

virtual void **allocateSolver**() const override

Create specifies solver method and initializes solver memory for the forward problem.

virtual void **setSStolerances**(double rtol, double atol) const override

sets scalar relative and absolute tolerances for the forward problem

**Parameters**

- **rtol** – relative tolerances
- **atol** – absolute tolerances

virtual void **setSensSStolerances**(double rtol, double const \*atol) const override

activates sets scalar relative and absolute tolerances for the sensitivity variables

**Parameters**

- **rtol** – relative tolerances
- **atol** – array of absolute tolerances for every sensitivity variable

virtual void **setSensErrCon**(bool error\_corr) const override

SetSensErrCon specifies whether error control is also enforced for sensitivities for the forward problem

**Parameters** **error\_corr** – activation flag

virtual void **setQuadErrConB**(int which, bool flag) const override

Specifies whether error control is also enforced for the backward quadrature problem.

**Parameters**

- **which** – identifier of the backwards problem
- **flag** – activation flag

virtual void **setQuadErrCon**(bool flag) const override

Specifies whether error control is also enforced for the forward quadrature problem.

**Parameters** **flag** – activation flag

virtual void **setErrorHandlerFn**() const override

Attaches the error handler function (errMsgIdAndTxt) to the solver.

virtual void **setUserData**() const override

Attaches the user data to the forward problem.

virtual void **setUserDataB**(int which) const override

attaches the user data to the backward problem

**Parameters** **which** – identifier of the backwards problem

virtual void **setMaxNumSteps**(long int mxsteps) const override  
 specifies the maximum number of steps for the forward problem

---

**Note:** in contrast to the SUNDIALS method, this sets the overall maximum, not the maximum between output times.

---

**Parameters** **mxsteps** – number of steps

virtual void **setStabLimDet**(int stldet) const override  
 activates stability limit detection for the forward problem

**Parameters** **stldet** – flag for stability limit detection (TRUE or FALSE)

virtual void **setStabLimDetB**(int which, int stldet) const override  
 activates stability limit detection for the backward problem

**Parameters**

- **which** – identifier of the backwards problem
- **stldet** – flag for stability limit detection (TRUE or FALSE)

virtual void **setId**(*Model* const \*model) const override  
 specify algebraic/differential components (DAE only)

**Parameters** **model** – model specification

virtual void **setSuppressAlg**(bool flag) const override  
 deactivates error control for algebraic components (DAE only)

**Parameters** **flag** – deactivation flag

void **resetState**(void \*cv\_mem, *const N\_Vector* y0) const  
 resetState reset the CVODES solver to restart integration after a rhs discontinuity.

**Parameters**

- **cv\_mem** – pointer to CVODES solver memory object
- **y0** – new state vector

virtual void **setSensParams**(*realtype* const \*p, *realtype* const \*pbar, int const \*plist) const override  
 specifies the scaling and indexes for sensitivity computation

**Parameters**

- **p** – parameters
- **pbar** – parameter scaling constants
- **plist** – parameter index list

virtual void **adjInit**() const override  
 initializes the adjoint problem

virtual void **quadInit**(*AmiVector* const &xQ0) const override  
 initializes the quadratures

**Parameters** **xQ0** – vector with initial values for xQ

virtual void **allocateSolverB**(int \*which) const override

Specifies solver method and initializes solver memory for the backward problem.

**Parameters** **which** – identifier of the backwards problem

virtual void **setSStolerancesB**(int which, *realtype* relTolB, *realtype* absTolB) const override

sets relative and absolute tolerances for the backward problem

**Parameters**

- **which** – identifier of the backwards problem
- **relTolB** – relative tolerances
- **absTolB** – absolute tolerances

virtual void **quadSStolerancesB**(int which, *realtype* reltolQB, *realtype* abstolQB) const override

sets relative and absolute tolerances for the quadrature backward problem

**Parameters**

- **which** – identifier of the backwards problem
- **reltolQB** – relative tolerances
- **abstolQB** – absolute tolerances

virtual void **quadSStolerances**(*realtype* reltolQ, *realtype* abstolQ) const override

sets relative and absolute tolerances for the quadrature problem

**Parameters**

- **reltolQB** – relative tolerances
- **abstolQB** – absolute tolerances

virtual void **setMaxNumStepsB**(int which, long int mxstepsB) const override

specifies the maximum number of steps for the forward problem

---

**Note:** in contrast to the SUNDIALS method, this sets the overall maximum, not the maximum between output times.

---

**Parameters**

- **which** – identifier of the backwards problem
- **mxstepsB** – number of steps

virtual void **diag**() const override

attaches a diagonal linear solver to the forward problem

virtual void **diagB**(int which) const override

attaches a diagonal linear solver to the backward problem

**Parameters** **which** – identifier of the backwards problem

virtual void **getNumSteps**(void const \*ami\_mem, long int \*numsteps) const override

reports the number of solver steps

**Parameters**

- **ami\_mem** – pointer to the solver memory instance (can be from forward or backward problem)
- **numsteps** – output array

virtual void **getNumRhsevals**(void const \*ami\_mem, long int \*numrhsevals) const override  
reports the number of right hand evaluations

#### Parameters

- **ami\_mem** – pointer to the solver memory instance (can be from forward or backward problem)
- **numrhsevals** – output array

virtual void **getNumErrTestFails**(void const \*ami\_mem, long int \*numerrtestfails) const override  
reports the number of local error test failures

#### Parameters

- **ami\_mem** – pointer to the solver memory instance (can be from forward or backward problem)
- **numerrtestfails** – output array

virtual void **getNumNonlinSolvConvFails**(void const \*ami\_mem, long int \*numnonlinsolvconvfails) const override  
reports the number of nonlinear convergence failures

#### Parameters

- **ami\_mem** – pointer to the solver memory instance (can be from forward or backward problem)
- **numnonlinsolvconvfails** – output array

virtual void **getLastOrder**(void const \*ami\_mem, int \*order) const override  
Reports the order of the integration method during the last internal step.

#### Parameters

- **ami\_mem** – pointer to the solver memory instance (can be from forward or backward problem)
- **order** – output array

virtual void **\*getAdjBmem**(void \*ami\_mem, int which) const override  
Retrieves the solver memory instance for the backward problem.

#### Parameters

- **which** – identifier of the backwards problem
- **ami\_mem** – pointer to the forward solver memory instance

**Returns** A (void \*) pointer to the CVODES memory allocated for the backward problem.

virtual void **init**(*realtype* t0, *AmiVector* const &x0, *AmiVector* const &dx0) const override  
Initializes the states at the specified initial timepoint.

#### Parameters

- **t0** – initial timepoint
- **x0** – initial states

- **dx0** – initial derivative states

virtual void **initSteadystate**(const *realtype* t0, *AmiVector* const &x0, *AmiVector* const &dx0) const override

Initializes the states at the specified initial timepoint.

**Parameters**

- **t0** – initial timepoint
- **x0** – initial states
- **dx0** – initial derivative states

virtual void **sensInit1**(*AmiVectorArray* const &sx0, *AmiVectorArray* const &sdx0) const override

Initializes the forward sensitivities.

**Parameters**

- **sx0** – initial states sensitivities
- **sdx0** – initial derivative states sensitivities

virtual void **binit**(int which, *realtype* tf, *AmiVector* const &xB0, *AmiVector* const &dxB0) const override

Initialize the adjoint states at the specified final timepoint.

**Parameters**

- **which** – identifier of the backwards problem
- **tf** – final timepoint
- **xB0** – initial adjoint state
- **dxB0** – initial adjoint derivative state

virtual void **qbinit**(int which, *AmiVector* const &QB0) const override

Initialize the quadrature states at the specified final timepoint.

**Parameters**

- **which** – identifier of the backwards problem
- **QB0** – initial adjoint quadrature state

virtual void **rootInit**(int ne) const override

Initializes the rootfinding for events.

**Parameters** **ne** – number of different events

virtual void **setDenseJacFn**() const override

Set the dense Jacobian function.

virtual void **setSparseJacFn**() const override

sets the sparse Jacobian function

virtual void **setBandJacFn**() const override

sets the banded Jacobian function

virtual void **setJacTimesVecFn**() const override

sets the Jacobian vector multiplication function



virtual void **setDenseJacFnB**(int which) const override

sets the dense Jacobian function

**Parameters** **which** – identifier of the backwards problem

virtual void **setSparseJacFnB**(int which) const override

sets the sparse Jacobian function

**Parameters** **which** – identifier of the backwards problem

virtual void **setBandJacFnB**(int which) const override

sets the banded Jacobian function

**Parameters** **which** – identifier of the backwards problem

virtual void **setJacTimesVecFnB**(int which) const override

sets the Jacobian vector multiplication function

**Parameters** **which** – identifier of the backwards problem

virtual void **setSparseJacFn\_ss**() const override

sets the sparse Jacobian function for backward steady state case

## Friends

template<class **Archive**>

friend void **serialize**(*Archive* &ar, *CVodeSolver* &s, unsigned int)

Serialize *amici::CVodeSolver* to boost archive.

**Parameters**

- **ar** – Archive
- **s** – *Solver* instance to serialize

friend bool **operator==**(*CVodeSolver* const &a, *CVodeSolver* const &b)

Equality operator.

**Parameters**

- **a** –
- **b** –

**Returns** Whether a and b are equal

## Class ExpData

- Defined in file\_include\_amici\_edata.h

## Inheritance Relationships

### Base Type

- `public amici::SimulationParameters` (*Class SimulationParameters*)

### Class Documentation

class **ExpData** : public amici::SimulationParameters

*ExpData* carries all information about experimental or condition-specific data.

#### Public Functions

**ExpData**() = default  
default constructor

**ExpData**(*ExpData* const&) = default  
Copy constructor, needs to be declared to be generated in swig.

**ExpData**(int nytrue, int nztrue, int nmaxevent)  
constructor that only initializes dimensions

##### Parameters

- **nytrue** – Number of observables
- **nztrue** – Number of event outputs
- **nmaxevent** – Maximal number of events to track

**ExpData**(int nytrue, int nztrue, int nmaxevent, std::vector<*realtype*> ts)  
constructor that initializes timepoints from vectors

##### Parameters

- **nytrue** – Number of observables
- **nztrue** – Number of event outputs
- **nmaxevent** – Maximal number of events to track
- **ts** – Timepoints (dimension: nt)

**ExpData**(int nytrue, int nztrue, int nmaxevent, std::vector<*realtype*> ts, std::vector<*realtype*> fixedParameters)  
constructor that initializes timepoints and fixed parameters from vectors

##### Parameters

- **nytrue** – Number of observables
- **nztrue** – Number of event outputs
- **nmaxevent** – Maximal number of events to track
- **ts** – Timepoints (dimension: nt)
- **fixedParameters** – *Model* constants (dimension: nk)

**ExpData**(int nytrue, int nztrue, int nmaxevent, std::vector<realtype> ts, std::vector<realtype> const &observedData, std::vector<realtype> const &observedDataStdDev, std::vector<realtype> const &observedEvents, std::vector<realtype> const &observedEventsStdDev)

constructor that initializes timepoints and data from vectors

#### Parameters

- **nytrue** – Number of observables
- **nztrue** – Number of event outputs
- **nmaxevent** – Maximal number of events to track
- **ts** – Timepoints (dimension: nt)
- **observedData** – observed data (dimension: nt x nytrue, row-major)
- **observedDataStdDev** – standard deviation of observed data (dimension: nt x nytrue, row-major)
- **observedEvents** – observed events (dimension: nmaxevents x nztrue, row-major)
- **observedEventsStdDev** – standard deviation of observed events/roots (dimension: nmaxevents x nztrue, row-major)

explicit **ExpData**(*Model* const &model)

constructor that initializes with *Model*

**Parameters** *model* – pointer to model specification object

**ExpData**(*ReturnData* const &rdata, realtype sigma\_y, realtype sigma\_z)

constructor that initializes with returnData, adds noise according to specified sigmas

#### Parameters

- **rdata** – return data pointer with stored simulation results
- **sigma\_y** – scalar standard deviations for all observables
- **sigma\_z** – scalar standard deviations for all event observables

**ExpData**(*ReturnData* const &rdata, std::vector<realtype> sigma\_y, std::vector<realtype> sigma\_z)

constructor that initializes with returnData, adds noise according to specified sigmas

#### Parameters

- **rdata** – return data pointer with stored simulation results
- **sigma\_y** – vector of standard deviations for observables (dimension: nytrue or nt x nytrue, row-major)
- **sigma\_z** – vector of standard deviations for event observables (dimension: nztrue or nmax-event x nztrue, row-major)

**~ExpData**() = default

int **nytrue**() const

number of observables of the non-augmented model

**Returns** number of observables of the non-augmented model

int **nztrue**() const

number of event observables of the non-augmented model

**Returns** number of event observables of the non-augmented model

int **nmaxevent**() const

maximal number of events to track

**Returns** maximal number of events to track

int **nt**() const

number of timepoints

**Returns** number of timepoints

void **setTimepoints**(std::vector<*realtype*> const &ts)

Set output timepoints.

**Parameters** **ts** – timepoints

std::vector<*realtype*> const &**getTimepoints**() const

Get output timepoints.

**Returns** ExpData::ts

*realtype* **getTimepoint**(int it) const

Get timepoint for the given index.

**Parameters** **it** – timepoint index

**Returns** timepoint timepoint at index

void **setObservedData**(std::vector<*realtype*> const &observedData)

Set all measurements.

**Parameters** **observedData** – observed data (dimension: nt x nytrue, row-major)

void **setObservedData**(std::vector<*realtype*> const &observedData, int iy)

Set measurements for a given observable index.

**Parameters**

- **observedData** – observed data (dimension: nt)
- **iy** – observed data index

bool **isSetObservedData**(int it, int iy) const

Whether there is a measurement for the given time- and observable- index.

**Parameters**

- **it** – time index
- **iy** – observable index

**Returns** boolean specifying if data was set

std::vector<*realtype*> const &**getObservedData**() const

Get all measurements.

**Returns** observed data (dimension: nt x nytrue, row-major)

*realtype* const \***getObservedDataPtr**(int it) const

Get measurements for a given timepoint index.

**Parameters** **it** – timepoint index

**Returns** pointer to observed data at index (dimension: nytrue)

void **setObservedDataStdDev**(std::vector<*realtype*> const &observedDataStdDev)

Set standard deviations for measurements.

**Parameters** **observedDataStdDev** – standard deviation of observed data (dimension: nt x nytrue, row-major)

void **setObservedDataStdDev**(*realtype* stdDev)

Set identical standard deviation for all measurements.

**Parameters** **stdDev** – standard deviation (dimension: scalar)

void **setObservedDataStdDev**(std::vector<*realtype*> const &observedDataStdDev, int iy)

Set standard deviations of observed data for a specific observable index.

**Parameters**

- **observedDataStdDev** – standard deviation of observed data (dimension: nt)
- **iy** – observed data index

void **setObservedDataStdDev**(*realtype* stdDev, int iy)

Set all standard deviation for a given observable index to the input value.

**Parameters**

- **stdDev** – standard deviation (dimension: scalar)
- **iy** – observed data index

bool **isSetObservedDataStdDev**(int it, int iy) const

Whether standard deviation for a measurement at specified timepoint- and observable index has been set.

**Parameters**

- **it** – time index
- **iy** – observable index

**Returns** boolean specifying if standard deviation of data was set

std::vector<*realtype*> const &**getObservedDataStdDev**() const

Get measurement standard deviations.

**Returns** standard deviation of observed data

*realtype* const \***getObservedDataStdDevPtr**(int it) const

Get pointer to measurement standard deviations.

**Parameters** **it** – timepoint index

**Returns** pointer to standard deviation of observed data at index

void **setObservedEvents**(std::vector<*realtype*> const &observedEvents)

set function that copies observed event data from input to ExpData::observedEvents

**Parameters** **observedEvents** – observed data (dimension: nmaxevent x nztrue, row-major)

void **setObservedEvents**(std::vector<*realtype*> const &observedEvents, int iz)

set function that copies observed event data for specific event observable

**Parameters**

- **observedEvents** – observed data (dimension: nmaxevent)
- **iz** – observed event data index

bool **isSetObservedEvents**(int ie, int iz) const

get function that checks whether event data at specified indices has been set

**Parameters**

- **ie** – event index
- **iz** – event observable index

**Returns** boolean specifying if data was set

std::vector<*realtype*> const &**getObservedEvents**() const

get function that copies data from ExpData::mz to output

**Returns** observed event data

*realtype* const \***getObservedEventsPtr**(int ie) const

get function that returns a pointer to observed data at ieth occurrence

**Parameters** **ie** – event occurrence

**Returns** pointer to observed event data at ieth occurrence

void **setObservedEventsStdDev**(std::vector<*realtype*> const &observedEventsStdDev)

set function that copies data from input to ExpData::observedEventsStdDev

**Parameters** **observedEventsStdDev** – standard deviation of observed event data

void **setObservedEventsStdDev**(*realtype* stdDev)

set function that sets all ExpData::observedDataStdDev to the input value

**Parameters** **stdDev** – standard deviation (dimension: scalar)

void **setObservedEventsStdDev**(std::vector<*realtype*> const &observedEventsStdDev, int iz)

set function that copies standard deviation of observed data for specific observable

**Parameters**

- **observedEventsStdDev** – standard deviation of observed data (dimension: nmaxevent)
- **iz** – observed data index

void **setObservedEventsStdDev**(*realtype* stdDev, int iz)

set function that sets all standard deviation of a specific observable to the input value

**Parameters**

- **stdDev** – standard deviation (dimension: scalar)
- **iz** – observed data index

bool **isSetObservedEventsStdDev**(int ie, int iz) const

get function that checks whether standard deviation of even data at specified indices has been set

**Parameters**

- **ie** – event index
- **iz** – event observable index

**Returns** boolean specifying if standard deviation of event data was set

std::vector<*realtype*> const &**getObservedEventsStdDev**() const

get function that copies data from ExpData::observedEventsStdDev to output

**Returns** standard deviation of observed event data

*realtype* const \***getObservedEventsStdDevPtr**(int ie) const

get function that returns a pointer to standard deviation of observed event data at ie-th occurrence

**Parameters** **ie** – event occurrence

**Returns** pointer to standard deviation of observed event data at ie-th occurrence

## Public Members

std::string **id**

Arbitrary (not necessarily unique) identifier.

## Protected Functions

void **applyDimensions**()

resizes observedData, observedDataStdDev, observedEvents and observedEventsStdDev

void **applyDataDimension**()

resizes observedData and observedDataStdDev

void **applyEventDimension**()

resizes observedEvents and observedEventsStdDev

void **checkDataDimension**(std::vector<*realtype*> const &input, char const \*fieldname) const

checker for dimensions of input observedData or observedDataStdDev

### Parameters

- **input** – vector input to be checked
- **fieldname** – name of the input

void **checkEventsDimension**(std::vector<*realtype*> const &input, char const \*fieldname) const

checker for dimensions of input observedEvents or observedEventsStdDev

### Parameters

- **input** – vector input to be checked
- **fieldname** – name of the input

## Protected Attributes

int **nytrue\_** = {0}

number of observables

int **nztrue\_** = {0}

number of event observables

int **nmaxevent\_** = {0}

maximal number of event occurrences

std::vector<*realtype*> **observed\_data\_**

observed data (dimension: nt x nytrue, row-major)

std::vector<*realtype*> **observed\_data\_std\_dev\_**

standard deviation of observed data (dimension: nt x nytrue, row-major)

std::vector<*realtype*> **observed\_events\_**

observed events (dimension: nmaxevents x nztrue, row-major)

std::vector<*realtype*> **observed\_events\_std\_dev\_**

standard deviation of observed events/roots (dimension: nmaxevents x nztrue, row-major)

## Friends

inline friend bool **operator==(***ExpData* const &lhs, *ExpData* const &rhs)

Equality operator.

### Parameters

- **lhs** – some object
- **rhs** – another object

**Returns** true, if both arguments are equal; false otherwise.

## Class FinalStateStorer

- Defined in file\_include\_amici\_forwardproblem.h

## Inheritance Relationships

### Base Type

- public amici::ContextManager (*Class ContextManager*)

## Class Documentation

class **FinalStateStorer** : public amici::ContextManager

stores the stimulation state when it goes out of scope



## Public Functions

inline explicit **FinalStateStorer**(*ForwardProblem* \*fwd)

constructor, attaches problem pointer

**Parameters** *fwd* – problem from which the simulation state is to be stored

*FinalStateStorer* &**operator**=(*FinalStateStorer* const &other) = delete

inline **~FinalStateStorer**()

destructor, stores simulation state

## Class ForwardProblem

- Defined in file\_include\_amici\_forwardproblem.h

## Class Documentation

class **ForwardProblem**

The *ForwardProblem* class groups all functions for solving the forward problem.

## Public Functions

**ForwardProblem**(*ExpData* const \*edata, *Model* \*model, *Solver* \*solver, *SteadystateProblem* const \*preeq)

Constructor.

### Parameters

- **edata** – pointer to *ExpData* instance
- **model** – pointer to *Model* instance
- **solver** – pointer to *Solver* instance
- **preeq** – preequilibration with which to initialize the forward problem, pass nullptr for no initialization

**~ForwardProblem**() = default

void **workForwardProblem**()

Solve the forward problem.

If forward sensitivities are enabled this will also compute sensitivities.

void **getAdjointUpdates**(*Model* &model, *ExpData* const &edata)

computes adjoint updates dJydx according to provided model and expdata

### Parameters

- **model** – *Model* instance
- **edata** – experimental data

inline *realtype* **getTime**() const

Accessor for t.

**Returns** t

inline *AmiVector* const &getState() const

Accessor for x.

**Returns** x

inline *AmiVector* const &getStateDerivative() const

Accessor for dx.

**Returns** dx

inline *AmiVectorArray* const &getStateSensitivity() const

Accessor for sx.

**Returns** sx

inline std::vector<*AmiVector*> const &getStatesAtDiscontinuities() const

Accessor for x\_disc.

**Returns** x\_disc

inline std::vector<*AmiVector*> const &getRHSAtDiscontinuities() const

Accessor for xdot\_disc.

**Returns** xdot\_disc

inline std::vector<*AmiVector*> const &getRHSBeforeDiscontinuities() const

Accessor for xdot\_old\_disc.

**Returns** xdot\_old\_disc

inline std::vector<int> const &getNumberOfRoots() const

Accessor for nroots.

**Returns** nroots

inline std::vector<*realtype*> const &getDiscontinuities() const

Accessor for discs.

**Returns** discs

inline std::vector<std::vector<int>> const &getRootIndexes() const

Accessor for rootidx.

**Returns** rootidx

inline std::vector<*realtype*> const &getDJydx() const

Accessor for dJydx.

**Returns** dJydx

inline std::vector<*realtype*> const &getDJzdx() const

Accessor for dJzdx.

**Returns** dJzdx

inline *AmiVector* \*getStatePointer()

Accessor for pointer to x.

**Returns** &x

inline *AmiVector* \***getStateDerivativePointer**()

Accessor for pointer to dx.

**Returns** &dx

inline *AmiVectorArray* \***getStateSensitivityPointer**()

accessor for pointer to sx

**Returns** &sx

inline *AmiVectorArray* \***getStateDerivativeSensitivityPointer**()

Accessor for pointer to sdx.

**Returns** &sdx

inline int **getCurrentTimeIteration**() const

Accessor for it.

**Returns** it

inline *realtype* **getFinalTime**() const

Returns final time point for which simulations are available.

**Returns** time point

inline int **getEventCounter**() const

Returns maximal event index for which simulations are available.

**Returns** index

inline int **getRootCounter**() const

Returns maximal event index for which the timepoint is available.

**Returns** index

inline *SimulationState* const &**getSimulationStateTimepoint**(int it) const

Retrieves the carbon copy of the simulation state variables at the specified timepoint index.

**Parameters** *it* – timepoint index

**Returns** state

inline *SimulationState* const &**getSimulationStateEvent**(int iroot) const

Retrieves the carbon copy of the simulation state variables at the specified event index.

**Parameters** *iroot* – event index

**Returns** *SimulationState*

inline *SimulationState* const &**getInitialSimulationState**() const

Retrieves the carbon copy of the simulation state variables at the initial timepoint.

**Returns** *SimulationState*

inline *SimulationState* const &**getFinalSimulationState**() const

Retrieves the carbon copy of the simulation state variables at the final timepoint (or when simulation failed)

**Returns** *SimulationState*

## Public Members

*Model* \***model**

pointer to model instance

*Solver* \***solver**

pointer to solver instance

*ExpData* const \***edata**

pointer to experimental data instance

## Class HermiteSpline

- Defined in file\_include\_amici\_splinefunctions.h

## Inheritance Relationships

### Base Type

- public amici::AbstractSpline (*Class AbstractSpline*)

## Class Documentation

class **HermiteSpline** : public amici::AbstractSpline

AMICI Hermite spline class.

Instances of this class represent Hermite splines, which are uniquely determined by their nodes, the values at their nodes, the derivatives at their nodes (defaulting to finite difference approximations from the node values), boundary conditions and extrapolation conditions. Optionally, the spline can be defined in log-space in order to ensure positivity.

## Public Functions

**HermiteSpline**() = default

**HermiteSpline**(std::vector<*realtype*> nodes, std::vector<*realtype*> node\_values, std::vector<*realtype*> node\_values\_derivative, *SplineBoundaryCondition* firstNodeBC, *SplineBoundaryCondition* lastNodeBC, *SplineExtrapolation* firstNodeExtrapol, *SplineExtrapolation* lastNodeExtrapol, bool node\_derivative\_by\_FD, bool equidistant\_spacing, bool logarithmic\_parametrization)

Construct a *HermiteSpline*.

### Parameters

- **nodes** – the nodes defining the position at which the value of the spline is known (if equidistant\_spacing is true, it must contain only the first and the last node; the other nodes will be automatically inserted, assuming they are uniformly spaced)
- **node\_values** – the values assumed by the spline at the nodes

- **node\_values\_derivative** – the derivatives of the spline at the nodes (if `node_derivative_by_FD` is true, it will be resized and filled with finite difference approximations computed from `node_values`)
- **firstNodeBC** – boundary condition at the first node
- **lastNodeBC** – boundary condition at the last node
- **firstNodeExtrapol** – extrapolation method on the left side
- **lastNodeExtrapol** – extrapolation method on the right side
- **node\_derivative\_by\_FD** – whether derivatives are to be computed by finite differences
- **equidistant\_spacing** – whether equidistant nodes are to be computed
- **logarithmic\_parametrization** – if true, the spline interpolation will occur in log-space in order to ensure positivity of the interpolant (which strictly speaking will no longer be a spline)

virtual void **compute\_coefficients**() override

Compute the coefficients for all polynomial segments of this spline.

virtual void **compute\_coefficients\_sensi**(int nplist, int spline\_offset, gsl::span<realtype> dvaluesdp, gsl::span<realtype> dslopesdp) override

Compute the coefficients for all polynomial segments of the derivatives of this spline with respect to the parameters.

---

#### Remark

The contents of `dvaluesdp` and `dslopesdp` may be modified by this function.

---

#### Parameters

- **nplist** – number of parameters
- **spline\_offset** – offset of this spline inside `dvaluesdp` and `dslopesdp`
- **dvaluesdp** – derivatives of the spline values with respect to the parameters (for all splines in the model, not just this one)
- **dslopesdp** – derivatives of the spline derivatives with respect to the parameters (for all splines in the model, not just this one)

virtual void **compute\_final\_value**() override

Compute the limit value of the spline as the evaluation point tends to positive infinity.

virtual void **compute\_final\_sensitivity**(int nplist, int spline\_offset, gsl::span<realtype> dspline\_valuesdp, gsl::span<realtype> dspline\_slopesdp) override

Compute the limit of the value of the sensitivity as the evaluation point tends to positive infinity.

#### Parameters

- **nplist** – number of parameters
- **spline\_offset** – offset of this spline inside `dspline_valuesdp` and `dspline_slopesdp`

- **dspline\_valuesdp** – derivatives of the spline values with respect to the parameters (for all splines in the model, not just this one)
- **dspline\_slopesdp** – derivatives of the spline derivatives with respect to the parameters (for all splines in the model, not just this one)

virtual *realtype* **get\_value\_scaled**(const *realtype* t) const override

Get the value of this spline at a given point in the scale in which interpolation is carried out (e.g., log-scale)

**Parameters** **t** – point at which the spline is to be evaluated

**Returns** scaled value of the spline at **t**

*realtype* **get\_node\_derivative**(int const i) const

Get the derivative of the spline at a given node.

**Parameters** **i** – index of the node at which the spline is to be evaluated

**Returns** value of the derivative at the **i**-th node

*realtype* **get\_node\_derivative\_scaled**(int const i) const

Get the derivative of the spline at a given node in the scale in which interpolation is carried out (e.g., log-scale)

**Parameters** **i** – index of the node at which the spline is to be evaluated

**Returns** scaled value of the derivative at the **i**-th node

virtual *realtype* **get\_sensitivity\_scaled**(const *realtype* t, int const ip) const override

Get the derivative of this spline with respect to a given parameter at a given point in the scale in which interpolation is carried out (e.g., log-scale)

**Parameters**

- **t** – point at which the sensitivity is to be evaluated
- **ip** – index of the parameter

**Returns** scaled sensitivity of the spline with respect to the **ip**th parameter at **t**

inline bool **get\_node\_derivative\_by\_fd**() const

Whether derivatives of this spline are computed by finite differences.

**Returns** boolean flag

## Class IDAException

- Defined in file\_include\_amici\_exception.h

## Inheritance Relationships

### Base Type

- public amici::AmiException (*Class AmiException*)

## Class Documentation

class **IDAException** : public amici::*AmiException*  
 ida exception handler class

### Public Functions

**IDAException**(int error\_code, char const \*function)  
 Constructor.

#### Parameters

- **error\_code** – error code returned by ida function
- **function** – ida function name

## Class IDASolver

- Defined in file\_include\_amici\_solver\_idas.h

## Inheritance Relationships

### Base Type

- public amici::Solver (*Class Solver*)

## Class Documentation

class **IDASolver** : public amici::*Solver*  
 The *IDASolver* class is a wrapper around the SUNDIALS IDAS solver.

### Public Functions

**~IDASolver**() override = default

virtual *Solver* \***clone**() const override  
 Clone this instance.

**Returns** The clone

virtual void **reInitPostProcessF**(*realtype* tnext) const override  
 reInitPostProcessF postprocessing of the solver memory after a discontinuity in the forward problem

**Parameters** **tnext** – next timepoint (defines integration direction)

virtual void **reInitPostProcessB**(*realtype* tnext) const override  
 reInitPostProcessB postprocessing of the solver memory after a discontinuity in the backward problem

**Parameters** **tnext** – next timepoint (defines integration direction)

virtual void **reInit**(*realtype* t0, *AmiVector* const &yy0, *AmiVector* const &yp0) const override

Reinitializes the states in the solver after an event occurrence.

**Parameters**

- **t0** – reinitialization timepoint
- **yy0** – initial state variables
- **yp0** – initial derivative state variables (DAE only)

virtual void **senseReInit**(*AmiVectorArray* const &yyS0, *AmiVectorArray* const &ypS0) const override

Reinitializes the state sensitivities in the solver after an event occurrence.

**Parameters**

- **yyS0** – new state sensitivity
- **ypS0** – new derivative state sensitivities (DAE only)

virtual void **sensToggleOff**() const override

Switches off computation of state sensitivities without deallocating the memory for sensitivities.

virtual void **reInitB**(int which, *realtype* tB0, *AmiVector* const &yyB0, *AmiVector* const &ypB0) const override

Reinitializes the adjoint states after an event occurrence.

**Parameters**

- **which** – identifier of the backwards problem
- **tB0** – reinitialization timepoint
- **yyB0** – new adjoint state
- **ypB0** – new adjoint derivative state

virtual void **quadReInitB**(int which, *AmiVector* const &yQB0) const override

Reinitialize the adjoint states after an event occurrence.

**Parameters**

- **which** – identifier of the backwards problem
- **yQB0** – new adjoint quadrature state

virtual void **quadSStolerancesB**(int which, *realtype* reltolQB, *realtype* abstolQB) const override

sets relative and absolute tolerances for the quadrature backward problem

**Parameters**

- **which** – identifier of the backwards problem
- **reltolQB** – relative tolerances
- **abstolQB** – absolute tolerances

virtual void **quadSStolerances**(*realtype* reltolQ, *realtype* abstolQ) const override

sets relative and absolute tolerances for the quadrature problem

**Parameters**

- **reltolQB** – relative tolerances
- **abstolQB** – absolute tolerances



virtual int **solve**(*realtype* tout, int itask) const override

Solves the forward problem until a predefined timepoint.

**Parameters**

- **tout** – timepoint until which simulation should be performed
- **itask** – task identifier, can be CV\_NORMAL or CV\_ONE\_STEP

**Returns** status flag indicating success of execution

virtual int **solveF**(*realtype* tout, int itask, int \*ncheckPtr) const override

Solves the forward problem until a predefined timepoint (adjoint only)

**Parameters**

- **tout** – timepoint until which simulation should be performed
- **itask** – task identifier, can be CV\_NORMAL or CV\_ONE\_STEP
- **ncheckPtr** – pointer to a number that counts the internal checkpoints

**Returns** status flag indicating success of execution

virtual void **solveB**(*realtype* tBout, int itaskB) const override

Solves the backward problem until a predefined timepoint (adjoint only)

**Parameters**

- **tBout** – timepoint until which simulation should be performed
- **itaskB** – task identifier, can be CV\_NORMAL or CV\_ONE\_STEP

virtual void **getRootInfo**(int \*rootsfound) const override

getRootInfo extracts information which event occurred

**Parameters** **rootsfound** – array with flags indicating whether the respective event occurred

virtual void **getDky**(*realtype* t, int k) const override

interpolates the (derivative of the) solution at the requested timepoint

**Parameters**

- **t** – timepoint
- **k** – derivative order

virtual void **getSens**() const override

extracts the state sensitivity at the current timepoint from solver memory and writes it to the sx member variable

virtual void **getSensDky**(*realtype* t, int k) const override

interpolates the (derivative of the) solution at the requested timepoint

**Parameters**

- **t** – timepoint
- **k** – derivative order

virtual void **getB**(int which) const override

extracts the adjoint state at the current timepoint from solver memory and writes it to the xB member variable

**Parameters** **which** – index of the backwards problem

virtual void **getDkyB**(*realtype* t, int k, int which) const override  
interpolates the (derivative of the) solution at the requested timepoint

**Parameters**

- **t** – timepoint
- **k** – derivative order
- **which** – index of backward problem

virtual void **getQuadB**(int which) const override  
extracts the adjoint quadrature state at the current timepoint from solver memory and writes it to the xQB member variable

**Parameters** **which** – index of the backwards problem

virtual void **getQuadDkyB**(*realtype* t, int k, int which) const override  
interpolates the (derivative of the) solution at the requested timepoint

**Parameters**

- **t** – timepoint
- **k** – derivative order
- **which** – index of backward problem

virtual void **getQuad**(*realtype* &t) const override  
extracts the quadrature at the current timepoint from solver memory and writes it to the xQ member variable

**Parameters** **t** – timepoint for quadrature extraction

virtual void **getQuadDky**(*realtype* t, int k) const override  
interpolates the (derivative of the) solution at the requested timepoint

**Parameters**

- **t** – timepoint
- **k** – derivative order

virtual void **calcIC**(*realtype* tout1) const override  
Calculates consistent initial conditions, assumes initial states to be correct (DAE only)

**Parameters** **tout1** – next timepoint to be computed (sets timescale)

virtual void **calcICB**(int which, *realtype* tout1) const override  
Calculates consistent initial conditions for the backwards problem, assumes initial states to be correct (DAE only)

**Parameters**

- **which** – identifier of the backwards problem
- **tout1** – next timepoint to be computed (sets timescale)

virtual void **setStopTime**(*realtype* tstop) const override  
Sets a timepoint at which the simulation will be stopped.

**Parameters** **tstop** – timepoint until which simulation should be performed

virtual void **turnOffRootFinding**() const override  
Disable rootfinding.

virtual *Model* const \***getModel()** const override

Accessor function to the model stored in the user data

**Returns** user data model

virtual void **setLinearSolver()** const override

Sets the linear solver for the forward problem.

virtual void **setLinearSolverB**(int which) const override

Sets the linear solver for the backward problem.

**Parameters** **which** – index of the backward problem

virtual void **setNonLinearSolver()** const override

Set the non-linear solver for the forward problem.

virtual void **setNonLinearSolverSens()** const override

Set the non-linear solver for sensitivities.

virtual void **setNonLinearSolverB**(int which) const override

Set the non-linear solver for the backward problem.

**Parameters** **which** – index of the backward problem

**Solver()** = default

Default constructor.

**Solver**(*Solver* const &other)

*Solver* copy constructor.

**Parameters** **other** –

## Protected Functions

void **reInitPostProcess**(void \*ida\_mem, *realtype* \*t, *AmiVector* \*yout, *AmiVector* \*ypout, *realtype* tout)  
const

Postprocessing of the solver memory after a discontinuity.

**Parameters**

- **ida\_mem** – pointer to IDAS solver memory object
- **t** – pointer to integration time
- **yout** – new state vector
- **ypout** – new state derivative vector
- **tout** – anticipated next integration timepoint.

virtual void **allocateSolver**() const override

Create specifies solver method and initializes solver memory for the forward problem.

virtual void **setSStolerances**(*realtype* rtol, *realtype* atol) const override

sets scalar relative and absolute tolerances for the forward problem

**Parameters**

- **rtol** – relative tolerances
- **atol** – absolute tolerances

virtual void **setSensSStolerances**(*realtype* rtol, *realtype* const \*atol) const override  
activates sets scalar relative and absolute tolerances for the sensitivity variables

**Parameters**

- **rtol** – relative tolerances
- **atol** – array of absolute tolerances for every sensitivity variable

virtual void **setSensErrCon**(bool error\_corr) const override

SetSensErrCon specifies whether error control is also enforced for sensitivities for the forward problem

**Parameters** **error\_corr** – activation flag

virtual void **setQuadErrConB**(int which, bool flag) const override

Specifies whether error control is also enforced for the backward quadrature problem.

**Parameters**

- **which** – identifier of the backwards problem
- **flag** – activation flag

virtual void **setQuadErrCon**(bool flag) const override

Specifies whether error control is also enforced for the forward quadrature problem.

**Parameters** **flag** – activation flag

virtual void **setErrorHandlerFn**() const override

Attaches the error handler function (errMsgIdAndTxt) to the solver.

virtual void **setUserData**() const override

Attaches the user data to the forward problem.

virtual void **setUserDataB**(int which) const override

attaches the user data to the backward problem

**Parameters** **which** – identifier of the backwards problem

virtual void **setMaxNumSteps**(long int mxsteps) const override

specifies the maximum number of steps for the forward problem

---

**Note:** in contrast to the SUNDIALS method, this sets the overall maximum, not the maximum between output times.

---

**Parameters** **mxsteps** – number of steps

virtual void **setStabLimDet**(int stldet) const override

activates stability limit detection for the forward problem

**Parameters** **stldet** – flag for stability limit detection (TRUE or FALSE)

virtual void **setStabLimDetB**(int which, int stldet) const override

activates stability limit detection for the backward problem

**Parameters**

- **which** – identifier of the backwards problem
- **stldet** – flag for stability limit detection (TRUE or FALSE)

virtual void **setId**(*Model* const \*model) const override  
specify algebraic/differential components (DAE only)

**Parameters** **model** – model specification

virtual void **setSuppressAlg**(bool flag) const override  
deactivates error control for algebraic components (DAE only)

**Parameters** **flag** – deactivation flag

void **resetState**(void \*ida\_mem, *const\_N\_Vector* yy0, *const\_N\_Vector* yp0) const  
resetState reset the IDAS solver to restart integration after a rhs discontinuity.

**Parameters**

- **ida\_mem** – pointer to IDAS solver memory object
- **yy0** – new state vector
- **yp0** – new state derivative vector

virtual void **setSensParams**(*realtype* const \*p, *realtype* const \*pbar, int const \*plist) const override  
specifies the scaling and indexes for sensitivity computation

**Parameters**

- **p** – parameters
- **pbar** – parameter scaling constants
- **plist** – parameter index list

virtual void **adjInit**() const override  
initializes the adjoint problem

virtual void **quadInit**(*AmiVector* const &xQ0) const override  
initializes the quadratures

**Parameters** **xQ0** – vector with initial values for xQ

virtual void **allocateSolverB**(int \*which) const override  
Specifies solver method and initializes solver memory for the backward problem.

**Parameters** **which** – identifier of the backwards problem

virtual void **setMaxNumStepsB**(int which, long int mxstepsB) const override  
specifies the maximum number of steps for the forward problem

---

**Note:** in contrast to the SUNDIALS method, this sets the overall maximum, not the maximum between output times.

---

**Parameters**

- **which** – identifier of the backwards problem
- **mxstepsB** – number of steps

virtual void **setSStolerancesB**(int which, *realtype* relTolB, *realtype* absTolB) const override  
sets relative and absolute tolerances for the backward problem

**Parameters**

- **which** – identifier of the backwards problem
- **relTolB** – relative tolerances
- **absTolB** – absolute tolerances

virtual void **diag**() const override

attaches a diagonal linear solver to the forward problem

virtual void **diagB**(int which) const override

attaches a diagonal linear solver to the backward problem

**Parameters** **which** – identifier of the backwards problem

virtual void **getNumSteps**(void const \*ami\_mem, long int \*numsteps) const override

reports the number of solver steps

**Parameters**

- **ami\_mem** – pointer to the solver memory instance (can be from forward or backward problem)
- **numsteps** – output array

virtual void **getNumRhsEvals**(void const \*ami\_mem, long int \*numrhsevals) const override

reports the number of right hand evaluations

**Parameters**

- **ami\_mem** – pointer to the solver memory instance (can be from forward or backward problem)
- **numrhsevals** – output array

virtual void **getNumErrTestFails**(void const \*ami\_mem, long int \*numerrtestfails) const override

reports the number of local error test failures

**Parameters**

- **ami\_mem** – pointer to the solver memory instance (can be from forward or backward problem)
- **numerrtestfails** – output array

virtual void **getNumNonlinSolvConvFails**(void const \*ami\_mem, long int \*numnonlinsolvconvfails) const override

reports the number of nonlinear convergence failures

**Parameters**

- **ami\_mem** – pointer to the solver memory instance (can be from forward or backward problem)
- **numnonlinsolvconvfails** – output array

virtual void **getLastOrder**(void const \*ami\_mem, int \*order) const override

Reports the order of the integration method during the last internal step.

**Parameters**

- **ami\_mem** – pointer to the solver memory instance (can be from forward or backward problem)
- **order** – output array

virtual void **\*getAdjBmem**(void \*ami\_mem, int which) const override

Retrieves the solver memory instance for the backward problem.

**Parameters**

- **which** – identifier of the backwards problem
- **ami\_mem** – pointer to the forward solver memory instance

**Returns** A (void \*) pointer to the CVODES memory allocated for the backward problem.

virtual void **init**(*realtype* t0, *AmiVector* const &x0, *AmiVector* const &dx0) const override

Initializes the states at the specified initial timepoint.

**Parameters**

- **t0** – initial timepoint
- **x0** – initial states
- **dx0** – initial derivative states

virtual void **initSteadystate**(const *realtype* t0, *AmiVector* const &x0, *AmiVector* const &dx0) const override

Initializes the states at the specified initial timepoint.

**Parameters**

- **t0** – initial timepoint
- **x0** – initial states
- **dx0** – initial derivative states

virtual void **sensInit1**(*AmiVectorArray* const &sx0, *AmiVectorArray* const &sdx0) const override

Initializes the forward sensitivities.

**Parameters**

- **sx0** – initial states sensitivities
- **sdx0** – initial derivative states sensitivities

virtual void **binit**(int which, *realtype* tf, *AmiVector* const &xB0, *AmiVector* const &dxB0) const override

Initialize the adjoint states at the specified final timepoint.

**Parameters**

- **which** – identifier of the backwards problem
- **tf** – final timepoint
- **xB0** – initial adjoint state
- **dxB0** – initial adjoint derivative state

virtual void **qbinit**(int which, *AmiVector* const &xB0) const override

Initialize the quadrature states at the specified final timepoint.

**Parameters**

- **which** – identifier of the backwards problem
- **xB0** – initial adjoint quadrature state

virtual void **rootInit**(int ne) const override

Initializes the rootfinding for events.

**Parameters** **ne** – number of different events

virtual void **setDenseJacFn**() const override

Set the dense Jacobian function.

virtual void **setSparseJacFn**() const override

sets the sparse Jacobian function

virtual void **setBandJacFn**() const override

sets the banded Jacobian function

virtual void **setJacTimesVecFn**() const override

sets the Jacobian vector multiplication function

virtual void **setDenseJacFnB**(int which) const override

sets the dense Jacobian function

**Parameters** **which** – identifier of the backwards problem

virtual void **setSparseJacFnB**(int which) const override

sets the sparse Jacobian function

**Parameters** **which** – identifier of the backwards problem

virtual void **setBandJacFnB**(int which) const override

sets the banded Jacobian function

**Parameters** **which** – identifier of the backwards problem

virtual void **setJacTimesVecFnB**(int which) const override

sets the Jacobian vector multiplication function

**Parameters** **which** – identifier of the backwards problem

virtual void **setSparseJacFn\_ss**() const override

sets the sparse Jacobian function for backward steady state case

## Class IntegrationFailure

- Defined in file\_include\_amici\_exception.h

## Inheritance Relationships

### Base Type

- public amici::AmiException (*Class AmiException*)



## Class Documentation

class **IntegrationFailure** : public amici::AmiException

Integration failure exception for the forward problem.

This exception should be thrown when an integration failure occurred for this exception we can assume that we can recover from the exception and return a solution struct to the user

### Public Functions

**IntegrationFailure**(int code, *realtype* t)

Constructor.

#### Parameters

- **code** – error code returned by ccode/ida
- **t** – time of integration failure

### Public Members

int **error\_code**

error code returned by ccodes/idas

*realtype* **time**

time of integration failure

## Class IntegrationFailureB

- Defined in file\_include\_amici\_exception.h

## Inheritance Relationships

### Base Type

- public amici::AmiException (*Class AmiException*)

## Class Documentation

class **IntegrationFailureB** : public amici::AmiException

Integration failure exception for the backward problem.

This exception should be thrown when an integration failure occurred for this exception we can assume that we can recover from the exception and return a solution struct to the user

## Public Functions

**IntegrationFailureB**(int code, *realtype* t)

Constructor.

### Parameters

- **code** – error code returned by ccode/ida
- **t** – time of integration failure

## Public Members

int **error\_code**

error code returned by ccode/ida

*realtype* **time**

time of integration failure

## Class Logger

- Defined in file\_include\_amici\_logging.h

## Class Documentation

class **Logger**

A logger, holding a list of error messages.

## Public Functions

**Logger**() = default

void **log**(*LogSeverity* severity, std::string const &identifier, std::string const &message)

Add a log entry.

### Parameters

- **severity** – Severity level
- **identifier** – Short identifier for the logged event
- **message** – A more detailed message

void **log**(*LogSeverity* severity, std::string const &identifier, char const \*format, ...)

Add a log entry with printf-like message formatting.

### Parameters

- **severity** – Severity level
- **identifier** – Short identifier for the logged event
- **format** – printf format string

## Public Members

`std::vector<LogItem> items`

The log items

## Class Model

- Defined in file\_include\_amici\_model.h

## Inheritance Relationships

### Base Types

- `public amici::AbstractModel` (*Class AbstractModel*)
- `public amici::ModelDimensions` (*Struct ModelDimensions*)

### Derived Types

- `public amici::Model_DAE` (*Class Model\_DAE*)
- `public amici::Model_ODE` (*Class Model\_ODE*)

## Class Documentation

class **Model** : public amici::AbstractModel, public amici::ModelDimensions

The *Model* class represents an AMICI ODE/DAE model.

The model can compute various model related quantities based on symbolically generated code.

Subclassed by *amici::Model\_DAE*, *amici::Model\_ODE*

### Public Functions

**Model**() = default

Default constructor

**Model**(*ModelDimensions* const &model\_dimensions, *SimulationParameters* simulation\_parameters, amici::SecondOrderMode o2mode, std::vector<amici::realtype> idlist, std::vector<int> z2event, bool pythonGenerated = false, int ndxdotdp\_explicit = 0, int ndxdotdx\_explicit = 0, int w\_recursion\_depth = 0)

Constructor with model dimensions.

#### Parameters

- **model\_dimensions** – *Model* dimensions
- **simulation\_parameters** – Simulation parameters
- **o2mode** – Second order sensitivity mode

- **idlist** – Indexes indicating algebraic components (DAE only)
- **z2event** – Mapping of event outputs to events
- **pythonGenerated** – Flag indicating matlab or python wrapping
- **ndxdotdp\_explicit** – Number of nonzero elements in `ndxdotdp_explicit`
- **ndxdotdx\_explicit** – Number of nonzero elements in `ndxdotdx_explicit`
- **w\_recursion\_depth** – Recursion depth of fw

`~Model()` override = default

Destructor.

`Model &operator=(Model const &other)` = delete

Copy assignment is disabled until const members are removed.

**Parameters** **other** – Object to copy from

**Returns**

virtual `Model *clone()` const = 0

Clone this instance.

**Returns** The clone

void **initialize**(*AmiVector* &x, *AmiVector* &dx, *AmiVectorArray* &sx, *AmiVectorArray* &sdx, bool computeSensitivities, std::vector<int> &roots\_found)

Initialize model properties.

**Parameters**

- **x** – Reference to state variables
- **dx** – Reference to time derivative of states (DAE only)
- **sx** – Reference to state variable sensitivities
- **sdx** – Reference to time derivative of state sensitivities (DAE only)
- **computeSensitivities** – Flag indicating whether sensitivities are to be computed
- **roots\_found** – boolean indicators indicating whether roots were found at t0 by this fun

void **initializeB**(*AmiVector* &xB, *AmiVector* &dxB, *AmiVector* &xQB, bool posteq) const

Initialize model properties.

**Parameters**

- **xB** – Adjoint state variables
- **dxB** – Time derivative of adjoint states (DAE only)
- **xQB** – Adjoint quadratures
- **posteq** – Flag indicating whether postequilibration was performed

void **initializeStates**(*AmiVector* &x)

Initialize initial states.

**Parameters** **x** – State vector to be initialized

void **initializeStateSensitivities**(*AmiVectorArray* &sx, *AmiVector* const &x)

Initialize initial state sensitivities.

**Parameters**

- **sx** – Reference to state variable sensitivities
- **x** – Reference to state variables

void **initializeSplines**()

Initialization of spline functions.

void **initializeSplineSensitivities**()

Initialization of spline sensitivity functions.

void **initEvents**(*AmiVector* const &x, *AmiVector* const &dx, std::vector<int> &roots\_found)

Initialize the Heaviside variables **h** at the initial time **t0**.

Heaviside variables activate/deactivate on event occurrences.

**Parameters**

- **x** – Reference to state variables
- **dx** – Reference to time derivative of states (DAE only)
- **roots\_found** – boolean indicators indicating whether roots were found at t0 by this fun

int **nplist**() const

Get number of parameters wrt to which sensitivities are computed.

**Returns** Length of sensitivity index vector

int **np**() const

Get total number of model parameters.

**Returns** Length of parameter vector

int **nk**() const

Get number of constants.

**Returns** Length of constant vector

int **ncl**() const

Get number of conservation laws.

**Returns** Number of conservation laws (i.e., difference between **nx\_rdata** and **nx\_solver**).

int **nx\_reinit**() const

Get number of solver states subject to reinitialization.

**Returns** *Model* member **nx\_solver\_reinit**

double const \***k**() const

Get fixed parameters.

**Returns** Pointer to constants array

int **nMaxEvent**() const

Get maximum number of events that may occur for each type.

**Returns** Maximum number of events that may occur for each type

void **setMaxEvent**(int nmaxevent)

Set maximum number of events that may occur for each type.

**Parameters** **nmaxevent** – Maximum number of events that may occur for each type

int **nt**() const

Get number of timepoints.

**Returns** Number of timepoints

std::vector<*ParameterScaling*> const &**getParameterScale**() const

Get parameter scale for each parameter.

**Returns** Vector of parameter scales

void **setParameterScale**(*ParameterScaling* pscale)

Set parameter scale for each parameter.

NOTE: Resets initial state sensitivities.

**Parameters** **pscale** – Scalar parameter scale to be set for all parameters

void **setParameterScale**(std::vector<*ParameterScaling*> const &pscaleVec)

Set parameter scale for each parameter.

NOTE: Resets initial state sensitivities.

**Parameters** **pscaleVec** – Vector of parameter scales

std::vector<*realtype*> const &**getUnscaledParameters**() const

Get parameters with transformation according to parameter scale applied.

**Returns** Unscaled parameters

std::vector<*realtype*> const &**getParameters**() const

Get parameter vector.

**Returns** The user-set parameters (see also *Model::getUnscaledParameters*)

*realtype* **getParameterById**(std::string const &par\_id) const

Get value of first model parameter with the specified ID.

**Parameters** **par\_id** – Parameter ID

**Returns** Parameter value

*realtype* **getParameterByName**(std::string const &par\_name) const

Get value of first model parameter with the specified name.

**Parameters** **par\_name** – Parameter name

**Returns** Parameter value

void **setParameters**(std::vector<*realtype*> const &p)

Set the parameter vector.

**Parameters** **p** – Vector of parameters

void **setParameterById**(std::map<std::string, *realtype*> const &p, bool ignoreErrors = false)

Set model parameters according to the parameter IDs and mapped values.

**Parameters**

- **p** – Map of parameters IDs and values

- **ignoreErrors** – Ignore errors such as parameter IDs in *p* which are not model parameters

void **setParameterById**(std::string const &par\_id, *realtype* value)

Set value of first model parameter with the specified ID.

#### Parameters

- **par\_id** – Parameter ID
- **value** – Parameter value

int **setParametersByIdRegex**(std::string const &par\_id\_regex, *realtype* value)

Set all values of model parameters with IDs matching the specified regular expression.

#### Parameters

- **par\_id\_regex** – Parameter ID regex
- **value** – Parameter value

**Returns** Number of parameter IDs that matched the regex

void **setParameterByName**(std::string const &par\_name, *realtype* value)

Set value of first model parameter with the specified name.

#### Parameters

- **par\_name** – Parameter name
- **value** – Parameter value

void **setParameterByName**(std::map<std::string, *realtype*> const &p, bool ignoreErrors = false)

Set model parameters according to the parameter name and mapped values.

#### Parameters

- **p** – Map of parameters names and values
- **ignoreErrors** – Ignore errors such as parameter names in *p* which are not model parameters

int **setParametersByNameRegex**(std::string const &par\_name\_regex, *realtype* value)

Set all values of all model parameters with names matching the specified regex.

#### Parameters

- **par\_name\_regex** – Parameter name regex
- **value** – Parameter value

**Returns** Number of fixed parameter names that matched the regex

std::vector<*realtype*> const &**getFixedParameters**() const

Get values of fixed parameters.

**Returns** Vector of fixed parameters with same ordering as in *Model::getFixedParameterIds*

*realtype* **getFixedParameterById**(std::string const &par\_id) const

Get value of fixed parameter with the specified ID.

**Parameters** **par\_id** – Parameter ID

**Returns** Parameter value

*realtype* **getFixedParameterByName**(std::string const &par\_name) const

Get value of fixed parameter with the specified name.

If multiple parameters have the same name, the first parameter with matching name is returned.

**Parameters** **par\_name** – Parameter name

**Returns** Parameter value

void **setFixedParameters**(std::vector<*realtype*> const &k)

Set values for constants.

**Parameters** **k** – Vector of fixed parameters

void **setFixedParameterById**(std::string const &par\_id, *realtype* value)

Set value of first fixed parameter with the specified ID.

**Parameters**

- **par\_id** – Fixed parameter id
- **value** – Fixed parameter value

int **setFixedParametersByIdRegex**(std::string const &par\_id\_regex, *realtype* value)

Set values of all fixed parameters with the ID matching the specified regex.

**Parameters**

- **par\_id\_regex** – Fixed parameter name regex
- **value** – Fixed parameter value

**Returns** Number of fixed parameter IDs that matched the regex

void **setFixedParameterByName**(std::string const &par\_name, *realtype* value)

Set value of first fixed parameter with the specified name.

**Parameters**

- **par\_name** – Fixed parameter ID
- **value** – Fixed parameter value

int **setFixedParametersByNameRegex**(std::string const &par\_name\_regex, *realtype* value)

Set value of all fixed parameters with name matching the specified regex.

**Parameters**

- **par\_name\_regex** – Fixed parameter name regex
- **value** – Fixed parameter value

**Returns** Number of fixed parameter names that matched the regex

virtual std::string **getName**() const

Get the model name.

**Returns** *Model* name

virtual bool **hasParameterNames**() const

Report whether the model has parameter names set.

**Returns** Boolean indicating whether parameter names were set. Also returns true if the number of corresponding variables is just zero.



virtual std::vector<std::string> **getParameterNames()** const

Get names of the model parameters.

**Returns** The parameter names

virtual bool **hasStateNames()** const

Report whether the model has state names set.

**Returns** Boolean indicating whether state names were set. Also returns `true` if the number of corresponding variables is just zero.

virtual std::vector<std::string> **getStateNames()** const

Get names of the model states.

**Returns** State names

virtual std::vector<std::string> **getStateNamesSolver()** const

Get names of the solver states.

**Returns** State names

virtual bool **hasFixedParameterNames()** const

Report whether the model has fixed parameter names set.

**Returns** Boolean indicating whether fixed parameter names were set. Also returns `true` if the number of corresponding variables is just zero.

virtual std::vector<std::string> **getFixedParameterNames()** const

Get names of the fixed model parameters.

**Returns** Fixed parameter names

virtual bool **hasObservableNames()** const

Report whether the model has observable names set.

**Returns** Boolean indicating whether observable names were set. Also returns `true` if the number of corresponding variables is just zero.

virtual std::vector<std::string> **getObservableNames()** const

Get names of the observables.

**Returns** Observable names

virtual bool **hasExpressionNames()** const

Report whether the model has expression names set.

**Returns** Boolean indicating whether expression names were set. Also returns `true` if the number of corresponding variables is just zero.

virtual std::vector<std::string> **getExpressionNames()** const

Get names of the expressions.

**Returns** Expression names

virtual bool **hasParameterIds()** const

Report whether the model has parameter IDs set.

**Returns** Boolean indicating whether parameter IDs were set. Also returns `true` if the number of corresponding variables is just zero.

virtual std::vector<std::string> **getParameterIds()** const

Get IDs of the model parameters.

**Returns** Parameter IDs

virtual bool **hasStateIds()** const

Report whether the model has state IDs set.

**Returns** Boolean indicating whether state IDs were set. Also returns `true` if the number of corresponding variables is just zero.

virtual std::vector<std::string> **getStateIds()** const

Get IDs of the model states.

**Returns** State IDs

virtual std::vector<std::string> **getStateIdsSolver()** const

Get IDs of the solver states.

**Returns** State IDs

virtual bool **hasFixedParameterIds()** const

Report whether the model has fixed parameter IDs set.

**Returns** Boolean indicating whether fixed parameter IDs were set. Also returns `true` if the number of corresponding variables is just zero.

virtual std::vector<std::string> **getFixedParameterIds()** const

Get IDs of the fixed model parameters.

**Returns** Fixed parameter IDs

virtual bool **hasObservableIds()** const

Report whether the model has observable IDs set.

**Returns** Boolean indicating whether observable ids were set. Also returns `true` if the number of corresponding variables is just zero.

virtual std::vector<std::string> **getObservableIds()** const

Get IDs of the observables.

**Returns** Observable IDs

virtual bool **hasExpressionIds()** const

Report whether the model has expression IDs set.

**Returns** Boolean indicating whether expression ids were set. Also returns `true` if the number of corresponding variables is just zero.

virtual std::vector<std::string> **getExpressionIds()** const

Get IDs of the expression.

**Returns** Expression IDs

virtual bool **hasQuadraticLLH()** const

Checks whether the defined noise model is gaussian, i.e., the nllh is quadratic.

**Returns** boolean flag

```
std::vector<realtype> const &getTimepoints() const
```

Get the timepoint vector.

**Returns** Timepoint vector

```
realtype getTimepoint(int it) const
```

Get simulation timepoint for time index *it*.

**Parameters** *it* – Time index

**Returns** Timepoint

```
void setTimepoints(std::vector<realtype> const &ts)
```

Set the timepoint vector.

**Parameters** *ts* – New timepoint vector

```
double t0() const
```

Get simulation start time.

**Returns** Simulation start time

```
void setT0(double t0)
```

Set simulation start time.

**Parameters** *t0* – Simulation start time

```
std::vector<bool> const &getStateIsNonNegative() const
```

Get flags indicating whether states should be treated as non-negative.

**Returns** Vector of flags

```
void setStateIsNonNegative(std::vector<bool> const &stateIsNonNegative)
```

Set flags indicating whether states should be treated as non-negative.

**Parameters** *stateIsNonNegative* – Vector of flags

```
void setAllStatesNonNegative()
```

Set flags indicating that all states should be treated as non-negative.

```
inline ModelState const &getModelState() const
```

Get the current model state.

**Returns** Current model state

```
inline void setModelState(ModelState const &state)
```

Set the current model state.

**Parameters** *state* – *Model* state

```
inline void setMinimumSigmaResiduals(double min_sigma)
```

Sets the estimated lower boundary for *sigma\_y*. When `:meth:setAddSigmaResiduals` is activated, this lower boundary must ensure that  $\log(\text{sigma}) + \text{min\_sigma} > 0$ .

**Parameters** *min\_sigma* – lower boundary

```
inline realtype getMinimumSigmaResiduals() const
```

Gets the specified estimated lower boundary for *sigma\_y*.

**Returns** lower boundary

inline void **setAddSigmaResiduals**(bool sigma\_res)

Specifies whether residuals should be added to account for parameter dependent sigma.

If set to true, additional residuals of the form  $\sqrt{\log(\sigma) + C}$  will be added. This enables least-squares optimization for variables with Gaussian noise assumption and parameter dependent standard deviation sigma. The constant  $C$  can be set via :meth:setMinimumSigmaResiduals.

**Parameters** **sigma\_res** – if true, additional residuals are added

inline bool **getAddSigmaResiduals**() const

Checks whether residuals should be added to account for parameter dependent sigma.

**Returns** sigma\_res

std::vector<int> const &**getParameterList**() const

Get the list of parameters for which sensitivities are computed.

**Returns** List of parameter indices

int **plist**(int pos) const

Get entry in parameter list by index.

**Parameters** **pos** – Index in sensitivity parameter list

**Returns** Index in parameter list

void **setParameterList**(std::vector<int> const &plist)

Set the list of parameters for which sensitivities are to be computed.

NOTE: Resets initial state sensitivities.

**Parameters** **plist** – List of parameter indices

std::vector<*realtype*> **getInitialStates**()

Get the initial states.

**Returns** Initial state vector

void **setInitialStates**(std::vector<*realtype*> const &x0)

Set the initial states.

**Parameters** **x0** – Initial state vector

bool **hasCustomInitialStates**() const

Return whether custom initial states have been set.

**Returns** true if has custom initial states, otherwise false

std::vector<*realtype*> **getInitialStateSensitivities**()

Get the initial states sensitivities.

**Returns** vector of initial state sensitivities

void **setInitialStateSensitivities**(std::vector<*realtype*> const &sx0)

Set the initial state sensitivities.

**Parameters** **sx0** – vector of initial state sensitivities with chainrule applied. This could be a slice of *ReturnData::sx* or *ReturnData::sx0*

bool **hasCustomInitialStateSensitivities**() const

Return whether custom initial state sensitivities have been set.

**Returns** true if has custom initial state sensitivities, otherwise false.

void **setUnscaledInitialStateSensitivities**(std::vector<*realtype*> const &sx0)

Set the initial state sensitivities.

**Parameters** **sx0** – Vector of initial state sensitivities without chainrule applied. This could be the readin from a `model.sx0data` saved to HDF5.

void **setSteadyStateComputationMode**(*SteadyStateComputationMode* mode)

Set the mode how steady state is computed in the steadystate simulation.

**Parameters** **mode** – Steadystate computation mode

*SteadyStateComputationMode* **getSteadyStateComputationMode**() const

Gets the mode how steady state is computed in the steadystate simulation.

**Returns** Mode

void **setSteadyStateSensitivityMode**(*SteadyStateSensitivityMode* mode)

Set the mode how sensitivities are computed in the steadystate simulation.

**Parameters** **mode** – Steadystate sensitivity mode

*SteadyStateSensitivityMode* **getSteadyStateSensitivityMode**() const

Gets the mode how sensitivities are computed in the steadystate simulation.

**Returns** Mode

void **setReinitializeFixedParameterInitialStates**(bool flag)

Set whether initial states depending on fixed parameters are to be reinitialized after preequilibration and presimulation.

**Parameters** **flag** – Fixed parameters reinitialized?

bool **getReinitializeFixedParameterInitialStates**() const

Get whether initial states depending on fixedParameters are to be reinitialized after preequilibration and presimulation.

**Returns** flag true / false

void **requireSensitivitiesForAllParameters**()

Require computation of sensitivities for all parameters p [0..np[ in natural order.

NOTE: Resets initial state sensitivities.

void **getExpression**(gsl::span<*realtype*> w, const *realtype* t, *AmiVector* const &x)

Get time-resolved w.

**Parameters**

- **w** – Buffer (shape nw)
- **t** – Current timepoint
- **x** – Current state

void **getObservable**(gsl::span<*realtype*> y, const *realtype* t, *AmiVector* const &x)

Get time-resolved observables.

**Parameters**

- **y** – Buffer (shape ny)
- **t** – Current timepoint
- **x** – Current state

virtual *ObservableScaling* **getObservableScaling**(int iy) const

Get scaling type for observable.

**Parameters** *iy* – observable index

**Returns** scaling type

void **getObservableSensitivity**(gsl::span<*realtype*> sy, const *realtype* t, *AmiVector* const &x,  
*AmiVectorArray* const &sx)

Get sensitivity of time-resolved observables.

Total derivative  $sy = dydx * sx + dydp$  (only for forward sensitivities).

**Parameters**

- **sy** – buffer (shape ny x nplist, row-major)
- **t** – Timepoint
- **x** – State variables
- **sx** – State sensitivities

void **getObservableSigma**(gsl::span<*realtype*> sigmay, int const it, *ExpData* const \*edata)

Get time-resolved observable standard deviations.

**Parameters**

- **sigmay** – Buffer (shape ny)
- **it** – Timepoint index
- **edata** – Pointer to experimental data instance (optional, pass nullptr to ignore)

void **getObservableSigmaSensitivity**(gsl::span<*realtype*> ssigmay, gsl::span<*realtype* const> sy, int  
const it, *ExpData* const \*edata)

Sensitivity of time-resolved observable standard deviation.

Total derivative (can be used with both adjoint and forward sensitivity).

**Parameters**

- **ssigmay** – Buffer (shape ny x nplist, row-major)
- **sy** – Sensitivity of time-resolved observables for current timepoint
- **it** – Timepoint index
- **edata** – Pointer to experimental data instance (optional, pass nullptr to ignore)

void **addObservableObjective**(*realtype* &Jy, int const it, *AmiVector* const &x, *ExpData* const &edata)

Add time-resolved measurement negative log-likelihood *Jy*.

**Parameters**

- **Jy** – Buffer (shape 1)
- **it** – Timepoint index
- **x** – State variables
- **edata** – Experimental data

```
void addObservableObjectiveSensitivity(std::vector<realtype> &s1lh, std::vector<realtype> &s2lh, int
                                     const it, AmiVector const &x, AmiVectorArray const &sx,
                                     ExpData const &edata)
```

Add sensitivity of time-resolved measurement negative log-likelihood  $J_y$ .

#### Parameters

- **s1lh** – First-order buffer (shape `nplist`)
- **s2lh** – Second-order buffer (shape `nJ - 1 x nplist`, row-major)
- **it** – Timepoint index
- **x** – State variables
- **sx** – State sensitivities
- **edata** – Experimental data

```
void addPartialObservableObjectiveSensitivity(std::vector<realtype> &s1lh, std::vector<realtype>
                                             &s2lh, int const it, AmiVector const &x, ExpData
                                             const &edata)
```

Add sensitivity of time-resolved measurement negative log-likelihood  $J_y$ .

Partial derivative (to be used with adjoint sensitivities).

#### Parameters

- **s1lh** – First order output buffer (shape `nplist`)
- **s2lh** – Second order output buffer (shape `nJ - 1 x nplist`, row-major)
- **it** – Timepoint index
- **x** – State variables
- **edata** – Experimental data

```
void getAdjointStateObservableUpdate(gsl::span<realtype> dJydx, int const it, AmiVector const &x,
                                     ExpData const &edata)
```

Get state sensitivity of the negative loglikelihood  $J_y$ , partial derivative (to be used with adjoint sensitivities).

#### Parameters

- **dJydx** – Output buffer (shape `nJ x nx_solver`, row-major)
- **it** – Timepoint index
- **x** – State variables
- **edata** – Experimental data instance

```
void getEvent(gsl::span<realtype> z, int const ie, const realtype t, AmiVector const &x)
```

Get event-resolved observables.

#### Parameters

- **z** – Output buffer (shape `nz`)
- **ie** – Event index
- **t** – Timepoint
- **x** – State variables

void **getEventSensitivity**(gsl::span<*realtype*> sz, int const ie, const *realtype* t, *AmiVector* const &x, *AmiVectorArray* const &sx)

Get sensitivities of event-resolved observables.

Total derivative (only forward sensitivities).

**Parameters**

- **sz** – Output buffer (shape `nz x nplist`, row-major)
- **ie** – Event index
- **t** – Timepoint
- **x** – State variables
- **sx** – State sensitivities

void **getUnobservedEventSensitivity**(gsl::span<*realtype*> sz, int const ie)

Get sensitivity of `z` at final timepoint.

Ignores sensitivity of timepoint. Total derivative.

**Parameters**

- **sz** – Output buffer (shape `nz x nplist`, row-major)
- **ie** – Event index

void **getEventRegularization**(gsl::span<*realtype*> rz, int const ie, const *realtype* t, *AmiVector* const &x)

Get regularization for event-resolved observables.

**Parameters**

- **rz** – Output buffer (shape `nz`)
- **ie** – Event index
- **t** – Timepoint
- **x** – State variables

void **getEventRegularizationSensitivity**(gsl::span<*realtype*> srz, int const ie, const *realtype* t, *AmiVector* const &x, *AmiVectorArray* const &sx)

Get sensitivities of regularization for event-resolved observables.

Total derivative. Only forward sensitivities.

**Parameters**

- **srz** – Output buffer (shape `nz x nplist`, row-major)
- **ie** – Event index
- **t** – Timepoint
- **x** – State variables
- **sx** – State sensitivities

void **getEventSigma**(gsl::span<*realtype*> sigmaz, int const ie, int const nroots, const *realtype* t, *ExpData* const \*edata)

Get event-resolved observable standard deviations.

**Parameters**

- **sigmaz** – Output buffer (shape `nz`)



- **ie** – Event index
- **nroots** – Event occurrence
- **t** – Timepoint
- **edata** – Pointer to experimental data (optional, pass `nullptr` to ignore)

void **getEventSigmaSensitivity**(gsl::span<*realtype*> ssmaz, int const ie, int const nroots, const *realtype* t, *ExpData* const \*edata)

Get sensitivities of event-resolved observable standard deviations.

Total derivative (only forward sensitivities).

#### Parameters

- **ssmaz** – Output buffer (shape `nz x nplist`, row-major)
- **ie** – Event index
- **nroots** – Event occurrence
- **t** – Timepoint
- **edata** – Pointer to experimental data (optional, pass `nullptr` to ignore)

void **addEventObjective**(*realtype* &Jz, int const ie, int const nroots, const *realtype* t, *AmiVector* const &x, *ExpData* const &edata)

Add event-resolved observable negative log-likelihood.

#### Parameters

- **Jz** – Output buffer (shape 1)
- **ie** – Event index
- **nroots** – Event occurrence
- **t** – Timepoint
- **x** – State variables
- **edata** – Experimental data

void **addEventObjectiveRegularization**(*realtype* &Jrz, int const ie, int const nroots, const *realtype* t, *AmiVector* const &x, *ExpData* const &edata)

Add event-resolved observable negative log-likelihood.

#### Parameters

- **Jrz** – Output buffer (shape 1)
- **ie** – Event index
- **nroots** – Event occurrence
- **t** – Timepoint
- **x** – State variables
- **edata** – Experimental data

void **addEventObjectiveSensitivity**(std::vector<*realtype*> &sllh, std::vector<*realtype*> &s2llh, int const ie, int const nroots, const *realtype* t, *AmiVector* const &x, *AmiVectorArray* const &sx, *ExpData* const &edata)

Add sensitivity of time-resolved measurement negative log-likelihood  $J_y$ .

Total derivative (to be used with forward sensitivities).

#### Parameters

- **s1lh** – First order buffer (shape `nplist`)
- **s2lh** – Second order buffer (shape `nJ-1 x nplist`, row-major)
- **ie** – Event index
- **nroots** – Event occurrence
- **t** – Timepoint
- **x** – State variables
- **sx** – State sensitivities
- **edata** – Experimental data

```
void addPartialEventObjectiveSensitivity(std::vector<realtype> &s1lh, std::vector<realtype> &s2lh,  
                                         int const ie, int const nroots, const realtype t, AmiVector  
                                         const &x, ExpData const &edata)
```

Add sensitivity of time-resolved measurement negative log-likelihood  $J_y$ .

Partial derivative (to be used with adjoint sensitivities).

#### Parameters

- **s1lh** – First order buffer (shape `nplist`)
- **s2lh** – Second order buffer (shape `(nJ-1) x nplist`, row-major)
- **ie** – Event index
- **nroots** – Event occurrence
- **t** – Timepoint
- **x** – State variables
- **edata** – Experimental data

```
void getAdjointStateEventUpdate(gsl::span<realtype> dJzdx, int const ie, int const nroots, const realtype  
                                t, AmiVector const &x, ExpData const &edata)
```

State sensitivity of the negative loglikelihood  $J_z$ .

Partial derivative (to be used with adjoint sensitivities).

#### Parameters

- **dJzdx** – Output buffer (shape `nJ x nx_solver`, row-major)
- **ie** – Event index
- **nroots** – Event occurrence
- **t** – Timepoint
- **x** – State variables
- **edata** – Experimental data

void **getEventTimeSensitivity**(std::vector<realtype> &stau, const realtype t, int const ie, *AmiVector* const &x, *AmiVectorArray* const &sx)

Sensitivity of event timepoint, total derivative.

Only forward sensitivities.

#### Parameters

- **stau** – Timepoint sensitivity (shape nplist)
- **t** – Timepoint
- **ie** – Event index
- **x** – State variables
- **sx** – State sensitivities

void **addStateEventUpdate**(*AmiVector* &x, int const ie, const realtype t, *AmiVector* const &xdot, *AmiVector* const &xdot\_old)

Update state variables after event.

#### Parameters

- **x** – Current state (will be overwritten)
- **ie** – Event index
- **t** – Current timepoint
- **xdot** – Current residual function values
- **xdot\_old** – Value of residual function before event

void **addStateSensitivityEventUpdate**(*AmiVectorArray* &sx, int const ie, const realtype t, *AmiVector* const &x\_old, *AmiVector* const &xdot, *AmiVector* const &xdot\_old, std::vector<realtype> const &stau)

Update state sensitivity after event.

#### Parameters

- **sx** – Current state sensitivity (will be overwritten)
- **ie** – Event index
- **t** – Current timepoint
- **x\_old** – Current state
- **xdot** – Current residual function values
- **xdot\_old** – Value of residual function before event
- **stau** – Timepoint sensitivity, to be computed with *Model::getEventTimeSensitivity*

void **addAdjointStateEventUpdate**(*AmiVector* &xB, int const ie, const realtype t, *AmiVector* const &x, *AmiVector* const &xdot, *AmiVector* const &xdot\_old)

Update adjoint state after event.

#### Parameters

- **xB** – Current adjoint state (will be overwritten)
- **ie** – Event index
- **t** – Current timepoint

- **x** – Current state
- **xdot** – Current residual function values
- **xdot\_old** – Value of residual function before event

void **addAdjointQuadratureEventUpdate**(*AmiVector* xQB, int const ie, const *realtype* t, *AmiVector* const &x, *AmiVector* const &xB, *AmiVector* const &xdot, *AmiVector* const &xdot\_old)

Update adjoint quadratures after event.

#### Parameters

- **xQB** – Current quadrature state (will be overwritten)
- **ie** – Event index
- **t** – Current timepoint
- **x** – Current state
- **xB** – Current adjoint state
- **xdot** – Current residual function values
- **xdot\_old** – Value of residual function before event

void **updateHeaviside**(std::vector<int> const &rootsfound)

Update the Heaviside variables **h** on event occurrences.

**Parameters rootsfound** – Provides the direction of the zero-crossing, so adding it will give the right update to the Heaviside variables (zero if no root was found)

void **updateHeavisideB**(int const \*rootsfound)

Updates the Heaviside variables **h** on event occurrences in the backward problem.

**Parameters rootsfound** – Provides the direction of the zero-crossing, so adding it will give the right update to the Heaviside variables (zero if no root was found)

int **checkFinite**(gsl::span<*realtype* const> array, *ModelQuantity* model\_quantity) const

Check if the given array has only finite elements.

For (1D) spans.

#### Parameters

- **array** –
- **model\_quantity** – The model quantity array corresponds to

#### Returns

int **checkFinite**(gsl::span<*realtype* const> array, *ModelQuantity* model\_quantity, size\_t num\_cols) const

Check if the given array has only finite elements.

For flattened 2D arrays.

#### Parameters

- **array** – Flattened matrix
- **model\_quantity** – The model quantity array corresponds to
- **num\_cols** – Number of columns of the non-flattened matrix

#### Returns

int **checkFinite**(SUNMatrix m, *ModelQuantity* model\_quantity, *realtype* t) const

Check if the given array has only finite elements.

For SUNMatrix.

#### Parameters

- **m** – Matrix to check
- **model\_quantity** – The model quantity **m** corresponds to
- **t** – current timepoint

#### Returns

void **setAlwaysCheckFinite**(bool alwaysCheck)

Set whether the result of every call to `Model::f*` should be checked for finiteness.

#### Parameters **alwaysCheck** –

bool **getAlwaysCheckFinite**() const

Get setting of whether the result of every call to `Model::f*` should be checked for finiteness.

#### Returns that

void **fx0**(*AmiVector* &x)

Compute/get initial states.

#### Parameters **x** – Output buffer.

void **fx0\_fixedParameters**(*AmiVector* &x)

Set only those initial states that are specified via fixed parameters.

#### Parameters **x** – Output buffer.

void **fsx0**(*AmiVectorArray* &sx, *AmiVector* const &x)

Compute/get initial value for initial state sensitivities.

#### Parameters

- **sx** – Output buffer for state sensitivities
- **x** – State variables

void **fsx0\_fixedParameters**(*AmiVectorArray* &sx, *AmiVector* const &x)

Get only those initial states sensitivities that are affected from `amici::Model::fx0_fixedParameters`.

#### Parameters

- **sx** – Output buffer for state sensitivities
- **x** – State variables

virtual void **fsdx0**()

Compute sensitivity of derivative initial states sensitivities **sdx0**.

Only necessary for DAEs.

void **fx\_rdata**(*AmiVector* &x\_rdata, *AmiVector* const &x\_solver)

Expand conservation law for states.

#### Parameters

- **x\_rdata** – Output buffer for state variables with conservation laws expanded (stored in `amici::ReturnData`).

- **x\_solver** – State variables with conservation laws applied (solver returns this)

void **fsx\_rdata**(*AmiVectorArray* &sx\_rdata, *AmiVectorArray* const &sx\_solver, *AmiVector* const &x\_solver)

Expand conservation law for state sensitivities.

**Parameters**

- **sx\_rdata** – Output buffer for state variables sensitivities with conservation laws expanded (stored in *amici::ReturnData*).
- **sx\_solver** – State variables sensitivities with conservation laws applied (solver returns this)
- **x\_solver** – State variables with conservation laws applied (solver returns this)

void **setReinitializationStateIdxs**(std::vector<int> const &idxs)

Set indices of states to be reinitialized based on provided constants / fixed parameters.

**Parameters** **idxs** – Array of state indices

std::vector<int> const &**getReinitializationStateIdxs**() const

Return indices of states to be reinitialized based on provided constants / fixed parameters.

**Returns** Those indices.

*AmiVectorArray* const &**get\_dxdotdp**() const

getter for dxdotdp (matlab generated)

**Returns** dxdotdp

*SUNMatrixWrapper* const &**get\_dxdotdp\_full**() const

getter for dxdotdp (python generated)

**Returns** dxdotdp

virtual void **fdeltaqB**(*realtype* \*deltaqB, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, int ip, int ie, *realtype* const \*xdot, *realtype* const \*xdot\_old, *realtype* const \*xB)

Model-specific implementation of fdeltaqB.

**Parameters**

- **deltaqB** – sensitivity update
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **ip** – sensitivity index
- **ie** – event index
- **xdot** – new model right hand side
- **xdot\_old** – previous model right hand side
- **xB** – adjoint state

```
virtual void fdeltasx(realtype *deltasx, const realtype t, realtype const *x, realtype const *p, realtype const
    *k, realtype const *h, realtype const *w, int ip, int ie, realtype const *xdot, realtype
    const *xdot_old, realtype const *sx, realtype const *stau, realtype const *tcl)
```

Model-specific implementation of fdeltasx.

#### Parameters

- **deltasx** – sensitivity update
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **w** – repeating elements vector
- **ip** – sensitivity index
- **ie** – event index
- **xdot** – new model right hand side
- **xdot\_old** – previous model right hand side
- **sx** – state sensitivity
- **stau** – event-time sensitivity
- **tcl** – total abundances for conservation laws

```
virtual void fdeltax(realtype *deltax, const realtype t, realtype const *x, realtype const *p, realtype const *k,
    realtype const *h, int ie, realtype const *xdot, realtype const *xdot_old)
```

Model-specific implementation of fdeltax.

#### Parameters

- **deltax** – state update
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **ie** – event index
- **xdot** – new model right hand side
- **xdot\_old** – previous model right hand side

```
virtual void fdeltaxB(realtype *deltaxB, const realtype t, realtype const *x, realtype const *p, realtype const
    *k, realtype const *h, int ie, realtype const *xdot, realtype const *xdot_old, realtype
    const *xB)
```

Model-specific implementation of fdeltaxB.

#### Parameters

- **deltaxB** – adjoint state update

- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **ie** – event index
- **xdot** – new model right hand side
- **xdot\_old** – previous model right hand side
- **xB** – current adjoint state

virtual void **fdJrzdsigma**(*realtype* \*dJrzdsigma, int iz, *realtype* const \*p, *realtype* const \*k, *realtype* const \*rz, *realtype* const \*sigmaz)

Model-specific implementation of fdJrzdsigma.

#### Parameters

- **dJrzdsigma** – Sensitivity of event penalization Jrz w.r.t. standard deviation sigmaz
- **iz** – event output index
- **p** – parameter vector
- **k** – constant vector
- **rz** – model root output at timepoint
- **sigmaz** – event measurement standard deviation at timepoint

virtual void **fdJrzdz**(*realtype* \*dJrzdz, int iz, *realtype* const \*p, *realtype* const \*k, *realtype* const \*rz, *realtype* const \*sigmaz)

Model-specific implementation of fdJrzdz.

#### Parameters

- **dJrzdz** – partial derivative of event penalization Jrz
- **iz** – event output index
- **p** – parameter vector
- **k** – constant vector
- **rz** – model root output at timepoint
- **sigmaz** – event measurement standard deviation at timepoint

virtual void **fdJydsigma**(*realtype* \*dJydsigma, int iy, *realtype* const \*p, *realtype* const \*k, *realtype* const \*y, *realtype* const \*sigmay, *realtype* const \*my)

Model-specific implementation of fdJydsigma.

#### Parameters

- **dJydsigma** – Sensitivity of time-resolved measurement negative log-likelihood Jy w.r.t. standard deviation sigmay
- **iy** – output index
- **p** – parameter vector
- **k** – constant vector



- **y** – model output at timepoint
- **sigmay** – measurement standard deviation at timepoint
- **my** – measurement at timepoint

virtual void **fdJydy**(*realtype* \*dJydy, int iy, *realtype* const \*p, *realtype* const \*k, *realtype* const \*y, *realtype* const \*sigmay, *realtype* const \*my)

Model-specific implementation of fdJydy.

#### Parameters

- **dJydy** – partial derivative of time-resolved measurement negative log-likelihood Jy
- **iy** – output index
- **p** – parameter vector
- **k** – constant vector
- **y** – model output at timepoint
- **sigmay** – measurement standard deviation at timepoint
- **my** – measurement at timepoint

virtual void **fdJydy\_colptrs**(*SUNMatrixWrapper* &dJydy, int index)

Model-specific implementation of fdJydy colptrs.

#### Parameters

- **dJydy** – sparse matrix to which colptrs will be written
- **index** – ytrue index

virtual void **fdJydy\_rowvals**(*SUNMatrixWrapper* &dJydy, int index)

Model-specific implementation of fdJydy rowvals.

#### Parameters

- **dJydy** – sparse matrix to which rowvals will be written
- **index** – ytrue index

virtual void **fdJzdsigma**(*realtype* \*dJzdsigma, int iz, *realtype* const \*p, *realtype* const \*k, *realtype* const \*z, *realtype* const \*sigmaz, *realtype* const \*mz)

Model-specific implementation of fdJzdsigma.

#### Parameters

- **dJzdsigma** – Sensitivity of event measurement negative log-likelihood Jz w.r.t. standard deviation sigmaz
- **iz** – event output index
- **p** – parameter vector
- **k** – constant vector
- **z** – model event output at timepoint
- **sigmaz** – event measurement standard deviation at timepoint
- **mz** – event measurement at timepoint

```
virtual void fdJzdz(realtype *dJzdz, int iz, realtype const *p, realtype const *k, realtype const *z, realtype const *sigmaz, realtype const *mz)
```

Model-specific implementation of fdJzdz.

#### Parameters

- **dJzdz** – partial derivative of event measurement negative log-likelihood Jz
- **iz** – event output index
- **p** – parameter vector
- **k** – constant vector
- **z** – model event output at timepoint
- **sigmaz** – event measurement standard deviation at timepoint
- **mz** – event measurement at timepoint

```
virtual void fdrzdp(realtype *drzdp, int ie, const realtype t, realtype const *x, realtype const *p, realtype const *k, realtype const *h, int ip)
```

Model-specific implementation of fdrzdp.

#### Parameters

- **drzdp** – partial derivative of root output rz w.r.t. model parameters p
- **ie** – event index
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **ip** – parameter index w.r.t. which the derivative is requested

```
virtual void fdrzdx(realtype *drzdx, int ie, const realtype t, realtype const *x, realtype const *p, realtype const *k, realtype const *h)
```

Model-specific implementation of fdrzdx.

#### Parameters

- **drzdx** – partial derivative of root output rz w.r.t. model states x
- **ie** – event index
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector

```
virtual void fdsigmaydp(realtype *dsigmaydp, const realtype t, realtype const *p, realtype const *k, realtype const *y, int ip)
```

Model-specific implementation of fdsigmaydp.

#### Parameters

- **dsigmaydp** – partial derivative of standard deviation of measurements
- **t** – current time
- **p** – parameter vector
- **k** – constant vector
- **y** – model output at timepoint t
- **ip** – sensitivity index

virtual void **fdsigmaydy**(*realtype* \*dsigmaydy, const *realtype* t, *realtype* const \*p, *realtype* const \*k, *realtype* const \*y)

Model-specific implementation of fsigmay.

#### Parameters

- **dsigmaydy** – partial derivative of standard deviation of measurements w.r.t. model outputs
- **t** – current time
- **p** – parameter vector
- **k** – constant vector
- **y** – model output at timepoint t

virtual void **fdsigmazdp**(*realtype* \*dsigmazdp, const *realtype* t, *realtype* const \*p, *realtype* const \*k, int ip)

Model-specific implementation of fsigmaz.

#### Parameters

- **dsigmazdp** – partial derivative of standard deviation of event measurements
- **t** – current time
- **p** – parameter vector
- **k** – constant vector
- **ip** – sensitivity index

virtual void **fdtotal\_cldp**(*realtype* \*dtotal\_cldp, *realtype* const \*x\_rdata, *realtype* const \*p, *realtype* const \*k, int const ip)

Compute dtotal\_cl / dp.

#### Parameters

- **dtotal\_cldp** – dtotal\_cl / dp
- **x\_rdata** – State variables with conservation laws applied
- **p** – parameter vector
- **k** – constant vector
- **ip** – Sensitivity index

virtual void **fdtotal\_cldx\_rdata**(*realtype* \*dtotal\_cldx\_rdata, *realtype* const \*x\_rdata, *realtype* const \*p, *realtype* const \*k, *realtype* const \*tcl)

Compute dtotal\_cl / dx\_rdata.

#### Parameters

- **dtotal\_cldx\_rdata** – dtotal\_cl / dx\_rdata
- **x\_rdata** – State variables with conservation laws applied

- **p** – parameter vector
- **k** – constant vector
- **tcl** – Total abundances for conservation laws

virtual void **fdtotal\_cldx\_rdata\_colptrs**(*SUNMatrixWrapper* &dtotal\_cldx\_rdata)

Model-specific implementation of fdtotal\_cldx\_rdata, colptrs part.

**Parameters** **dtotal\_cldx\_rdata** – sparse matrix to which colptrs will be written

virtual void **fdtotal\_cldx\_rdata\_rowvals**(*SUNMatrixWrapper* &dtotal\_cldx\_rdata)

Model-specific implementation of fdtotal\_cldx\_rdata, rowvals part.

**Parameters** **dtotal\_cldx\_rdata** – sparse matrix to which rowvals will be written

virtual void **fdwdp**(*realtype* \*dwdp, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w, *realtype* const \*tcl, *realtype* const \*stcl, *realtype* const \*spl, *realtype* const \*sspl)

Model-specific sparse implementation of dwdp.

**Parameters**

- **dwdp** – Recurring terms in xdot, parameter derivative
- **t** – timepoint
- **x** – vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **w** – vector with helper variables
- **tcl** – total abundances for conservation laws
- **stcl** – sensitivities of total abundances for conservation laws
- **spl** – spline value vector
- **sspl** – sensitivities of spline values vector w.r.t. parameters *p*

virtual void **fdwdp**(*realtype* \*dwdp, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w, *realtype* const \*tcl, *realtype* const \*stcl, *realtype* const \*spl, *realtype* const \*sspl, int ip)

Model-specific sensitivity implementation of dwdp.

**Parameters**

- **dwdp** – Recurring terms in xdot, parameter derivative
- **t** – timepoint
- **x** – vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **w** – vector with helper variables
- **tcl** – total abundances for conservation laws

- **stcl** – sensitivities of total abundances for conservation laws
- **spl** – spline value vector
- **sspl** – sensitivities of spline values vector
- **ip** – sensitivity parameter index

virtual void **fdwdp\_colptrs**(*SUNMatrixWrapper* &dwdp)

Model-specific implementation for dwdp, column pointers.

**Parameters** **dwdp** – sparse matrix to which colptrs will be written

virtual void **fdwdp\_rowvals**(*SUNMatrixWrapper* &dwdp)

Model-specific implementation for dwdp, row values.

**Parameters** **dwdp** – sparse matrix to which rowvals will be written

virtual void **fdwdw**(*realtype* \*dwdw, *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w, *realtype* const \*tcl)

Model-specific implementation of fdwdw, no w chainrule (Py)

**Parameters**

- **dwdw** – partial derivative w wrt w
- **t** – timepoint
- **x** – Vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **w** – vector with helper variables
- **tcl** – Total abundances for conservation laws

virtual void **fdwdw\_colptrs**(*SUNMatrixWrapper* &dwdw)

Model-specific implementation of fdwdw, colptrs part.

**Parameters** **dwdw** – sparse matrix to which colptrs will be written

virtual void **fdwdw\_rowvals**(*SUNMatrixWrapper* &dwdw)

Model-specific implementation of fdwdw, rowvals part.

**Parameters** **dwdw** – sparse matrix to which rowvals will be written

virtual void **fdwdx**(*realtype* \*dwdx, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w, *realtype* const \*tcl, *realtype* const \*spl)

Model-specific implementation of dwdx, data part.

**Parameters**

- **dwdx** – Recurring terms in xdot, state derivative
- **t** – timepoint
- **x** – vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector

- **w** – vector with helper variables
- **tcl** – total abundances for conservation laws
- **spl** – spline value vector

virtual void **fdwdx\_colptrs**(*SUNMatrixWrapper* &dwdx)

Model-specific implementation for dwdx, column pointers.

**Parameters** **dwdx** – sparse matrix to which colptrs will be written

virtual void **fdwdx\_rowvals**(*SUNMatrixWrapper* &dwdx)

Model-specific implementation for dwdx, row values.

**Parameters** **dwdx** – sparse matrix to which rowvals will be written

virtual void **fdx\_rdatadp**(*realtype* \*dx\_rdatadp, *realtype* const \*x, *realtype* const \*tcl, *realtype* const \*p, *realtype* const \*k, int const ip)

Compute dx\_rdata / dp.

**Parameters**

- **dx\_rdatadp** – dx\_rdata / dp
- **p** – parameter vector
- **k** – constant vector
- **x** – State variables with conservation laws applied
- **tcl** – Total abundances for conservation laws
- **ip** – Sensitivity index

virtual void **fdx\_rdatadtcl**(*realtype* \*dx\_rdatadtcl, *realtype* const \*x, *realtype* const \*tcl, *realtype* const \*p, *realtype* const \*k)

Compute dx\_rdata / dtcl.

**Parameters**

- **dx\_rdatadtcl** – dx\_rdata / dtcl
- **p** – parameter vector
- **k** – constant vector
- **x** – State variables with conservation laws applied
- **tcl** – Total abundances for conservation laws

virtual void **fdx\_rdatadtcl\_colptrs**(*SUNMatrixWrapper* &dx\_rdatadtcl)

Model-specific implementation of fdx\_rdatadtcl, colptrs part.

**Parameters** **dx\_rdatadtcl** – sparse matrix to which colptrs will be written

virtual void **fdx\_rdatadtcl\_rowvals**(*SUNMatrixWrapper* &dx\_rdatadtcl)

Model-specific implementation of fdx\_rdatadtcl, rowvals part.

**Parameters** **dx\_rdatadtcl** – sparse matrix to which rowvals will be written

virtual void **fdx\_rdatadx\_solver**(*realtype* \*dx\_rdatadx\_solver, *realtype* const \*x, *realtype* const \*tcl, *realtype* const \*p, *realtype* const \*k)

Compute dx\_rdata / dx\_solver.

**Parameters**

- **dx\_rdatadx\_solver** – dx\_rdata / dx\_solver
- **p** – parameter vector
- **k** – constant vector
- **x** – State variables with conservation laws applied
- **tcl** – Total abundances for conservation laws

virtual void **fdx\_rdatadx\_solver\_colptrs**(*SUNMatrixWrapper* &dxrdatadxsolver)

Model-specific implementation of fdx\_rdatadx\_solver, colptrs part.

**Parameters dxrdatadxsolver** – sparse matrix to which colptrs will be written

virtual void **fdx\_rdatadx\_solver\_rowvals**(*SUNMatrixWrapper* &dxrdatadxsolver)

Model-specific implementation of fdx\_rdatadx\_solver, rowvals part.

**Parameters dxrdatadxsolver** – sparse matrix to which rowvals will be written

virtual void **fdydp**(*realtype* \*dydp, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, int ip, *realtype* const \*w, *realtype* const \*dwdp)

Model-specific implementation of fdydp (MATLAB-only)

**Parameters**

- **dydp** – partial derivative of observables y w.r.t. model parameters p
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **ip** – parameter index w.r.t. which the derivative is requested
- **w** – repeating elements vector
- **dwdp** – Recurring terms in xdot, parameter derivative

virtual void **fdydp**(*realtype* \*dydp, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, int ip, *realtype* const \*w, *realtype* const \*tcl, *realtype* const \*dtcldp, *realtype* const \*spl, *realtype* const \*sspl)

Model-specific implementation of fdydp (Python)

**Parameters**

- **dydp** – partial derivative of observables y w.r.t. model parameters p
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **ip** – parameter index w.r.t. which the derivative is requested
- **w** – repeating elements vector
- **tcl** – total abundances for conservation laws

- **dtcl****dp** – Sensitivities of total abundances for conservation laws
- **spl** – spline value vector
- **sspl** – sensitivities of spline values vector w.r.t. parameters  $p$

virtual void **fdydx**(*realtype* \*dydx, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w, *realtype* const \*dwdx)

Model-specific implementation of fdydx.

#### Parameters

- **dydx** – partial derivative of observables  $y$  w.r.t. model states  $x$
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **w** – repeating elements vector
- **dwdx** – Recurring terms in  $\dot{x}$ , state derivative

virtual void **fdzdp**(*realtype* \*dzdp, int ie, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, int ip)

Model-specific implementation of fdzdp.

#### Parameters

- **dzdp** – partial derivative of event-resolved output  $z$  w.r.t. model parameters  $p$
- **ie** – event index
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **ip** – parameter index w.r.t. which the derivative is requested

virtual void **fdzdx**(*realtype* \*dzdx, int ie, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h)

Model-specific implementation of fdzdx.

#### Parameters

- **dzdx** – partial derivative of event-resolved output  $z$  w.r.t. model states  $x$
- **ie** – event index
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector



- **h** – Heaviside vector

virtual void **fJrz**(*realtype* \*nllh, int iz, *realtype* const \*p, *realtype* const \*k, *realtype* const \*z, *realtype* const \*sigmaz)

Model-specific implementation of fJrz.

#### Parameters

- **nllh** – regularization for event measurements z
- **iz** – event output index
- **p** – parameter vector
- **k** – constant vector
- **z** – model event output at timepoint
- **sigmaz** – event measurement standard deviation at timepoint

virtual void **fJy**(*realtype* \*nllh, int iy, *realtype* const \*p, *realtype* const \*k, *realtype* const \*y, *realtype* const \*sigmay, *realtype* const \*my)

Model-specific implementation of fJy.

#### Parameters

- **nllh** – negative log-likelihood for measurements y
- **iy** – output index
- **p** – parameter vector
- **k** – constant vector
- **y** – model output at timepoint
- **sigmay** – measurement standard deviation at timepoint
- **my** – measurements at timepoint

virtual void **fJz**(*realtype* \*nllh, int iz, *realtype* const \*p, *realtype* const \*k, *realtype* const \*z, *realtype* const \*sigmaz, *realtype* const \*mz)

Model-specific implementation of fJz.

#### Parameters

- **nllh** – negative log-likelihood for event measurements z
- **iz** – event output index
- **p** – parameter vector
- **k** – constant vector
- **z** – model event output at timepoint
- **sigmaz** – event measurement standard deviation at timepoint
- **mz** – event measurements at timepoint

virtual void **frz**(*realtype* \*rz, int ie, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h)

Model-specific implementation of frz.

#### Parameters

- **rz** – value of root function at current timepoint (non-output events not included)

- **ie** – event index
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector

virtual void **fsgmay**(*realtype* \*sigmay, const *realtype* t, *realtype* const \*p, *realtype* const \*k, *realtype* const \*y)

Model-specific implementation of fsgmay.

#### Parameters

- **sigmay** – standard deviation of measurements
- **t** – current time
- **p** – parameter vector
- **k** – constant vector
- **y** – model output at timepoint t

virtual void **fsgmaz**(*realtype* \*sigmaz, const *realtype* t, *realtype* const \*p, *realtype* const \*k)

Model-specific implementation of fsgmaz.

#### Parameters

- **sigmaz** – standard deviation of event measurements
- **t** – current time
- **p** – parameter vector
- **k** – constant vector

virtual void **fsrz**(*realtype* \*srz, int ie, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*sx, int ip)

Model-specific implementation of fsrz.

#### Parameters

- **srz** – Sensitivity of rz, total derivative
- **ie** – event index
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **sx** – current state sensitivity
- **h** – Heaviside vector
- **ip** – sensitivity index

```
virtual void fstau(realtype *stau, const realtype t, realtype const *x, realtype const *p, realtype const *k,
                  realtype const *h, realtype const *tcl, realtype const *sx, int ip, int ie)
```

Model-specific implementation of fstau.

#### Parameters

- **stau** – total derivative of event timepoint
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **tcl** – total abundances for conservation laws
- **sx** – current state sensitivity
- **ip** – sensitivity index
- **ie** – event index

```
virtual void fsx0(realtype *sx0, const realtype t, realtype const *x0, realtype const *p, realtype const *k, int
                  ip)
```

Model-specific implementation of fsx0.

#### Parameters

- **sx0** – initial state sensitivities
- **t** – initial time
- **x0** – initial state
- **p** – parameter vector
- **k** – constant vector
- **ip** – sensitivity index

```
virtual void fsx0_fixedParameters(realtype *sx0, const realtype t, realtype const *x0, realtype const *p,
                                   realtype const *k, int ip, gsl::span<int const>
                                   reinitialization_state_idx)
```

Model-specific implementation of fsx0\_fixedParameters.

#### Parameters

- **sx0** – initial state sensitivities
- **t** – initial time
- **x0** – initial state
- **p** – parameter vector
- **k** – constant vector
- **ip** – sensitivity index
- **reinitialization\_state\_idx** – Indices of states to be reinitialized based on provided constants / fixed parameters.

```
virtual void fsz(realtype *sz, int ie, const realtype t, realtype const *x, realtype const *p, realtype const *k,  
                realtype const *h, realtype const *sx, int ip)
```

Model-specific implementation of fsz.

#### Parameters

- **sz** – Sensitivity of rz, total derivative
- **ie** – event index
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **sx** – current state sensitivity
- **ip** – sensitivity index

```
virtual void fw(realtype *w, const realtype t, realtype const *x, realtype const *p, realtype const *k, realtype  
               const *h, realtype const *tcl, realtype const *spl)
```

Model-specific implementation of fw.

#### Parameters

- **w** – Recurring terms in xdot
- **t** – timepoint
- **x** – vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **tcl** – total abundances for conservation laws
- **spl** – spline value vector

```
virtual void fx0(realtype *x0, const realtype t, realtype const *p, realtype const *k)
```

Model-specific implementation of fx0.

#### Parameters

- **x0** – initial state
- **t** – initial time
- **p** – parameter vector
- **k** – constant vector

```
virtual void fx0_fixedParameters(realtype *x0, const realtype t, realtype const *p, realtype const *k,  
                                gsl::span<int const> reinitialization_state_idx)
```

Model-specific implementation of fx0\_fixedParameters.

#### Parameters

- **x0** – initial state
- **t** – initial time

- **p** – parameter vector
- **k** – constant vector
- **reinitialization\_state\_idxs** – Indices of states to be reinitialized based on provided constants / fixed parameters.

virtual void **fy**(*realtype* \*y, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w)

Model-specific implementation of fy.

#### Parameters

- **y** – model output at current timepoint
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector
- **w** – repeating elements vector

virtual void **fz**(*realtype* \*z, int ie, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h)

Model-specific implementation of fz.

#### Parameters

- **z** – value of event output
- **ie** – event index
- **t** – current time
- **x** – current state
- **p** – parameter vector
- **k** – constant vector
- **h** – Heaviside vector

## Public Members

bool **pythonGenerated**

Flag indicating Matlab- or Python-based model generation

*SecondOrderMode* **o2mode** = {*SecondOrderMode::none*}

Flag indicating whether for amici::Solver::sensi\_ == amici::SensitivityOrder::second directional or full second order derivative will be computed

std::vector<*realtype*> **idlist**

Flag array for DAE equations

*Logger* \*logger = nullptr  
*Logger*

## Protected Functions

void **writeSliceEvent**(gsl::span<*realtype* const> slice, gsl::span<*realtype*> buffer, int const ie)

Write part of a slice to a buffer according to indices specified in z2event.

### Parameters

- **slice** – Input data slice
- **buffer** – Output data slice
- **ie** – Event index

void **writeSensitivitySliceEvent**(gsl::span<*realtype* const> slice, gsl::span<*realtype*> buffer, int const ie)

Write part of a sensitivity slice to a buffer according to indices specified in z2event.

### Parameters

- **slice** – source data slice
- **buffer** – output data slice
- **ie** – event index

void **writeLLHSensitivitySlice**(std::vector<*realtype*> const &dLLhdp, std::vector<*realtype*> &s1lh, std::vector<*realtype*> &s2llh)

Separate first and second order objective sensitivity information and write them into the respective buffers.

### Parameters

- **dLLhdp** – Data with mangled first- and second-order information
- **s1lh** – First order buffer
- **s2llh** – Second order buffer

void **checkLLHBufferSize**(std::vector<*realtype*> const &s1lh, std::vector<*realtype*> const &s2llh) const

Verify that the provided buffers have the expected size.

### Parameters

- **s1lh** – first order buffer
- **s2llh** – second order buffer

void **initializeVectors**()

Set the nplist-dependent vectors to their proper sizes.

void **fy**(*realtype* t, *AmiVector* const &x)

Compute observables / measurements.

### Parameters

- **t** – Current timepoint
- **x** – Current state

void **fdydp**(*realtype* t, *AmiVector* const &x)

Compute partial derivative of observables  $y$  w.r.t. model parameters  $p$ .

**Parameters**

- **t** – Current timepoint
- **x** – Current state

void **fdydx**(*realtype* t, *AmiVector* const &x)

Compute partial derivative of observables  $y$  w.r.t. state variables  $x$ .

**Parameters**

- **t** – Current timepoint
- **x** – Current state

void **fsigmay**(int it, *ExpData* const \*edata)

Compute standard deviation of measurements.

**Parameters**

- **it** – Timepoint index
- **edata** – Experimental data

void **fdsigmaydp**(int it, *ExpData* const \*edata)

Compute partial derivative of standard deviation of measurements w.r.t. model parameters.

**Parameters**

- **it** – Timepoint index
- **edata** – pointer to `amici::ExpData` data instance holding sigma values

void **fdsigmaydy**(int it, *ExpData* const \*edata)

Compute partial derivative of standard deviation of measurements w.r.t. model outputs.

**Parameters**

- **it** – Timepoint index
- **edata** – pointer to `amici::ExpData` data instance holding sigma values

void **fJy**(*realtype* &Jy, int it, *AmiVector* const &y, *ExpData* const &edata)

Compute negative log-likelihood of measurements  $y$ .

**Parameters**

- **Jy** – Variable to which llh will be added
- **it** – Timepoint index
- **y** – Simulated observable
- **edata** – Pointer to experimental data instance

void **fdJydy**(int it, *AmiVector* const &x, *ExpData* const &edata)

Compute partial derivative of time-resolved measurement negative log-likelihood  $Jy$ .

**Parameters**

- **it** – timepoint index
- **x** – state variables

- **edata** – Pointer to experimental data

void **fdJydsigma**(int it, *AmiVector* const &x, *ExpData* const &edata)

Sensitivity of time-resolved measurement negative log-likelihood  $J_y$  w.r.t. standard deviation sigma.

**Parameters**

- **it** – timepoint index
- **x** – state variables
- **edata** – pointer to experimental data instance

void **fdJydp**(int const it, *AmiVector* const &x, *ExpData* const &edata)

Compute sensitivity of time-resolved measurement negative log-likelihood  $J_y$  w.r.t. parameters for the given timepoint.

**Parameters**

- **it** – timepoint index
- **x** – state variables
- **edata** – pointer to experimental data instance

void **fdJydx**(int const it, *AmiVector* const &x, *ExpData* const &edata)

Sensitivity of time-resolved measurement negative log-likelihood  $J_y$  w.r.t. state variables.

**Parameters**

- **it** – Timepoint index
- **x** – State variables
- **edata** – Pointer to experimental data instance

void **fz**(int ie, *realtype* t, *AmiVector* const &x)

Compute event-resolved output.

**Parameters**

- **ie** – Event index
- **t** – Current timepoint
- **x** – Current state

void **fdzdp**(int ie, *realtype* t, *AmiVector* const &x)

Compute partial derivative of event-resolved output z w.r.t. model parameters p

**Parameters**

- **ie** – event index
- **t** – current timepoint
- **x** – current state

void **fdzdx**(int ie, *realtype* t, *AmiVector* const &x)

Compute partial derivative of event-resolved output z w.r.t. model states x.

**Parameters**

- **ie** – Event index
- **t** – Current timepoint
- **x** – Current state



void **frz**(int ie, *realtype* t, *AmiVector* const &x)

Compute event root function of events.

Equal to *Model::froot* but does not include non-output events.

#### Parameters

- **ie** – Event index
- **t** – Current timepoint
- **x** – Current state

void **fdrzdp**(int ie, *realtype* t, *AmiVector* const &x)

Compute sensitivity of event-resolved root output w.r.t. model parameters **p**.

#### Parameters

- **ie** – Event index
- **t** – Current timepoint
- **x** – Current state

void **fdrzdx**(int ie, *realtype* t, *AmiVector* const &x)

Compute sensitivity of event-resolved measurements *rz* w.r.t. model states **x**.

#### Parameters

- **ie** – Event index
- **t** – Current timepoint
- **x** – Current state

void **fsmgma**(int const ie, int const nroots, const *realtype* t, *ExpData* const \*edata)

Compute standard deviation of events.

#### Parameters

- **ie** – Event index
- **nroots** – Event index
- **t** – Current timepoint
- **edata** – Experimental data

void **fdsigmazdp**(int ie, int nroots, *realtype* t, *ExpData* const \*edata)

Compute sensitivity of standard deviation of events measurements w.r.t. model parameters **p**.

#### Parameters

- **ie** – Event index
- **nroots** – Event occurrence
- **t** – Current timepoint
- **edata** – Pointer to experimental data instance

void **fJz**(*realtype* &Jz, int nroots, *AmiVector* const &z, *ExpData* const &edata)

Compute negative log-likelihood of event-resolved measurements **z**.

#### Parameters

- **Jz** – Variable to which llh will be added

- **nroots** – Event index
- **z** – Simulated event
- **edata** – Experimental data

void **fdJzdz**(int const ie, int const nroots, const *realtype* t, *AmiVector* const &x, *ExpData* const &edata)

Compute partial derivative of event measurement negative log-likelihood  $J_z$ .

**Parameters**

- **ie** – Event index
- **nroots** – Event index
- **t** – Current timepoint
- **x** – State variables
- **edata** – Experimental data

void **fdJzdsigma**(int const ie, int const nroots, const *realtype* t, *AmiVector* const &x, *ExpData* const &edata)

Compute sensitivity of event measurement negative log-likelihood  $J_z$  w.r.t. standard deviation  $\sigma_{maz}$ .

**Parameters**

- **ie** – Event index
- **nroots** – Event index
- **t** – Current timepoint
- **x** – State variables
- **edata** – Pointer to experimental data instance

void **fdJzdp**(int const ie, int const nroots, *realtype* t, *AmiVector* const &x, *ExpData* const &edata)

Compute sensitivity of event-resolved measurement negative log-likelihood  $J_z$  w.r.t. parameters.

**Parameters**

- **ie** – Event index
- **nroots** – Event index
- **t** – Current timepoint
- **x** – State variables
- **edata** – Pointer to experimental data instance

void **fdJzdx**(int const ie, int const nroots, *realtype* t, *AmiVector* const &x, *ExpData* const &edata)

Compute sensitivity of event-resolved measurement negative log-likelihood  $J_z$  w.r.t. state variables.

**Parameters**

- **ie** – Event index
- **nroots** – Event index
- **t** – Current timepoint
- **x** – State variables
- **edata** – Experimental data

void **fJrz**(*realtype* &Jrz, int nroots, *AmiVector* const &rz, *ExpData* const &edata)

Compute regularization of negative log-likelihood with roots of event-resolved measurements rz.

**Parameters**

- **Jrz** – Variable to which regularization will be added
- **nroots** – Event index
- **rz** – Regularization variable
- **edata** – Experimental data

void **fdJrzdz**(int const ie, int const nroots, const *realtype* t, *AmiVector* const &x, *ExpData* const &edata)

Compute partial derivative of event measurement negative log-likelihood J.

**Parameters**

- **ie** – Event index
- **nroots** – Event index
- **t** – Current timepoint
- **x** – State variables
- **edata** – Experimental data

void **fdJrzdsigma**(int const ie, int const nroots, const *realtype* t, *AmiVector* const &x, *ExpData* const &edata)

Compute sensitivity of event measurement negative log-likelihood Jz w.r.t. standard deviation sigma<sub>z</sub>.

**Parameters**

- **ie** – event index
- **nroots** – event index
- **t** – current timepoint
- **x** – state variables
- **edata** – pointer to experimental data instance

void **fspl**(*realtype* t)

Spline functions.

**Parameters** **t** – timepoint

void **fsspl**(*realtype* t)

Parametric derivatives of splines functions.

**Parameters** **t** – timepoint

void **fw**(*realtype* t, *realtype* const \*x)

Compute recurring terms in xdot.

**Parameters**

- **t** – Timepoint
- **x** – Array with the states

void **fdwdp**(*realtype* t, *realtype* const \*x)

Compute parameter derivative for recurring terms in xdot.

**Parameters**

- **t** – Timepoint
- **x** – Array with the states

void **fdwdx**(*realtype* t, *realtype* const \*x)

Compute state derivative for recurring terms in xdot.

**Parameters**

- **t** – Timepoint
- **x** – Array with the states

void **fdwdw**(*realtype* t, *realtype* const \*x)

Compute self derivative for recurring terms in xdot.

**Parameters**

- **t** – Timepoint
- **x** – Array with the states

virtual void **fx\_rdata**(*realtype* \*x\_rdata, *realtype* const \*x\_solver, *realtype* const \*tcl, *realtype* const \*p, *realtype* const \*k)

Compute fx\_rdata.

To be implemented by derived class if applicable.

**Parameters**

- **x\_rdata** – State variables with conservation laws expanded
- **x\_solver** – State variables with conservation laws applied
- **tcl** – Total abundances for conservation laws
- **p** – parameter vector
- **k** – constant vector

virtual void **fsx\_rdata**(*realtype* \*sx\_rdata, *realtype* const \*sx\_solver, *realtype* const \*stcl, *realtype* const \*p, *realtype* const \*k, *realtype* const \*x\_solver, *realtype* const \*tcl, int const ip)

Compute fsx\_solver.

To be implemented by derived class if applicable.

**Parameters**

- **sx\_rdata** – State sensitivity variables with conservation laws expanded
- **sx\_solver** – State sensitivity variables with conservation laws applied
- **stcl** – Sensitivities of total abundances for conservation laws
- **p** – parameter vector
- **k** – constant vector
- **x\_solver** – State variables with conservation laws applied
- **tcl** – Total abundances for conservation laws
- **ip** – Sensitivity index

virtual void **fx\_solver**(*realtype* \*x\_solver, *realtype* const \*x\_rdata)

Compute fx\_solver.

To be implemented by derived class if applicable.

#### Parameters

- **x\_solver** – State variables with conservation laws applied
- **x\_rdata** – State variables with conservation laws expanded

virtual void **fsx\_solver**(*realtype* \*sx\_solver, *realtype* const \*sx\_rdata)

Compute fsx\_solver.

To be implemented by derived class if applicable.

#### Parameters

- **sx\_rdata** – State sensitivity variables with conservation laws expanded
- **sx\_solver** – State sensitivity variables with conservation laws applied

virtual void **ftotal\_cl**(*realtype* \*total\_cl, *realtype* const \*x\_rdata, *realtype* const \*p, *realtype* const \*k)

Compute ftotal\_cl.

To be implemented by derived class if applicable.

#### Parameters

- **total\_cl** – Total abundances of conservation laws
- **x\_rdata** – State variables with conservation laws expanded
- **p** – parameter vector
- **k** – constant vector

virtual void **fstotal\_cl**(*realtype* \*stotal\_cl, *realtype* const \*sx\_rdata, int const ip, *realtype* const \*x\_rdata, *realtype* const \*p, *realtype* const \*k, *realtype* const \*tcl)

Compute fstotal\_cl.

To be implemented by derived class if applicable.

#### Parameters

- **stotal\_cl** – Sensitivities for the total abundances of conservation laws
- **sx\_rdata** – State sensitivity variables with conservation laws expanded
- **ip** – Sensitivity index
- **x\_rdata** – State variables with conservation laws expanded
- **p** – parameter vector
- **k** – constant vector
- **tcl** – Total abundances for conservation laws

*const\_N\_Vector* **computeX\_pos**(*const\_N\_Vector* x)

Compute non-negative state vector.

Compute non-negative state vector according to stateIsNonNegative. If anyStateNonNegative is set to false, i.e., all entries in stateIsNonNegative are false, this function directly returns x, otherwise all entries of x are copied in to amici::Model::x\_pos\_tmp\_ and negative values are replaced by 0 where applicable.

**Parameters** **x** – State vector possibly containing negative values

**Returns** State vector with negative values replaced by 0 according to `stateIsNonNegative`

*realtype* const \***computeX\_pos**(*AmiVector* const &x)

Compute non-negative state vector.

Compute non-negative state vector according to `stateIsNonNegative`. If `anyStateNonNegative` is set to `false`, i.e., all entries in `stateIsNonNegative` are `false`, this function directly returns **x**, otherwise all entries of **x** are copied in to `amici::Model::x_pos_tmp_` and negative values are replaced by 0 where applicable.

**Parameters** **x** – State vector possibly containing negative values

**Returns** State vector with negative values replaced by 0 according to `stateIsNonNegative`

## Protected Attributes

*ModelState* **state\_**

All variables necessary for function evaluation

*ModelStateDerived* **derived\_state\_**

Storage for model quantities beyond *ModelState* for the current timepoint

std::vector<*HermiteSpline*> **splines\_**

Storage for splines of the model

std::vector<int> **z2event\_**

index indicating to which event an event output belongs

std::vector<*realtype*> **x0data\_**

state initialization (size `nx_solver`)

std::vector<*realtype*> **sx0data\_**

sensitivity initialization (size `nx_rdata` x `nplist`, row-major)

std::vector<bool> **state\_is\_non\_negative\_**

vector of bools indicating whether state variables are to be assumed to be positive

std::vector<bool> **root\_initial\_values\_**

Vector of booleans indicating the initial boolean value for every event trigger function. Events at `t0` can only trigger if the initial value is set to `false`. Must be specified during model compilation by setting the `initialValue` attribute of an event trigger.

bool **any\_state\_non\_negative\_** = {false}

boolean indicating whether any entry in `stateIsNonNegative` is `true`

int **nmaxevent\_** = {10}

maximal number of events to track

*SteadyStateComputationMode*

**steadystate\_computation\_mode\_**{*SteadyStateComputationMode::integrateIfNewtonFails*}

method for steady-state computation

*SteadyStateSensitivityMode*

**steadystate\_sensitivity\_mode\_**{*SteadyStateSensitivityMode::integrateIfNewtonFails*}

method for steadystate sensitivities computation

bool **always\_check\_finite\_** = {false}

Indicates whether the result of every call to `Model::f*` should be checked for finiteness

bool **sigma\_res\_** = {false}

indicates whether sigma residuals are to be added for every datapoint

*realtype* **min\_sigma\_** = {50.0}

offset to ensure positivity of sigma residuals, only has an effect when `sigma_res_` is true

## Friends

template<class **Archive**>

friend void **serialize**(*Archive* &ar, *Model* &m, unsigned int version)

Serialize *Model* (see `boost::serialization::serialize`).

### Parameters

- **ar** – Archive to serialize to
- **m** – Data to serialize
- **version** – Version number

friend bool **operator==**(*Model* const &a, *Model* const &b)

Check equality of data members.

### Parameters

- **a** – First model instance
- **b** – Second model instance

**Returns** Equality

## Class `Model_DAE`

- Defined in `file_include_amici_model_dae.h`

## Inheritance Relationships

### Base Type

- `public amici::Model` (*Class Model*)

### Class Documentation

class **Model\_DAE** : public amici::*Model*

The *Model* class represents an AMICI DAE model.

The model does not contain any data, but represents the state of the model at a specific time *t*. The states must not always be in sync, but may be updated asynchronously.

### Public Functions

**Model\_DAE**() = default

default constructor

inline **Model\_DAE**(*ModelDimensions* const &model\_dimensions, *SimulationParameters* simulation\_parameters, const *SecondOrderMode* o2mode, std::vector<*realtype*> const &idlist, std::vector<int> const &z2event, bool const pythonGenerated = false, int const ndxdotdp\_explicit = 0, int const ndxdotdx\_explicit = 0, int const w\_recursion\_depth = 0)

Constructor with model dimensions.

#### Parameters

- **model\_dimensions** – *Model* dimensions
- **simulation\_parameters** – Simulation parameters
- **o2mode** – second order sensitivity mode
- **idlist** – indexes indicating algebraic components (DAE only)
- **z2event** – mapping of event outputs to events
- **pythonGenerated** – flag indicating matlab or python wrapping
- **ndxdotdp\_explicit** – number of nonzero elements dxdotdp\_explicit
- **ndxdotdx\_explicit** – number of nonzero elements dxdotdx\_explicit
- **w\_recursion\_depth** – Recursion depth of fw

virtual void **fJ**(*realtype* t, *realtype* cj, *AmiVector* const &x, *AmiVector* const &dx, *AmiVector* const &xdot, SUNMatrix J) override

Dense Jacobian function.

#### Parameters

- **t** – time
- **cj** – scaling factor (inverse of timestep, DAE only)
- **x** – state
- **dx** – time derivative of state (DAE only)



- **xdot** – values of residual function (unused)
- **J** – dense matrix to which values of the jacobian will be written

void **fJ**(*realtype* t, *realtype* cj, *const\_N\_Vector* x, *const\_N\_Vector* dx, *const\_N\_Vector* xdot, SUNMatrix J)

Jacobian of xdot with respect to states x.

#### Parameters

- **t** – timepoint
- **cj** – scaling factor, inverse of the step size
- **x** – Vector with the states
- **dx** – Vector with the derivative states
- **xdot** – Vector with the right hand side
- **J** – Matrix to which the Jacobian will be written

virtual void **fJB**(const *realtype* t, *realtype* cj, *AmiVector* const &x, *AmiVector* const &dx, *AmiVector* const &xB, *AmiVector* const &dxB, *AmiVector* const &xBdot, SUNMatrix JB) override

Dense Jacobian function.

#### Parameters

- **t** – time
- **cj** – scaling factor (inverse of timestep, DAE only)
- **x** – state
- **dx** – time derivative of state (DAE only)
- **xB** – Vector with the adjoint states
- **dxB** – Vector with the adjoint derivative states
- **xBdot** – Vector with the adjoint right hand side (unused)
- **JB** – dense matrix to which values of the jacobian will be written

void **fJB**(*realtype* t, *realtype* cj, *const\_N\_Vector* x, *const\_N\_Vector* dx, *const\_N\_Vector* xB, *const\_N\_Vector* dxB, SUNMatrix JB)

Jacobian of xBdot with respect to adjoint state xB.

#### Parameters

- **t** – timepoint
- **cj** – scaling factor, inverse of the step size
- **x** – Vector with the states
- **dx** – Vector with the derivative states
- **xB** – Vector with the adjoint states
- **dxB** – Vector with the adjoint derivative states
- **JB** – Matrix to which the Jacobian will be written

virtual void **fJSparse**(*realtype* t, *realtype* cj, *AmiVector* const &x, *AmiVector* const &dx, *AmiVector* const &xdot, SUNMatrix J) override

Sparse Jacobian function.

#### Parameters

- **t** – time
- **cj** – scaling factor (inverse of timestep, DAE only)
- **x** – state
- **dx** – time derivative of state (DAE only)
- **xdot** – values of residual function (unused)
- **J** – sparse matrix to which values of the Jacobian will be written

void **fJSparse**(*realtype* t, *realtype* cj, *const\_N\_Vector* x, *const\_N\_Vector* dx, SUNMatrix J)

J in sparse form (for sparse solvers from the SuiteSparse Package)

#### Parameters

- **t** – timepoint
- **cj** – scalar in Jacobian (inverse stepsize)
- **x** – Vector with the states
- **dx** – Vector with the derivative states
- **J** – Matrix to which the Jacobian will be written

virtual void **fJSparseB**(const *realtype* t, *realtype* cj, *AmiVector* const &x, *AmiVector* const &dx, *AmiVector* const &xB, *AmiVector* const &dxB, *AmiVector* const &xBdot, SUNMatrix JB)  
override

Sparse Jacobian function.

#### Parameters

- **t** – time
- **cj** – scaling factor (inverse of timestep, DAE only)
- **x** – state
- **dx** – time derivative of state (DAE only)
- **xB** – Vector with the adjoint states
- **dxB** – Vector with the adjoint derivative states
- **xBdot** – Vector with the adjoint right hand side (unused)
- **JB** – dense matrix to which values of the jacobian will be written

void **fJSparseB**(*realtype* t, *realtype* cj, *const\_N\_Vector* x, *const\_N\_Vector* dx, *const\_N\_Vector* xB, *const\_N\_Vector* dxB, SUNMatrix JB)

JB in sparse form (for sparse solvers from the SuiteSparse Package)

#### Parameters

- **t** – timepoint
- **cj** – scalar in Jacobian
- **x** – Vector with the states
- **dx** – Vector with the derivative states
- **xB** – Vector with the adjoint states
- **dxB** – Vector with the adjoint derivative states
- **JB** – Matrix to which the Jacobian will be written

virtual void **fJDiag**(*realtype* t, *AmiVector* &JDiag, *realtype* cj, *AmiVector* const &x, *AmiVector* const &dx)  
override

Diagonal of the Jacobian (for preconditioning)

#### Parameters

- **t** – timepoint
- **JDiag** – Vector to which the Jacobian diagonal will be written
- **cj** – scaling factor, inverse of the step size
- **x** – Vector with the states
- **dx** – Vector with the derivative states

virtual void **fJv**(*realtype* t, *AmiVector* const &x, *AmiVector* const &dx, *AmiVector* const &xdot, *AmiVector* const &v, *AmiVector* &nJv, *realtype* cj) override

Jacobian multiply function.

#### Parameters

- **t** – time
- **x** – state
- **dx** – time derivative of state (DAE only)
- **xdot** – values of residual function (unused)
- **v** – multiplication vector (unused)
- **nJv** – array to which result of multiplication will be written
- **cj** – scaling factor (inverse of timestep, DAE only)

void **fJv**(*realtype* t, *const\_N\_Vector* x, *const\_N\_Vector* dx, *const\_N\_Vector* v, *N\_Vector* Jv, *realtype* cj)

Matrix vector product of J with a vector v (for iterative solvers)

#### Parameters

- **t** – timepoint
- **cj** – scaling factor, inverse of the step size
- **x** – Vector with the states
- **dx** – Vector with the derivative states
- **v** – Vector with which the Jacobian is multiplied
- **Jv** – Vector to which the Jacobian vector product will be written

void **fJvB**(*realtype* t, *const\_N\_Vector* x, *const\_N\_Vector* dx, *const\_N\_Vector* xB, *const\_N\_Vector* dxB, *const\_N\_Vector* vB, *N\_Vector* JvB, *realtype* cj)

Matrix vector product of JB with a vector v (for iterative solvers)

#### Parameters

- **t** – timepoint
- **x** – Vector with the states
- **dx** – Vector with the derivative states
- **xB** – Vector with the adjoint states
- **dxB** – Vector with the adjoint derivative states

- **vB** – Vector with which the Jacobian is multiplied
- **JvB** – Vector to which the Jacobian vector product will be written
- **cj** – scalar in Jacobian (inverse stepsize)

virtual void **froot**(*realtype* t, *AmiVector* const &x, *AmiVector* const &dx, gsl::span<*realtype*> root) override  
Root function.

#### Parameters

- **t** – time
- **x** – state
- **dx** – time derivative of state (DAE only)
- **root** – array to which values of the root function will be written

void **froot**(*realtype* t, *const\_N\_Vector* x, *const\_N\_Vector* dx, gsl::span<*realtype*> root)  
Event trigger function for events.

#### Parameters

- **t** – timepoint
- **x** – Vector with the states
- **dx** – Vector with the derivative states
- **root** – array with root function values

virtual void **fxdot**(*realtype* t, *AmiVector* const &x, *AmiVector* const &dx, *AmiVector* &xdot) override  
Residual function.

#### Parameters

- **t** – time
- **x** – state
- **dx** – time derivative of state (DAE only)
- **xdot** – array to which values of the residual function will be written

void **fxdot**(*realtype* t, *const\_N\_Vector* x, *const\_N\_Vector* dx, *N\_Vector* xdot)  
Residual function of the DAE.

#### Parameters

- **t** – timepoint
- **x** – Vector with the states
- **dx** – Vector with the derivative states
- **xdot** – Vector with the right hand side

void **fxBdot**(*realtype* t, *const\_N\_Vector* x, *const\_N\_Vector* dx, *const\_N\_Vector* xB, *const\_N\_Vector* dxB, *N\_Vector* xBdot)

Right hand side of differential equation for adjoint state xB.

#### Parameters

- **t** – timepoint
- **x** – Vector with the states

- **dx** – Vector with the derivative states
- **xB** – Vector with the adjoint states
- **dxB** – Vector with the adjoint derivative states
- **xBdot** – Vector with the adjoint right hand side

void **fqBdot**(*realtype* t, *const\_N\_Vector* x, *const\_N\_Vector* dx, *const\_N\_Vector* xB, *const\_N\_Vector* dxB, *N\_Vector* qBdot)

Right hand side of integral equation for quadrature states qB.

#### Parameters

- **t** – timepoint
- **x** – Vector with the states
- **dx** – Vector with the derivative states
- **xB** – Vector with the adjoint states
- **dxB** – Vector with the adjoint derivative states
- **qBdot** – Vector with the adjoint quadrature right hand side

virtual void **fxBdot\_ss**(const *realtype* t, *AmiVector* const &xB, *AmiVector* const &dxB, *AmiVector* &xBdot) override

Residual function backward when running in steady state mode.

#### Parameters

- **t** – time
- **xB** – adjoint state
- **dxB** – time derivative of state (DAE only)
- **xBdot** – array to which values of the residual function will be written

void **fxBdot\_ss**(*realtype* t, *const\_N\_Vector* xB, *const\_N\_Vector* dxB, *N\_Vector* xBdot) const

Implementation of fxBdot for steady state case at the N\_Vector level.

#### Parameters

- **t** – timepoint
- **xB** – Vector with the adjoint state
- **dxB** – Vector with the adjoint derivative states
- **xBdot** – Vector with the adjoint right hand side

void **fqBdot\_ss**(*realtype* t, *const\_N\_Vector* xB, *const\_N\_Vector* dxB, *N\_Vector* qBdot) const

Implementation of fqBdot for steady state at the N\_Vector level.

#### Parameters

- **t** – timepoint
- **xB** – Vector with the adjoint states
- **dxB** – Vector with the adjoint derivative states
- **qBdot** – Vector with the adjoint quadrature right hand side

virtual void **fJSparseB\_ss**(SUNMatrix JB) override

Sparse Jacobian function backward, steady state case.

**Parameters** **JB** – sparse matrix to which values of the Jacobian will be written

virtual void **writeSteadystateJB**(const *realtype* t, *realtype* cj, *AmiVector* const &x, *AmiVector* const &dx, *AmiVector* const &xB, *AmiVector* const &dxB, *AmiVector* const &xBdot) override

Computes the sparse backward Jacobian for steadystate integration and writes it to the model member.

**Parameters**

- **t** – timepoint
- **cj** – scalar in Jacobian
- **x** – Vector with the states
- **dx** – Vector with the derivative states
- **xB** – Vector with the adjoint states
- **dxB** – Vector with the adjoint derivative states
- **xBdot** – Vector with the adjoint state right hand side

void **fdxdotdp**(*realtype* t, const *const\_N\_Vector* x, const *const\_N\_Vector* dx)

Sensitivity of dx/dt wrt model parameters p.

**Parameters**

- **t** – timepoint
- **x** – Vector with the states
- **dx** – Vector with the derivative states

inline virtual void **fdxdotdp**(const *realtype* t, *AmiVector* const &x, *AmiVector* const &dx) override

Model-specific sparse implementation of explicit parameter derivative of right hand side.

**Parameters**

- **t** – time
- **x** – state
- **dx** – time derivative of state (DAE only)

virtual void **fsxdot**(*realtype* t, *AmiVector* const &x, *AmiVector* const &dx, int ip, *AmiVector* const &sx, *AmiVector* const &sdx, *AmiVector* &sxdot) override

Sensitivity Residual function.

**Parameters**

- **t** – time
- **x** – state
- **dx** – time derivative of state (DAE only)
- **ip** – parameter index
- **sx** – sensitivity state
- **sdx** – time derivative of sensitivity state (DAE only)
- **sxdot** – array to which values of the sensitivity residual function will be written

void **fsxdot**(*realtype* t, *const\_N\_Vector* x, *const\_N\_Vector* dx, int ip, *const\_N\_Vector* sx, *const\_N\_Vector* sdx, *N\_Vector* sxdot)

Right hand side of differential equation for state sensitivities sx.

#### Parameters

- **t** – timepoint
- **x** – Vector with the states
- **dx** – Vector with the derivative states
- **ip** – parameter index
- **sx** – Vector with the state sensitivities
- **sdx** – Vector with the derivative state sensitivities
- **sxdot** – Vector with the sensitivity right hand side

void **fm**(*realtype* t, *const\_N\_Vector* x)

Mass matrix for DAE systems.

#### Parameters

- **t** – timepoint
- **x** – Vector with the states

virtual std::unique\_ptr<*Solver*> **getSolver**() override

Retrieves the solver object.

**Returns** The *Solver* instance

### Protected Functions

virtual void **fJSparse**(SUNMatrixContent\_Sparse JSparse, *realtype* t, *realtype* const \*x, double const \*p, double const \*k, *realtype* const \*h, *realtype* cj, *realtype* const \*dx, *realtype* const \*w, *realtype* const \*dwdx)

*Model* specific implementation for fJSparse.

#### Parameters

- **JSparse** – Matrix to which the Jacobian will be written
- **t** – timepoint
- **x** – Vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **cj** – scaling factor, inverse of the step size
- **dx** – Vector with the derivative states
- **w** – vector with helper variables
- **dwdx** – derivative of w wrt x

virtual void **froot**(*realtype* \*root, *realtype* t, *realtype* const \*x, double const \*p, double const \*k, *realtype* const \*h, *realtype* const \*dx)

*Model* specific implementation for froot.

#### Parameters

- **root** – values of the trigger function
- **t** – timepoint
- **x** – Vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **dx** – Vector with the derivative states

virtual void **fxdot**(*realtype* \*xdot, *realtype* t, *realtype* const \*x, double const \*p, double const \*k, *realtype* const \*h, *realtype* const \*dx, *realtype* const \*w) = 0

*Model* specific implementation for fxdot.

#### Parameters

- **xdot** – residual function
- **t** – timepoint
- **x** – Vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **w** – vector with helper variables
- **dx** – Vector with the derivative states

virtual void **fdxdotdp**(*realtype* \*dxdotdp, *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, int ip, *realtype* const \*dx, *realtype* const \*w, *realtype* const \*dwdp)

*Model* specific implementation of fdxdotdp.

#### Parameters

- **dxdotdp** – partial derivative xdot wrt p
- **t** – timepoint
- **x** – Vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **ip** – parameter index
- **dx** – Vector with the derivative states
- **w** – vector with helper variables
- **dwdp** – derivative of w wrt p



virtual void **fdxdotdp\_explicit**(*realtype* \*dxdotdp\_explicit, *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*dx, *realtype* const \*w)

*Model* specific implementation of fdxdotdp\_explicit, no w chainrule (Py)

#### Parameters

- **dxdotdp\_explicit** – partial derivative xdot wrt p
- **t** – timepoint
- **x** – Vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **dx** – Vector with the derivative states
- **w** – vector with helper variables

virtual void **fdxdotdp\_explicit\_colptrs**(*SUNMatrixWrapper* &dxdotdp)

*Model* specific implementation of fdxdotdp\_explicit, colptrs part.

**Parameters** **dxdotdp** – sparse matrix to which colptrs will be written

virtual void **fdxdotdp\_explicit\_rowvals**(*SUNMatrixWrapper* &dxdotdp)

*Model* specific implementation of fdxdotdp\_explicit, rowvals part.

**Parameters** **dxdotdp** – sparse matrix to which rowvals will be written

virtual void **fdxdotdx\_explicit**(*realtype* \*dxdotdx\_explicit, *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*dx, *realtype* const \*w)

*Model* specific implementation of fdxdotdx\_explicit, no w chainrule (Py)

#### Parameters

- **dxdotdx\_explicit** – partial derivative xdot wrt x
- **t** – timepoint
- **x** – Vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – heavyside vector
- **dx** – Vector with the derivative states
- **w** – vector with helper variables

virtual void **fdxdotdx\_explicit\_colptrs**(*SUNMatrixWrapper* &dxdotdx)

*Model* specific implementation of fdxdotdx\_explicit, colptrs part.

**Parameters** **dxdotdx** – sparse matrix to which colptrs will be written

virtual void **fdxdotdx\_explicit\_rowvals**(*SUNMatrixWrapper* &dxdotdx)

*Model* specific implementation of fdxdotdx\_explicit, rowvals part.

**Parameters** **dxdotdx** – sparse matrix to which rowvals will be written

virtual void **fdxdotdw**(*realtype* \*dxdotdw, *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*dx, *realtype* const \*w)

*Model* specific implementation of fdxdotdw, data part.

#### Parameters

- **dxdotdw** – partial derivative xdot wrt w
- **t** – timepoint
- **x** – Vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **dx** – Vector with the derivative states
- **w** – vector with helper variables

virtual void **fdxdotdw\_colptrs**(*SUNMatrixWrapper* &dxdotdw)

*Model* specific implementation of fdxdotdw, colptrs part.

**Parameters** **dxdotdw** – sparse matrix to which colptrs will be written

virtual void **fdxdotdw\_rowvals**(*SUNMatrixWrapper* &dxdotdw)

*Model* specific implementation of fdxdotdw, rowvals part.

**Parameters** **dxdotdw** – sparse matrix to which rowvals will be written

void **fdxdotdw**(*realtype* t, *const\_N\_Vector* x, *const\_N\_Vector* dx)

Sensitivity of dx/dt wrt model parameters w.

#### Parameters

- **t** – timepoint
- **x** – Vector with the states
- **dx** – Vector with the derivative states

virtual void **fM**(*realtype* \*M, const *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k)

*Model* specific implementation of fM.

#### Parameters

- **M** – mass matrix
- **t** – timepoint
- **x** – Vector with the states
- **p** – parameter vector
- **k** – constants vector

## Class `Model_ODE`

- Defined in file `include_amici_model_ode.h`

## Inheritance Relationships

### Base Type

- `public amici::Model` (*Class Model*)

## Class Documentation

class **Model\_ODE** : public `amici::Model`

The *Model* class represents an AMICI ODE model.

The model does not contain any data, but represents the state of the model at a specific time *t*. The states must not always be in sync, but may be updated asynchronously.

### Public Functions

**Model\_ODE**() = default

default constructor

```
inline Model_ODE(ModelDimensions const &model_dimensions, SimulationParameters
simulation_parameters, const SecondOrderMode o2mode, std::vector<realtype> const
&idlist, std::vector<int> const &z2event, bool const pythonGenerated = false, int const
ndxdotdp_explicit = 0, int const ndxdotdx_explicit = 0, int const w_recursion_depth = 0)
```

Constructor with model dimensions.

#### Parameters

- **model\_dimensions** – *Model* dimensions
- **simulation\_parameters** – Simulation parameters
- **o2mode** – second order sensitivity mode
- **idlist** – indexes indicating algebraic components (DAE only)
- **z2event** – mapping of event outputs to events
- **pythonGenerated** – flag indicating matlab or python wrapping
- **ndxdotdp\_explicit** – number of nonzero elements `dxdotdp_explicit`
- **ndxdotdx\_explicit** – number of nonzero elements `dxdotdx_explicit`
- **w\_recursion\_depth** – Recursion depth of fw

```
virtual void fJ(realtype t, realtype cj, AmiVector const &x, AmiVector const &dx, AmiVector const &xdot,
SUNMatrix J) override
```

Dense Jacobian function.

#### Parameters

- **t** – time

- **cj** – scaling factor (inverse of timestep, DAE only)
- **x** – state
- **dx** – time derivative of state (DAE only)
- **xdot** – values of residual function (unused)
- **J** – dense matrix to which values of the jacobian will be written

void **fJ**(*realtype* t, *const\_N\_Vector* x, *const\_N\_Vector* xdot, SUNMatrix J)

Implementation of fJ at the N\_Vector level.

This function provides an interface to the model specific routines for the solver implementation as well as the *AmiVector* level implementation

#### Parameters

- **t** – timepoint
- **x** – Vector with the states
- **xdot** – Vector with the right hand side
- **J** – Matrix to which the Jacobian will be written

virtual void **fJB**(*const realtype* t, *realtype* cj, *AmiVector* const &x, *AmiVector* const &dx, *AmiVector* const &xB, *AmiVector* const &dxB, *AmiVector* const &xBdot, SUNMatrix JB) override

Dense Jacobian function.

#### Parameters

- **t** – time
- **cj** – scaling factor (inverse of timestep, DAE only)
- **x** – state
- **dx** – time derivative of state (DAE only)
- **xB** – Vector with the adjoint states
- **dxB** – Vector with the adjoint derivative states
- **xBdot** – Vector with the adjoint right hand side (unused)
- **JB** – dense matrix to which values of the jacobian will be written

void **fJB**(*realtype* t, *const\_N\_Vector* x, *const\_N\_Vector* xB, *const\_N\_Vector* xBdot, SUNMatrix JB)

Implementation of fJB at the N\_Vector level, this function provides an interface to the model specific routines for the solver implementation.

#### Parameters

- **t** – timepoint
- **x** – Vector with the states
- **xB** – Vector with the adjoint states
- **xBdot** – Vector with the adjoint right hand side
- **JB** – Matrix to which the Jacobian will be written

virtual void **fJSparse**(*realtype* t, *realtype* cj, *AmiVector* const &x, *AmiVector* const &dx, *AmiVector* const &xdot, SUNMatrix J) override

Sparse Jacobian function.

**Parameters**

- **t** – time
- **cj** – scaling factor (inverse of timestep, DAE only)
- **x** – state
- **dx** – time derivative of state (DAE only)
- **xdot** – values of residual function (unused)
- **J** – sparse matrix to which values of the Jacobian will be written

void **fJSparse**(*realtype* t, *const\_N\_Vector* x, SUNMatrix J)

Implementation of fJSparse at the N\_Vector level, this function provides an interface to the model specific routines for the solver implementation as well as the *AmiVector* level implementation.

**Parameters**

- **t** – timepoint
- **x** – Vector with the states
- **J** – Matrix to which the Jacobian will be written

virtual void **fJSparseB**(const *realtype* t, *realtype* cj, *AmiVector* const &x, *AmiVector* const &dx, *AmiVector* const &xB, *AmiVector* const &dxB, *AmiVector* const &xBdot, SUNMatrix JB)  
override

Sparse Jacobian function.

**Parameters**

- **t** – time
- **cj** – scaling factor (inverse of timestep, DAE only)
- **x** – state
- **dx** – time derivative of state (DAE only)
- **xB** – Vector with the adjoint states
- **dxB** – Vector with the adjoint derivative states
- **xBdot** – Vector with the adjoint right hand side (unused)
- **JB** – dense matrix to which values of the jacobian will be written

void **fJSparseB**(*realtype* t, *const\_N\_Vector* x, *const\_N\_Vector* xB, *const\_N\_Vector* xBdot, SUNMatrix JB)

Implementation of fJSparseB at the N\_Vector level, this function provides an interface to the model specific routines for the solver implementation.

**Parameters**

- **t** – timepoint
- **x** – Vector with the states
- **xB** – Vector with the adjoint states
- **xBdot** – Vector with the adjoint right hand side
- **JB** – Matrix to which the Jacobian will be written

void **fJDiag**(*realtype* t, N\_Vector JDiag, *const\_N\_Vector* x)

Implementation of fJDiag at the N\_Vector level, this function provides an interface to the model specific routines for the solver implementation.

**Parameters**

- **t** – timepoint
- **JDiag** – Vector to which the Jacobian diagonal will be written
- **x** – Vector with the states

virtual void **fJDiag**(*realtype* t, *AmiVector* &JDiag, *realtype* cj, *AmiVector* const &x, *AmiVector* const &dx)  
override

Diagonal of the Jacobian (for preconditioning)

**Parameters**

- **t** – timepoint
- **JDiag** – Vector to which the Jacobian diagonal will be written
- **cj** – scaling factor, inverse of the step size
- **x** – Vector with the states
- **dx** – Vector with the derivative states

virtual void **fJv**(*realtype* t, *AmiVector* const &x, *AmiVector* const &dx, *AmiVector* const &xdot, *AmiVector* const &v, *AmiVector* &nJv, *realtype* cj) override

Jacobian multiply function.

**Parameters**

- **t** – time
- **x** – state
- **dx** – time derivative of state (DAE only)
- **xdot** – values of residual function (unused)
- **v** – multiplication vector (unused)
- **nJv** – array to which result of multiplication will be written
- **cj** – scaling factor (inverse of timestep, DAE only)

void **fJv**(*const\_N\_Vector* v, N\_Vector Jv, *realtype* t, *const\_N\_Vector* x)

Implementation of fJv at the N\_Vector level.

**Parameters**

- **t** – timepoint
- **x** – Vector with the states
- **v** – Vector with which the Jacobian is multiplied
- **Jv** – Vector to which the Jacobian vector product will be written

void **fJvB**(*const\_N\_Vector* vB, N\_Vector JvB, *realtype* t, *const\_N\_Vector* x, *const\_N\_Vector* xB)

Implementation of fJvB at the N\_Vector level.

**Parameters**

- **t** – timepoint

- **x** – Vector with the states
- **xB** – Vector with the adjoint states
- **vB** – Vector with which the Jacobian is multiplied
- **JvB** – Vector to which the Jacobian vector product will be written

virtual void **froot**(*realtype* t, *AmiVector* const &x, *AmiVector* const &dx, gsl::span<*realtype*> root) override  
Root function.

#### Parameters

- **t** – time
- **x** – state
- **dx** – time derivative of state (DAE only)
- **root** – array to which values of the root function will be written

void **froot**(*realtype* t, *const\_N\_Vector* x, gsl::span<*realtype*> root)

Implementation of froot at the N\_Vector level This function provides an interface to the model specific routines for the solver implementation as well as the *AmiVector* level implementation.

#### Parameters

- **t** – timepoint
- **x** – Vector with the states
- **root** – array with root function values

virtual void **fxdot**(*realtype* t, *AmiVector* const &x, *AmiVector* const &dx, *AmiVector* &xdot) override  
Residual function.

#### Parameters

- **t** – time
- **x** – state
- **dx** – time derivative of state (DAE only)
- **xdot** – array to which values of the residual function will be written

void **fxdot**(*realtype* t, *const\_N\_Vector* x, N\_Vector xdot)

Implementation of fxdot at the N\_Vector level, this function provides an interface to the model specific routines for the solver implementation as well as the *AmiVector* level implementation.

#### Parameters

- **t** – timepoint
- **x** – Vector with the states
- **xdot** – Vector with the right hand side

void **fxBdot**(*realtype* t, N\_Vector x, N\_Vector xB, N\_Vector xBdot)

Implementation of fxBdot at the N\_Vector level.

#### Parameters

- **t** – timepoint
- **x** – Vector with the states
- **xB** – Vector with the adjoint states

- **xBdot** – Vector with the adjoint right hand side

void **fqBdot**(*realtype* t, *const\_N\_Vector* x, *const\_N\_Vector* xB, N\_Vector qBdot)

Implementation of fqBdot at the N\_Vector level.

#### Parameters

- **t** – timepoint
- **x** – Vector with the states
- **xB** – Vector with the adjoint states
- **qBdot** – Vector with the adjoint quadrature right hand side

virtual void **fxBdot\_ss**(const *realtype* t, *AmiVector* const &xB, *AmiVector* const&, *AmiVector* &xBdot)  
override

Residual function backward when running in steady state mode.

#### Parameters

- **t** – time
- **xB** – adjoint state
- **dxB** – time derivative of state (DAE only)
- **xBdot** – array to which values of the residual function will be written

void **fxBdot\_ss**(*realtype* t, *const\_N\_Vector* xB, N\_Vector xBdot) const

Implementation of fxBdot for steady state at the N\_Vector level.

#### Parameters

- **t** – timepoint
- **xB** – Vector with the states
- **xBdot** – Vector with the adjoint right hand side

void **fqBdot\_ss**(*realtype* t, N\_Vector xB, N\_Vector qBdot) const

Implementation of fqBdot for steady state case at the N\_Vector level.

#### Parameters

- **t** – timepoint
- **xB** – Vector with the adjoint states
- **qBdot** – Vector with the adjoint quadrature right hand side

virtual void **fJSparseB\_ss**(SUNMatrix JB) override

Sparse Jacobian function backward, steady state case.

**Parameters** **JB** – sparse matrix to which values of the Jacobian will be written

virtual void **writeSteadystateJB**(const *realtype* t, *realtype* cj, *AmiVector* const &x, *AmiVector* const &dx,  
*AmiVector* const &xB, *AmiVector* const &dxB, *AmiVector* const &xBdot)  
override

Computes the sparse backward Jacobian for steadystate integration and writes it to the model member.

#### Parameters

- **t** – timepoint
- **cj** – scalar in Jacobian



- **x** – Vector with the states
- **dx** – Vector with the derivative states
- **xB** – Vector with the adjoint states
- **dxB** – Vector with the adjoint derivative states
- **xBdot** – Vector with the adjoint state right hand side

virtual void **fsxdot**(*realtype* t, *AmiVector* const &x, *AmiVector* const &dx, int ip, *AmiVector* const &sx, *AmiVector* const &sdx, *AmiVector* &sxdot) override

Sensitivity Residual function.

#### Parameters

- **t** – time
- **x** – state
- **dx** – time derivative of state (DAE only)
- **ip** – parameter index
- **sx** – sensitivity state
- **sdx** – time derivative of sensitivity state (DAE only)
- **sxdot** – array to which values of the sensitivity residual function will be written

void **fsxdot**(*realtype* t, *const\_N\_Vector* x, int ip, *const\_N\_Vector* sx, *N\_Vector* sxdot)

Implementation of fsxdot at the N\_Vector level.

#### Parameters

- **t** – timepoint
- **x** – Vector with the states
- **ip** – parameter index
- **sx** – Vector with the state sensitivities
- **sxdot** – Vector with the sensitivity right hand side

virtual std::unique\_ptr<*Solver*> **getSolver**() override

Retrieves the solver object.

**Returns** The *Solver* instance

## Protected Functions

virtual void **fJSparse**(*SUNMatrixContent\_Sparse* JSparse, *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w, *realtype* const \*dwdx)

*Model* specific implementation for fJSparse (Matlab)

#### Parameters

- **JSparse** – Matrix to which the Jacobian will be written
- **t** – timepoint
- **x** – Vector with the states
- **p** – parameter vector

- **k** – constants vector
- **h** – Heaviside vector
- **w** – vector with helper variables
- **dwdx** – derivative of w wrt x

virtual void **fJSparse**(*realtype* \*JSparse, *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w, *realtype* const \*dwdx)

*Model* specific implementation for fJSparse, data only (Py)

#### Parameters

- **JSparse** – Matrix to which the Jacobian will be written
- **t** – timepoint
- **x** – Vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **w** – vector with helper variables
- **dwdx** – derivative of w wrt x

virtual void **fJSparse\_colptrs**(*SUNMatrixWrapper* &JSparse)

*Model* specific implementation for fJSparse, column pointers.

**Parameters** **JSparse** – sparse matrix to which colptrs will be written

virtual void **fJSparse\_rowvals**(*SUNMatrixWrapper* &JSparse)

*Model* specific implementation for fJSparse, row values.

**Parameters** **JSparse** – sparse matrix to which rowvals will be written

virtual void **froot**(*realtype* \*root, *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*tcl)

*Model* specific implementation for froot.

#### Parameters

- **root** – values of the trigger function
- **t** – timepoint
- **x** – Vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **tcl** – total abundances for conservation laws

virtual void **fxdot**(*realtype* \*xdot, *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w) = 0

*Model* specific implementation for fxdot.

#### Parameters

- **xdot** – residual function

- **t** – timepoint
- **x** – Vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **w** – vector with helper variables

virtual void **fdxdotdp**(*realtype* \*dxdotdp, *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, int ip, *realtype* const \*w, *realtype* const \*dwdp)

*Model* specific implementation of fdxdotdp, with w chainrule (Matlab)

#### Parameters

- **dxdotdp** – partial derivative xdot wrt p
- **t** – timepoint
- **x** – Vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **ip** – parameter index
- **w** – vector with helper variables
- **dwdp** – derivative of w wrt p

virtual void **fdxdotdp\_explicit**(*realtype* \*dxdotdp\_explicit, *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w)

*Model* specific implementation of fdxdotdp\_explicit, no w chainrule (Py)

#### Parameters

- **dxdotdp\_explicit** – partial derivative xdot wrt p
- **t** – timepoint
- **x** – Vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **w** – vector with helper variables

virtual void **fdxdotdp\_explicit\_colptrs**(*SUNMatrixWrapper* &dxdotdp)

*Model* specific implementation of fdxdotdp\_explicit, colptrs part.

**Parameters** **dxdotdp** – sparse matrix to which colptrs will be written

virtual void **fdxdotdp\_explicit\_rowvals**(*SUNMatrixWrapper* &dxdotdp)

*Model* specific implementation of fdxdotdp\_explicit, rowvals part.

**Parameters** **dxdotdp** – sparse matrix to which rowvals will be written

virtual void **fdxdotdx\_explicit**(*realtype* \*dxdotdx\_explicit, *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w)

*Model* specific implementation of fdxdotdx\_explicit, no w chainrule (Py)

**Parameters**

- **dxdotdx\_explicit** – partial derivative  $\dot{x}$  wrt  $x$
- **t** – timepoint
- **x** – Vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – heavyside vector
- **w** – vector with helper variables

virtual void **fdxdotdx\_explicit\_colptrs**(*SUNMatrixWrapper* &dxdotdx)

*Model* specific implementation of fdxdotdx\_explicit, colptrs part.

**Parameters** **dxdotdx** – sparse matrix to which colptrs will be written

virtual void **fdxdotdx\_explicit\_rowvals**(*SUNMatrixWrapper* &dxdotdx)

*Model* specific implementation of fdxdotdx\_explicit, rowvals part.

**Parameters** **dxdotdx** – sparse matrix to which rowvals will be written

virtual void **fdxdotdw**(*realtype* \*dxdotdw, *realtype* t, *realtype* const \*x, *realtype* const \*p, *realtype* const \*k, *realtype* const \*h, *realtype* const \*w)

*Model* specific implementation of fdxdotdw, data part.

**Parameters**

- **dxdotdw** – partial derivative  $\dot{x}$  wrt  $w$
- **t** – timepoint
- **x** – Vector with the states
- **p** – parameter vector
- **k** – constants vector
- **h** – Heaviside vector
- **w** – vector with helper variables

virtual void **fdxdotdw\_colptrs**(*SUNMatrixWrapper* &dxdotdw)

*Model* specific implementation of fdxdotdw, colptrs part.

**Parameters** **dxdotdw** – sparse matrix to which colptrs will be written

virtual void **fdxdotdw\_rowvals**(*SUNMatrixWrapper* &dxdotdw)

*Model* specific implementation of fdxdotdw, rowvals part.

**Parameters** **dxdotdw** – sparse matrix to which rowvals will be written

void **fdxdotdw**(*realtype* t, *const\_N\_Vector* x)

Sensitivity of  $dx/dt$  wrt model parameters  $w$ .

**Parameters**

- **t** – timepoint

- **x** – Vector with the states

void **fdxdotdp**(*realtype* t, *const\_N\_Vector* x)

Explicit sensitivity of dx/dt wrt model parameters p

#### Parameters

- **t** – timepoint
- **x** – Vector with the states

virtual void **fdxdotdp**(*realtype* t, *AmiVector* const &x, *AmiVector* const &dx) override

Model-specific sparse implementation of explicit parameter derivative of right hand side.

#### Parameters

- **t** – time
- **x** – state
- **dx** – time derivative of state (DAE only)

## Class ModelContext

- Defined in file\_include\_amici\_rdata.h

## Inheritance Relationships

### Base Type

- public amici::ContextManager (*Class ContextManager*)

## Class Documentation

class **ModelContext** : public amici::ContextManager

The *ModelContext* temporarily stores amici::Model::state and restores it when going out of scope.

### Public Functions

explicit **ModelContext**(*Model* \*model)

initialize backup of the original values.

Parameters **model** –

*ModelContext* &**operator=**(*ModelContext* const &other) = delete

**~ModelContext**()

void **restore**()

Restore original state on constructor-supplied *amici::Model*. Will be called during destruction. Explicit call is generally not necessary.

## Class `NewtonFailure`

- Defined in `file_include_amici_exception.h`

## Inheritance Relationships

### Base Type

- `public amici::AmiException` (*Class `AmiException`*)

## Class Documentation

class **NewtonFailure** : public `amici::AmiException`

Newton failure exception.

This exception should be thrown when the steady state computation failed to converge for this exception we can assume that we can recover from the exception and return a solution struct to the user

### Public Functions

**NewtonFailure**(int code, char const \*function)

Constructor, simply calls *AmiException* constructor.

#### Parameters

- **function** – name of the function in which the error occurred
- **code** – error code

### Public Members

int **error\_code**

error code returned by solver

## Class `NewtonSolver`

- Defined in `file_include_amici_newton_solver.h`

## Inheritance Relationships

### Derived Types

- `public amici::NewtonSolverDense` (*Class `NewtonSolverDense`*)
- `public amici::NewtonSolverSparse` (*Class `NewtonSolverSparse`*)

## Class Documentation

### class **NewtonSolver**

The *NewtonSolver* class sets up the linear solver for the Newton method.

Subclassed by *amici::NewtonSolverDense*, *amici::NewtonSolverSparse*

### Public Functions

explicit **NewtonSolver**(*Model* const &model)

Initializes solver according to the dimensions in the provided model.

**Parameters** **model** – pointer to the model object

void **getStep**(*AmiVector* &delta, *Model* &model, *SimulationState* const &state)

Computes the solution of one Newton iteration.

**Parameters**

- **delta** – containing the RHS of the linear system, will be overwritten by solution to the linear system
- **model** – pointer to the model instance
- **state** – current simulation state

void **computeNewtonSensis**(*AmiVectorArray* &sx, *Model* &model, *SimulationState* const &state)

Computes steady state sensitivities.

**Parameters**

- **sx** – pointer to state variable sensitivities
- **model** – pointer to the model instance
- **state** – current simulation state

virtual void **prepareLinearSystem**(*Model* &model, *SimulationState* const &state) = 0

Writes the Jacobian for the Newton iteration and passes it to the linear solver.

**Parameters**

- **model** – pointer to the model instance
- **state** – current simulation state

virtual void **prepareLinearSystemB**(*Model* &model, *SimulationState* const &state) = 0

Writes the Jacobian (JB) for the Newton iteration and passes it to the linear solver

**Parameters**

- **model** – pointer to the model instance
- **state** – current simulation state

virtual void **solveLinearSystem**(*AmiVector* &rhs) = 0

Solves the linear system for the Newton step.

**Parameters** **rhs** – containing the RHS of the linear system, will be overwritten by solution to the linear system

virtual void **reinitialize**() = 0

Reinitialize the linear solver.

virtual bool **is\_singular**(*Model* &model, *SimulationState* const &state) const = 0

Checks whether linear system is singular.

**Parameters**

- **model** – pointer to the model instance
- **state** – current simulation state

**Returns** boolean indicating whether the linear system is singular (condition number < 1/machine precision)

virtual ~**NewtonSolver**() = default

## Public Static Functions

static std::unique\_ptr<*NewtonSolver*> **getSolver**(*Solver* const &simulationSolver, *Model* const &model)

Factory method to create a *NewtonSolver* based on linsolType.

**Parameters**

- **simulationSolver** – solver with settings
- **model** – pointer to the model instance

**Returns** solver *NewtonSolver* according to the specified linsolType

## Protected Attributes

*AmiVector* **xdot\_**

dummy rhs, used as dummy argument when computing J and JB

*AmiVector* **x\_**

dummy state, attached to linear solver

*AmiVector* **xB\_**

dummy adjoint state, used as dummy argument when computing JB

*AmiVector* **dxB\_**

dummy differential adjoint state, used as dummy argument when computing JB

## Class **NewtonSolverDense**

- Defined in file\_include\_amici\_newton\_solver.h



## Inheritance Relationships

### Base Type

- `public amici::NewtonSolver` (*Class NewtonSolver*)

### Class Documentation

class **NewtonSolverDense** : public amici::*NewtonSolver*

The *NewtonSolverDense* provides access to the dense linear solver for the Newton method.

#### Public Functions

explicit **NewtonSolverDense**(*Model* const &model)

constructor for sparse solver

**Parameters** *model* – model instance that provides problem dimensions

**NewtonSolverDense**(*NewtonSolverDense* const&) = delete

*NewtonSolverDense* &**operator=**(*NewtonSolverDense* const &other) = delete

**~NewtonSolverDense**() override

virtual void **solveLinearSystem**(*AmiVector* &rhs) override

Solves the linear system for the Newton step.

**Parameters** *rhs* – containing the RHS of the linear system, will be overwritten by solution to the linear system

virtual void **prepareLinearSystem**(*Model* &model, *SimulationState* const &state) override

Writes the Jacobian for the Newton iteration and passes it to the linear solver.

**Parameters**

- **model** – pointer to the model instance
- **state** – current simulation state

virtual void **prepareLinearSystemB**(*Model* &model, *SimulationState* const &state) override

Writes the Jacobian (JB) for the Newton iteration and passes it to the linear solver

**Parameters**

- **model** – pointer to the model instance
- **state** – current simulation state

virtual void **reinitialize**() override

Reinitialize the linear solver.

virtual bool **is\_singular**(*Model* &model, *SimulationState* const &state) const override

Checks whether linear system is singular.

**Parameters**

- **model** – pointer to the model instance

- **state** – current simulation state

**Returns** boolean indicating whether the linear system is singular (condition number < 1/machine precision)

## Class `NewtonSolverSparse`

- Defined in `file_include_amici_newton_solver.h`

## Inheritance Relationships

### Base Type

- `public amici::NewtonSolver` (*Class `NewtonSolver`*)

## Class Documentation

class **`NewtonSolverSparse`** : `public amici::NewtonSolver`

The *`NewtonSolverSparse`* provides access to the sparse linear solver for the Newton method.

### Public Functions

explicit **`NewtonSolverSparse`**(*Model* const &model)

constructor for dense solver

**Parameters** **model** – model instance that provides problem dimensions

**`NewtonSolverSparse`**(*NewtonSolverSparse* const&) = delete

*NewtonSolverSparse* &**operator=**(*NewtonSolverSparse* const &other) = delete

**~NewtonSolverSparse**() override

virtual void **`solveLinearSystem`**(*AmiVector* &rhs) override

Solves the linear system for the Newton step.

**Parameters** **rhs** – containing the RHS of the linear system, will be overwritten by solution to the linear system

virtual void **`prepareLinearSystem`**(*Model* &model, *SimulationState* const &state) override

Writes the Jacobian for the Newton iteration and passes it to the linear solver.

**Parameters**

- **model** – pointer to the model instance
- **state** – current simulation state

virtual void **`prepareLinearSystemB`**(*Model* &model, *SimulationState* const &state) override

Writes the Jacobian (JB) for the Newton iteration and passes it to the linear solver

**Parameters**

- **model** – pointer to the model instance

- **state** – current simulation state

virtual bool **is\_singular**(*Model* &model, *SimulationState* const &state) const override

Checks whether linear system is singular.

#### Parameters

- **model** – pointer to the model instance
- **state** – current simulation state

**Returns** boolean indicating whether the linear system is singular (condition number < 1/machine precision)

virtual void **reinitialize**() override

Reinitialize the linear solver.

## Class ReturnData

- Defined in file\_include\_amici\_rdata.h

## Inheritance Relationships

### Base Type

- public amici::ModelDimensions (*Struct ModelDimensions*)

## Class Documentation

class **ReturnData** : public amici::ModelDimensions

Stores all data to be returned by amici::runAmiciSimulation.

NOTE: multi-dimensional arrays are stored in row-major order (C-style)

### Public Functions

**ReturnData**() = default

Default constructor.

**ReturnData**(std::vector<realtype> ts, *ModelDimensions* const &model\_dimensions, int nplist, int nmaxevent, int nt, int newton\_maxsteps, std::vector<ParameterScaling> pscale, *SecondOrderMode* o2mode, *SensitivityOrder* sensi, *SensitivityMethod* sensi\_meth, *RDataReporting* rdrm, bool quadratic\_llh, bool sigma\_res, realtype sigma\_offset)

Constructor.

#### Parameters

- **ts** – see amici::SimulationParameters::ts
- **model\_dimensions** – *Model* dimensions
- **nplist** – see amici::ModelDimensions::nplist
- **nmaxevent** – see amici::ModelDimensions::nmaxevent

- **nt** – see `amici::ModelDimensions::nt`
- **newton\_maxsteps** – see `amici::Solver::newton_maxsteps`
- **pscale** – see *`amici::SimulationParameters::pscale`*
- **o2mode** – see `amici::SimulationParameters::o2mode`
- **sensi** – see `amici::Solver::sensi`
- **sensi\_meth** – see `amici::Solver::sensi_meth`
- **rdrn** – see `amici::Solver::rdata_reporting`
- **quadratic\_llh** – whether model defines a quadratic nllh and computing res, sres and FIM makes sense
- **sigma\_res** – indicates whether additional residuals are to be added for each sigma
- **sigma\_offset** – offset to ensure real-valuedness of sigma residuals

**ReturnData**(*`Solver`* const &solver, *`Model`* const &model)

constructor that uses information from model and solver to appropriately initialize fields

**Parameters**

- **solver** – solver instance
- **model** – model instance

**~ReturnData()** = default

void **processSimulationObjects**(*`SteadystateProblem`* const \*preeq, *`ForwardProblem`* const \*fwd, *`BackwardProblem`* const \*bwd, *`SteadystateProblem`* const \*posteq, *`Model`* &model, *`Solver`* const &solver, *`ExpData`* const \*edata)

constructor that uses information from model and solver to appropriately initialize fields

**Parameters**

- **preeq** – simulated preequilibration problem, pass `nullptr` to ignore
- **fwd** – simulated forward problem, pass `nullptr` to ignore
- **bwd** – simulated backward problem, pass `nullptr` to ignore
- **posteq** – simulated postequilibration problem, pass `nullptr` to ignore
- **model** – matching model instance
- **solver** – matching solver instance
- **edata** – matching experimental data

## Public Members

`std::string` **id**

Arbitrary (not necessarily unique) identifier.

`std::vector<realtype>` **ts**

timepoints (shape `nt`)

`std::vector<realtype> xdot`  
time derivative (shape `nx`)

`std::vector<realtype> J`  
Jacobian of differential equation right hand side (shape `nx x nx`, row-major)

`std::vector<realtype> w`  
w data from the model (recurring terms in `xdot`, for imported SBML models from python, this contains the flux vector) (shape `nt x nw`, row major)

`std::vector<realtype> z`  
event output (shape `nmaxevent x nz`, row-major)

`std::vector<realtype> sigmaz`  
event output sigma standard deviation (shape `nmaxevent x nz`, row-major)

`std::vector<realtype> sz`  
parameter derivative of event output (shape `nmaxevent x nplist x nz`, row-major)

`std::vector<realtype> ssigmaz`  
parameter derivative of event output standard deviation (shape `nmaxevent x nplist x nz`, row-major)

`std::vector<realtype> rz`  
event trigger output (shape `nmaxevent x nz`, row-major)

`std::vector<realtype> srz`  
parameter derivative of event trigger output (shape `nmaxevent x nplist x nz`, row-major)

`std::vector<realtype> s2rz`  
second-order parameter derivative of event trigger output (shape `nmaxevent x nztrue x nplist x nplist`, row-major)

`std::vector<realtype> x`  
state (shape `nt x nx`, row-major)

`std::vector<realtype> sx`  
parameter derivative of state (shape `nt x nplist x nx`, row-major)

`std::vector<realtype> y`  
observable (shape `nt x ny`, row-major)

`std::vector<realtype> sigmay`  
observable standard deviation (shape `nt x ny`, row-major)

`std::vector<realtype> sy`  
parameter derivative of observable (shape `nt x nplist x ny`, row-major)

`std::vector<realtype> ssigmay`  
parameter derivative of observable standard deviation (shape `nt x nplist x ny`, row-major)

`std::vector<realtype> res`  
observable (shape `nt*ny`, row-major)

`std::vector<realtype> sres`  
parameter derivative of residual (shape `nt*ny x nplist`, row-major)

`std::vector<realtype> FIM`  
fisher information matrix (shape `nplist x nplist`, row-major)

`std::vector<int> numsteps`  
number of integration steps forward problem (shape `nt`)

`std::vector<int> numstepsB`  
number of integration steps backward problem (shape `nt`)

`std::vector<int> numrhsevals`  
number of right hand side evaluations forward problem (shape `nt`)

`std::vector<int> numrhsevalsB`  
number of right hand side evaluations backward problem (shape `nt`)

`std::vector<int> numerrtestfails`  
number of error test failures forward problem (shape `nt`)

`std::vector<int> numerrtestfailsB`  
number of error test failures backward problem (shape `nt`)

`std::vector<int> numnonlinsolvconvfails`  
number of linear solver convergence failures forward problem (shape `nt`)

`std::vector<int> numnonlinsolvconvfailsB`  
number of linear solver convergence failures backward problem (shape `nt`)

`std::vector<int> order`  
employed order forward problem (shape `nt`)

`double cpu_time = 0.0`  
computation time of forward solve [ms]  
  
.. warning:: If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

double **cpu\_timeB** = 0.0

computation time of backward solve [ms]

.. warning:: If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

double **cpu\_time\_total** = 0.0

total CPU time from entering runAmiciSimulation until exiting [ms]

.. warning:: If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

std::vector<*SteadyStateStatus*> **preeq\_status**

flags indicating success of steady state solver (preequilibration)

double **preeq\_cpu\_time** = 0.0

computation time of the steady state solver [ms] (preequilibration)

.. warning:: If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

double **preeq\_cpu\_timeB** = 0.0

computation time of the steady state solver of the backward problem [ms] (preequilibration)

.. warning:: If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

std::vector<*SteadyStateStatus*> **posteq\_status**

flags indicating success of steady state solver (postequilibration)

double **posteq\_cpu\_time** = 0.0

computation time of the steady state solver [ms] (postequilibration)

.. warning:: If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

double **posteq\_cpu\_timeB** = 0.0

computation time of the steady state solver of the backward problem [ms] (postequilibration)

.. warning:: If AMICI was built without boost, this tracks the CPU-time of the current process. Therefore, in a multi-threaded context, this value may be incorrect.

std::vector<int> **preeq\_numsteps**

number of Newton steps for steady state problem (preequilibration) [newton, simulation, newton] (length = 3)

int **preeq\_numstepsB** = 0

number of simulation steps for adjoint steady state problem (preequilibration) [== 0 if analytical solution worked, > 0 otherwise]

`std::vector<int> posteq_numsteps`

number of Newton steps for steady state problem (preequilibration) [newton, simulation, newton] (shape 3) (postequilibration)

`int posteq_numstepsB = 0`

number of simulation steps for adjoint steady state problem (postequilibration) [== 0 if analytical solution worked, > 0 otherwise]

*realtype* `preeq_t = NAN`

time when steadystate was reached via simulation (preequilibration)

*realtype* `preeq_wrms = NAN`

weighted root-mean-square of the rhs when steadystate was reached (preequilibration)

*realtype* `posteq_t = NAN`

time when steadystate was reached via simulation (postequilibration)

*realtype* `posteq_wrms = NAN`

weighted root-mean-square of the rhs when steadystate was reached (postequilibration)

`std::vector<realtype> x0`

initial state (shape `nx`)

`std::vector<realtype> x_ss`

preequilibration steady state (shape `nx`)

`std::vector<realtype> sx0`

initial sensitivities (shape `nplist x nx`, row-major)

`std::vector<realtype> sx_ss`

preequilibration sensitivities (shape `nplist x nx`, row-major)

*realtype* `llh = 0.0`

log-likelihood value

*realtype* `chi2 = 0.0`

$\chi^2$  value

`std::vector<realtype> s1lh`

parameter derivative of log-likelihood (shape `nplist`)

`std::vector<realtype> s21lh`

second-order parameter derivative of log-likelihood (shape `nJ-1 x nplist`, row-major)



int **status** = *AMICI\_NOT\_RUN*

Simulation status code.

One of:

- *AMICI\_SUCCESS*, indicating successful simulation
- *AMICI\_MAX\_TIME\_EXCEEDED*, indicating that the simulation did not finish within the allowed time (see *Solver*.{set,get}MaxTime)
- *AMICI\_ERROR*, indicating that some error occurred during simulation (a more detailed error message will have been printed).
- *AMICI\_NOT\_RUN*, if no simulation was started

int **nx** = {0}

number of states (alias *nx\_rdata*, kept for backward compatibility)

int **nxtrue** = {0}

number of states in the unaugmented system (alias *nxtrue\_rdata*, kept for backward compatibility)

int **nplist** = {0}

number of parameter for which sensitivities were requested

int **nmaxevent** = {0}

maximal number of occurring events (for every event type)

int **nt** = {0}

number of considered timepoints

int **newton\_maxsteps** = {0}

maximal number of newton iterations for steady state calculation

std::vector<*ParameterScaling*> **pscale**

scaling of parameterization

*SecondOrderMode* **o2mode** = {*SecondOrderMode::none*}

flag indicating whether second-order sensitivities were requested

*SensitivityOrder* **sensi** = {*SensitivityOrder::none*}

sensitivity order

*SensitivityMethod* **sensi\_meth** = {*SensitivityMethod::none*}

sensitivity method

*RDataReporting* **rdata\_reporting** = {*RDataReporting::full*}

reporting mode

bool **sigma\_res**

boolean indicating whether residuals for standard deviations have been added

std::vector<*LogItem*> **messages**

log messages

## Protected Functions

void **initializeLikelihoodReporting**(bool quadratic\_llh)

initializes storage for likelihood reporting mode

**Parameters** **quadratic\_llh** – whether model defines a quadratic nllh and computing res, sres and FIM makes sense.

void **initializeResidualReporting**(bool enable\_res)

initializes storage for residual reporting mode

**Parameters** **enable\_res** – whether residuals are to be computed

void **initializeFullReporting**(bool enable\_fim)

initializes storage for full reporting mode

**Parameters** **enable\_fim** – whether FIM Hessian approximation is to be computed

void **initializeObjectiveFunction**(bool enable\_chi2)

initialize values for chi2 and llh and derivatives

**Parameters** **enable\_chi2** – whether chi2 values are to be computed

void **processPreEquilibration**(*SteadystateProblem* const &preeq, *Model* &model)

extracts data from a preequilibration *SteadystateProblem*

**Parameters**

- **preeq** – *SteadystateProblem* for preequilibration
- **model** – *Model* instance to compute return values

void **processPostEquilibration**(*SteadystateProblem* const &posteq, *Model* &model, *ExpData* const \*edata)

extracts data from a preequilibration *SteadystateProblem*

**Parameters**

- **posteq** – *SteadystateProblem* for postequilibration
- **model** – *Model* instance to compute return values
- **edata** – *ExpData* instance containing observable data

void **processForwardProblem**(*ForwardProblem* const &fwd, *Model* &model, *ExpData* const \*edata)

extracts results from forward problem

**Parameters**

- **fwd** – forward problem
- **model** – model that was used for forward simulation
- **edata** – *ExpData* instance containing observable data

void **processBackwardProblem**(*ForwardProblem* const &fwd, *BackwardProblem* const &bwd, *SteadystateProblem* const \*preeq, *Model* &model)

extracts results from backward problem

#### Parameters

- **fwd** – forward problem
- **bwd** – backward problem
- **preeq** – *SteadystateProblem* for preequilibration
- **model** – model that was used for forward/backward simulation

void **processSolver**(*Solver* const &solver)

extracts results from solver

**Parameters** **solver** – solver that was used for forward/backward simulation

template<class T>

inline void **storeJacobianAndDerivativeInReturnData**(T const &problem, *Model* &model)

Evaluates and stores the Jacobian and right hand side at final timepoint.

#### Parameters

- **problem** – forward problem or steadystate problem
- **model** – model that was used for forward/backward simulation

void **readSimulationState**(*SimulationState* const &state, *Model* &model)

sets member variables and model state according to provided simulation state

#### Parameters

- **state** – simulation state provided by Problem
- **model** – model that was used for forward/backward simulation

void **fres**(int it, *Model* &model, *ExpData* const &edata)

Residual function.

#### Parameters

- **it** – time index
- **model** – model that was used for forward/backward simulation
- **edata** – *ExpData* instance containing observable data

void **fchi2**(int it, *ExpData* const &edata)

Chi-squared function.

#### Parameters

- **it** – time index
- **edata** – *ExpData* instance containing observable data

void **fsres**(int it, *Model* &model, *ExpData* const &edata)

Residual sensitivity function.

#### Parameters

- **it** – time index
- **model** – model that was used for forward/backward simulation

- **edata** – *ExpData* instance containing observable data

void **ffim**(int it, *Model* &model, *ExpData* const &edata)

Fisher information matrix function.

**Parameters**

- **it** – time index
- **model** – model that was used for forward/backward simulation
- **edata** – *ExpData* instance containing observable data

void **invalidate**(int it\_start)

Set likelihood, state variables, outputs and respective sensitivities to NaN (typically after integration failure)

**Parameters** **it\_start** – time index at which to start invalidating

void **invalidateLLH**()

Set likelihood and chi2 to NaN (typically after integration failure)

void **invalidateSLLH**()

Set likelihood sensitivities to NaN (typically after integration failure)

void **applyChainRuleFactorToSimulationResults**(*Model* const &model)

applies the chain rule to account for parameter transformation in the sensitivities of simulation results

**Parameters** **model** – *Model* from which the *ReturnData* was obtained

inline bool **computingFSA**() const

Checks whether forward sensitivity analysis is performed.

**Returns** boolean indicator

void **getDataOutput**(int it, *Model* &model, *ExpData* const \*edata)

Extracts output information for data-points, expects that `x_solver_` and `sx_solver_` were set appropriately.

**Parameters**

- **it** – timepoint index
- **model** – model that was used in forward solve
- **edata** – *ExpData* instance carrying experimental data

void **getDataSensisFSA**(int it, *Model* &model, *ExpData* const \*edata)

Extracts data information for forward sensitivity analysis, expects that `x_solver_` and `sx_solver_` were set appropriately.

**Parameters**

- **it** – index of current timepoint
- **model** – model that was used in forward solve
- **edata** – *ExpData* instance carrying experimental data

void **getEventOutput**(*realtype* t, const std::vector<int> rootidx, *Model* &model, *ExpData* const \*edata)

Extracts output information for events, expects that `x_solver_` and `sx_solver_` were set appropriately.

**Parameters**

- **t** – event timepoint
- **rootidx** – information about which roots fired (1 indicating fired, 0/-1 for not)

- **model** – model that was used in forward solve
- **edata** – *ExpData* instance carrying experimental data

void **getEventSensisFSA**(int ie, *realtype* t, *Model* &model, *ExpData* const \*edata)

Extracts event information for forward sensitivity analysis, expects that `x_solver_` and `sx_solver_` were set appropriately.

#### Parameters

- **ie** – index of event type
- **t** – event timepoint
- **model** – model that was used in forward solve
- **edata** – *ExpData* instance carrying experimental data

void **handleSx0Backward**(*Model* const &model, *SteadystateProblem* const &preeq, std::vector<*realtype*> &llhS0, *AmiVector* &xQB) const

Updates contribution to likelihood from quadratures (xQB), if preequilibration was run in adjoint mode.

#### Parameters

- **model** – model that was used for forward/backward simulation
- **preeq** – *SteadystateProblem* for preequilibration
- **llhS0** – contribution to likelihood for initial state sensitivities of preequilibration
- **xQB** – vector with quadratures from adjoint computation

void **handleSx0Forward**(*Model* const &model, std::vector<*realtype*> &llhS0, *AmiVector* &xB) const

Updates contribution to likelihood for initial state sensitivities (llhS0), if no preequilibration was run or if forward sensitivities were used.

#### Parameters

- **model** – model that was used for forward/backward simulation
- **llhS0** – contribution to likelihood for initial state sensitivities
- **xB** – vector with final adjoint state (excluding conservation laws)

## Protected Attributes

*realtype* **sigma\_offset**

offset for sigma\_residuals

*realtype* **t\_**

timepoint for model evaluation

*AmiVector* **x\_solver\_**

partial state vector, excluding states eliminated from conservation laws

*AmiVector* **dx\_solver\_**

partial time derivative of state vector, excluding states eliminated from conservation laws

*AmiVectorArray* **sx\_solver\_**

partial sensitivity state vector array, excluding states eliminated from conservation laws

*AmiVector* **x\_rdata\_**

full state vector, including states eliminated from conservation laws

*AmiVectorArray* **sx\_rdata\_**

full sensitivity state vector array, including states eliminated from conservation laws

std::vector<int> **nroots\_**

array of number of found roots for a certain event type (shape ne)

## Friends

template<class **Archive**>

friend void **serialize**(*Archive* &ar, *ReturnData* &r, unsigned int version)

Serialize *ReturnData* (see boost::serialization::serialize)

### Parameters

- **ar** – Archive to serialize to
- **r** – Data to serialize
- **version** – Version number

## Class SetupFailure

- Defined in file\_include\_amici\_exception.h

## Inheritance Relationships

### Base Type

- public amici::AmiException (*Class AmiException*)

## Class Documentation

class **SetupFailure** : public amici::AmiException

Setup failure exception.

This exception should be thrown when the solver setup failed for this exception we can assume that we cannot recover from the exception and an error will be thrown

## Public Functions

explicit **SetupFailure**(char const \*fmt, ...)

Constructor with printf style interface.

### Parameters

- **fmt** – error message with printf format
- ... – printf formatting variables

## Class `SimulationParameters`

- Defined in file `include/amici_simulation_parameters.h`

## Inheritance Relationships

### Derived Type

- public `amici::ExpData` (*Class `ExpData`*)

## Class Documentation

class **SimulationParameters**

Container for various simulation parameters.

Subclassed by *`amici::ExpData`*

## Public Functions

**SimulationParameters**() = default

inline explicit **SimulationParameters**(std::vector<*realtype*> timepoints)

Constructor.

**Parameters** **timepoints** – Timepoints for which simulation results are requested

inline **SimulationParameters**(std::vector<*realtype*> fixedParameters, std::vector<*realtype*> parameters)

Constructor.

### Parameters

- **fixedParameters** – *Model* constants
- **parameters** – *Model* parameters

inline **SimulationParameters**(std::vector<*realtype*> fixedParameters, std::vector<*realtype*> parameters, std::vector<int> plist)

Constructor.

### Parameters

- **fixedParameters** – *Model* constants
- **parameters** – *Model* parameters

- **plist** – *Model* parameter indices w.r.t. which sensitivities are to be computed

inline **SimulationParameters**(std::vector<*realtype*> timepoints, std::vector<*realtype*> fixedParameters, std::vector<*realtype*> parameters)

Constructor.

#### Parameters

- **timepoints** – Timepoints for which simulation results are requested
- **fixedParameters** – *Model* constants
- **parameters** – *Model* parameters

void **reinitializeAllFixedParameterDependentInitialStatesForPresimulation**(int nx\_rdata)

Set reinitialization of all states based on model constants for presimulation (only meaningful if preequilibration is performed).

Convenience function to populate `reinitialization_state_idxes_presim` and `reinitialization_state_idxes_sim`

**Parameters** `nx_rdata` – Number of states (*Model::nx\_rdata*)

void **reinitializeAllFixedParameterDependentInitialStatesForSimulation**(int nx\_rdata)

Set reinitialization of all states based on model constants for the ‘main’ simulation (only meaningful if presimulation or preequilibration is performed).

Convenience function to populate `reinitialization_state_idxes_presim` and `reinitialization_state_idxes_sim`

**Parameters** `nx_rdata` – Number of states (*Model::nx\_rdata*)

void **reinitializeAllFixedParameterDependentInitialStates**(int nx\_rdata)

Set reinitialization of all states based on model constants for all simulation phases.

Convenience function to populate `reinitialization_state_idxes_presim` and `reinitialization_state_idxes_sim`

**Parameters** `nx_rdata` – Number of states (*Model::nx\_rdata*)

## Public Members

std::vector<*realtype*> **fixedParameters**

*Model* constants.

Vector of size *Model::nk()* or empty

std::vector<*realtype*> **fixedParametersPreequilibration**

*Model* constants for pre-equilibration.

Vector of size *Model::nk()* or empty.

std::vector<*realtype*> **fixedParametersPresimulation**

*Model* constants for pre-simulation.

Vector of size *Model::nk()* or empty.



`std::vector<realtype> parameters`

*Model* parameters.

Vector of size *Model::np()* or empty with parameter scaled according to `SimulationParameter::pscale`.

`std::vector<realtype> x0`

Initial state.

Vector of size `Model::nx()` or empty

`std::vector<realtype> sx0`

Initial state sensitivities.

Dimensions: `Model::nx() * Model::nplist()`, `Model::nx() * ExpData::plist.size()`, if *ExpData::plist* is not empty, or empty

`std::vector<ParameterScaling> pscale`

Parameter scales.

Vector of parameter scale of size *Model::np()*, indicating how/if each parameter is to be scaled.

`std::vector<int> plist`

Parameter indices w.r.t. which to compute sensitivities.

*realtype* `tstart_ = {0.0}`

starting time

*realtype* `t_presim = {0.0}`

Duration of pre-simulation.

If this is > 0, presimulation will be performed from (`model->t0 - t_presim`) to `model->t0` using the fixed-Parameters in `fixedParametersPresimulation`

`std::vector<realtype> ts_`

Timepoints for which model state/outputs/... are requested.

Vector of timepoints.

`bool reinitializeFixedParameterInitialStates = {false}`

Flag indicating whether reinitialization of states depending on fixed parameters is activated.

`std::vector<int> reinitialization_state_idxes_presim`

Indices of states to be reinitialized based on provided presimulation constants / fixed parameters.

`std::vector<int> reinitialization_state_idxes_sim`

Indices of states to be reinitialized based on provided constants / fixed parameters.

## Class Solver

- Defined in file\_include\_amici\_solver.h

## Inheritance Relationships

## Derived Types

- public amici::CVodeSolver (*Class CVodeSolver*)
- public amici::IDASolver (*Class IDASolver*)

## Class Documentation

### class Solver

The *Solver* class provides a generic interface to CVODES and IDAS solvers, individual realizations are realized in the *CVodeSolver* and the *IDASolver* class. All transient private/protected members (CVODES/IDAS memory, interface variables and status flags) are specified as mutable and not included in serialization or equality checks. No solver setting parameter should be marked mutable.

NOTE: Any changes in data members here must be propagated to copy ctor, equality operator, serialization functions in serialization.h, and amici::hdf5::(read/write)SolverSettings(From/To)HDF5 in hdf5.cpp.

Subclassed by *amici::CVodeSolver*, *amici::IDASolver*

### Public Types

using **user\_data\_type** = std::pair<*Model*\*, *Solver* const\*>

Type of what is passed to Sundials solvers as user\_data

### Public Functions

**Solver**() = default

Default constructor.

**Solver**(*Solver* const &other)

*Solver* copy constructor.

**Parameters** **other** –

virtual **~Solver**() = default

virtual *Solver* \***clone**() const = 0

Clone this instance.

**Returns** The clone

int **run**(*realtype* tout) const

runs a forward simulation until the specified timepoint

**Parameters** **tout** – next timepoint

**Returns** status flag

int **step**(*realtype* tout) const

makes a single step in the simulation

**Parameters** tout – next timepoint

**Returns** status flag

void **runB**(*realtype* tout) const

runs a backward simulation until the specified timepoint

**Parameters** tout – next timepoint

void **setup**(*realtype* t0, *Model* \*model, *AmiVector* const &x0, *AmiVector* const &dx0, *AmiVectorArray* const &sx0, *AmiVectorArray* const &sdx0) const

Initializes the ami memory object and applies specified options.

**Parameters**

- **t0** – initial timepoint
- **model** – pointer to the model instance
- **x0** – initial states
- **dx0** – initial derivative states
- **sx0** – initial state sensitivities
- **sdx0** – initial derivative state sensitivities

void **setupB**(int \*which, *realtype* tf, *Model* \*model, *AmiVector* const &xB0, *AmiVector* const &dxB0, *AmiVector* const &xQB0) const

Initializes the AMI memory object for the backwards problem.

**Parameters**

- **which** – index of the backward problem, will be set by this routine
- **tf** – final timepoint (initial timepoint for the bwd problem)
- **model** – pointer to the model instance
- **xB0** – initial adjoint states
- **dxB0** – initial adjoint derivative states
- **xQB0** – initial adjoint quadratures

void **setupSteadystate**(const *realtype* t0, *Model* \*model, *AmiVector* const &x0, *AmiVector* const &dx0, *AmiVector* const &xB0, *AmiVector* const &dxB0, *AmiVector* const &xQ0) const

Initializes the ami memory for quadrature computation.

**Parameters**

- **t0** – initial timepoint
- **model** – pointer to the model instance
- **x0** – initial states
- **dx0** – initial derivative states
- **xB0** – initial adjoint states
- **dxB0** – initial derivative adjoint states

- **xQ0** – initial quadrature vector

void **updateAndReinitStatesAndSensitivities**(*Model* \*model) const

Reinitializes state and respective sensitivities (if necessary) according to changes in fixedParameters.

**Parameters** **model** – pointer to the model instance

virtual void **getRootInfo**(int \*rootsfound) const = 0

getRootInfo extracts information which event occurred

**Parameters** **rootsfound** – array with flags indicating whether the respective event occurred

virtual void **calcIC**(*realtype* tout1) const = 0

Calculates consistent initial conditions, assumes initial states to be correct (DAE only)

**Parameters** **tout1** – next timepoint to be computed (sets timescale)

virtual void **calcICB**(int which, *realtype* tout1) const = 0

Calculates consistent initial conditions for the backwards problem, assumes initial states to be correct (DAE only)

**Parameters**

- **which** – identifier of the backwards problem
- **tout1** – next timepoint to be computed (sets timescale)

virtual void **solveB**(*realtype* tBout, int itaskB) const = 0

Solves the backward problem until a predefined timepoint (adjoint only)

**Parameters**

- **tBout** – timepoint until which simulation should be performed
- **itaskB** – task identifier, can be CV\_NORMAL or CV\_ONE\_STEP

virtual void **turnOffRootFinding**() const = 0

Disable rootfinding.

*SensitivityMethod* **getSensitivityMethod**() const

Return current sensitivity method.

**Returns** method enum

void **setSensitivityMethod**(*SensitivityMethod* sensi\_meth)

Set sensitivity method.

**Parameters** **sensi\_meth** –

*SensitivityMethod* **getSensitivityMethodPreequilibration**() const

Return current sensitivity method during preequilibration.

**Returns** method enum

void **setSensitivityMethodPreequilibration**(*SensitivityMethod* sensi\_meth\_preeq)

Set sensitivity method for preequilibration.

**Parameters** **sensi\_meth\_preeq** –

void **switchForwardSensisOff**() const

Disable forward sensitivity integration (used in steady state sim)

int **getNewtonMaxSteps**() const

Get maximum number of allowed Newton steps for steady state computation.

**Returns**

void **setNewtonMaxSteps**(int newton\_maxsteps)

Set maximum number of allowed Newton steps for steady state computation.

**Parameters** **newton\_maxsteps** –

*NewtonDampingFactorMode* **getNewtonDampingFactorMode**() const

Get a state of the damping factor used in the Newton solver.

**Returns**

void **setNewtonDampingFactorMode**(*NewtonDampingFactorMode* dampingFactorMode)

Turn on/off a damping factor in the Newton method.

**Parameters** **dampingFactorMode** –

double **getNewtonDampingFactorLowerBound**() const

Get a lower bound of the damping factor used in the Newton solver.

**Returns**

void **setNewtonDampingFactorLowerBound**(double dampingFactorLowerBound)

Set a lower bound of the damping factor in the Newton solver.

**Parameters** **dampingFactorLowerBound** –

*SensitivityOrder* **getSensitivityOrder**() const

Get sensitivity order.

**Returns** sensitivity order

void **setSensitivityOrder**(*SensitivityOrder* sensi)

Set the sensitivity order.

**Parameters** **sensi** – sensitivity order

double **getRelativeTolerance**() const

Get the relative tolerances for the forward problem.

Same tolerance is used for the backward problem if not specified differently via setRelativeToleranceASA.

**Returns** relative tolerances

void **setRelativeTolerance**(double rtol)

Sets the relative tolerances for the forward problem.

Same tolerance is used for the backward problem if not specified differently via setRelativeToleranceASA.

**Parameters** **rtol** – relative tolerance (non-negative number)

double **getAbsoluteTolerance**() const

Get the absolute tolerances for the forward problem.

Same tolerance is used for the backward problem if not specified differently via setAbsoluteToleranceASA.

**Returns** absolute tolerances

void **setAbsoluteTolerance**(double atol)

Sets the absolute tolerances for the forward problem.

Same tolerance is used for the backward problem if not specified differently via `setAbsoluteToleranceASA`.

**Parameters** `atol` – absolute tolerance (non-negative number)

double **getRelativeToleranceFSA**() const

Returns the relative tolerances for the forward sensitivity problem.

**Returns** relative tolerances

void **setRelativeToleranceFSA**(double rtol)

Sets the relative tolerances for the forward sensitivity problem.

**Parameters** `rtol` – relative tolerance (non-negative number)

double **getAbsoluteToleranceFSA**() const

Returns the absolute tolerances for the forward sensitivity problem.

**Returns** absolute tolerances

void **setAbsoluteToleranceFSA**(double atol)

Sets the absolute tolerances for the forward sensitivity problem.

**Parameters** `atol` – absolute tolerance (non-negative number)

double **getRelativeToleranceB**() const

Returns the relative tolerances for the adjoint sensitivity problem.

**Returns** relative tolerances

void **setRelativeToleranceB**(double rtol)

Sets the relative tolerances for the adjoint sensitivity problem.

**Parameters** `rtol` – relative tolerance (non-negative number)

double **getAbsoluteToleranceB**() const

Returns the absolute tolerances for the backward problem for adjoint sensitivity analysis.

**Returns** absolute tolerances

void **setAbsoluteToleranceB**(double atol)

Sets the absolute tolerances for the backward problem for adjoint sensitivity analysis.

**Parameters** `atol` – absolute tolerance (non-negative number)

double **getRelativeToleranceQuadratures**() const

Returns the relative tolerance for the quadrature problem.

**Returns** relative tolerance

void **setRelativeToleranceQuadratures**(double rtol)

sets the relative tolerance for the quadrature problem

**Parameters** `rtol` – relative tolerance (non-negative number)

double **getAbsoluteToleranceQuadratures**() const

returns the absolute tolerance for the quadrature problem

**Returns** absolute tolerance

void **setAbsoluteToleranceQuadratures**(double atol)

sets the absolute tolerance for the quadrature problem

**Parameters** **atol** – absolute tolerance (non-negative number)

double **getSteadyStateToleranceFactor**() const

returns the steady state simulation tolerance factor.

Steady state simulation tolerances are the product of the simulation tolerances and this factor, unless manually set with **set(Absolute/Relative)ToleranceSteadyState()**.

**Returns** steady state simulation tolerance factor

void **setSteadyStateToleranceFactor**(double factor)

set the steady state simulation tolerance factor.

Steady state simulation tolerances are the product of the simulation tolerances and this factor, unless manually set with **set(Absolute/Relative)ToleranceSteadyState()**.

**Parameters** **factor** – tolerance factor (non-negative number)

double **getRelativeToleranceSteadyState**() const

returns the relative tolerance for the steady state problem

**Returns** relative tolerance

void **setRelativeToleranceSteadyState**(double rtol)

sets the relative tolerance for the steady state problem

**Parameters** **rtol** – relative tolerance (non-negative number)

double **getAbsoluteToleranceSteadyState**() const

returns the absolute tolerance for the steady state problem

**Returns** absolute tolerance

void **setAbsoluteToleranceSteadyState**(double atol)

sets the absolute tolerance for the steady state problem

**Parameters** **atol** – absolute tolerance (non-negative number)

double **getSteadyStateSensiToleranceFactor**() const

returns the steady state sensitivity simulation tolerance factor.

Steady state sensitivity simulation tolerances are the product of the sensitivity simulation tolerances and this factor, unless manually set with **set(Absolute/Relative)ToleranceSteadyStateSensi()**.

**Returns** steady state simulation tolerance factor

void **setSteadyStateSensiToleranceFactor**(double factor)

set the steady state sensitivity simulation tolerance factor.

Steady state sensitivity simulation tolerances are the product of the sensitivity simulation tolerances and this factor, unless manually set with **set(Absolute/Relative)ToleranceSteadyStateSensi()**.

**Parameters** **factor** – tolerance factor (non-negative number)

double **getRelativeToleranceSteadyStateSensi**() const

returns the relative tolerance for the sensitivities of the steady state problem

**Returns** relative tolerance

void **setRelativeToleranceSteadyStateSensi**(double rtol)  
sets the relative tolerance for the sensitivities of the steady state problem  
**Parameters** **rtol** – relative tolerance (non-negative number)

double **getAbsoluteToleranceSteadyStateSensi**() const  
returns the absolute tolerance for the sensitivities of the steady state problem  
**Returns** absolute tolerance

void **setAbsoluteToleranceSteadyStateSensi**(double atol)  
sets the absolute tolerance for the sensitivities of the steady state problem  
**Parameters** **atol** – absolute tolerance (non-negative number)

long int **getMaxSteps**() const  
returns the maximum number of solver steps for the forward problem  
**Returns** maximum number of solver steps

void **setMaxSteps**(long int maxsteps)  
sets the maximum number of solver steps for the forward problem  
**Parameters** **maxsteps** – maximum number of solver steps (positive number)

double **getMaxTime**() const  
Returns the maximum time allowed for integration.  
**Returns** Time in seconds

void **setMaxTime**(double maxtime)  
Set the maximum CPU time allowed for integration.  
**Parameters** **maxtime** – Time in seconds. Zero means infinite time.

void **startTimer**() const  
Start timer for tracking integration time.

bool **timeExceeded**(int interval = 1) const  
Check whether maximum integration time was exceeded.  
**Parameters** **interval** – Only check the time every **interval** ths call to avoid potentially relatively expensive syscalls  
**Returns** True if the maximum integration time was exceeded, false otherwise.

long int **getMaxStepsBackwardProblem**() const  
returns the maximum number of solver steps for the backward problem  
**Returns** maximum number of solver steps

void **setMaxStepsBackwardProblem**(long int maxsteps)  
sets the maximum number of solver steps for the backward problem

---

**Note:** default behaviour (100 times the value for the forward problem) can be restored by passing maxsteps=0

---

**Parameters** **maxsteps** – maximum number of solver steps (non-negative number)



*LinearMultistepMethod* **getLinearMultistepMethod()** const

returns the linear system multistep method

**Returns** linear system multistep method

void **setLinearMultistepMethod**(*LinearMultistepMethod* lmm)

sets the linear system multistep method

**Parameters** **lmm** – linear system multistep method

*NonlinearSolverIteration* **getNonlinearSolverIteration()** const

returns the nonlinear system solution method

**Returns**

void **setNonlinearSolverIteration**(*NonlinearSolverIteration* iter)

sets the nonlinear system solution method

**Parameters** **iter** – nonlinear system solution method

*InterpolationType* **getInterpolationType()** const

getInterpolationType

**Returns**

void **setInterpolationType**(*InterpolationType* interpType)

sets the interpolation of the forward solution that is used for the backwards problem

**Parameters** **interpType** – interpolation type

int **getStateOrdering()** const

Gets KLU / SuperLUMT state ordering mode.

**Returns** State-ordering as integer according to *SUNLinSolKLU::StateOrdering* or *SUNLinSolSuperLUMT::StateOrdering* (which differ).

void **setStateOrdering**(int ordering)

Sets KLU / SuperLUMT state ordering mode.

This only applies when linsol is set to *LinearSolver::KLU* or *LinearSolver::SuperLUMT*. Mind the difference between *SUNLinSolKLU::StateOrdering* and *SUNLinSolSuperLUMT::StateOrdering*.

**Parameters** **ordering** – state ordering

bool **getStabilityLimitFlag()** const

returns stability limit detection mode

**Returns** stldet can be false (deactivated) or true (activated)

void **setStabilityLimitFlag**(bool stldet)

set stability limit detection mode

**Parameters** **stldet** – can be false (deactivated) or true (activated)

*LinearSolver* **getLinearSolver()** const

getLinearSolver

**Returns**

void **setLinearSolver**(*LinearSolver* linsol)

setLinearSolver

**Parameters** **linsol** –

*InternalSensitivityMethod* **getInternalSensitivityMethod()** const

returns the internal sensitivity method

**Returns** internal sensitivity method

void **setInternalSensitivityMethod**(*InternalSensitivityMethod* ism)

sets the internal sensitivity method

**Parameters** **ism** – internal sensitivity method

*RDataReporting* **getReturnDataReportingMode()** const

returns the *ReturnData* reporting mode

**Returns** *ReturnData* reporting mode

void **setReturnDataReportingMode**(*RDataReporting* rdrm)

sets the *ReturnData* reporting mode

**Parameters** **rdrm** – *ReturnData* reporting mode

void **writeSolution**(*realtype* \*t, *AmiVector* &x, *AmiVector* &dx, *AmiVectorArray* &sx, *AmiVector* &xQ)  
const

write solution from forward simulation

**Parameters**

- **t** – time
- **x** – state
- **dx** – derivative state
- **sx** – state sensitivity
- **xQ** – quadrature

void **writeSolutionB**(*realtype* \*t, *AmiVector* &xB, *AmiVector* &dxB, *AmiVector* &xQB, int which) const

write solution from backward simulation

**Parameters**

- **t** – time
- **xB** – adjoint state
- **dxB** – adjoint derivative state
- **xQB** – adjoint quadrature
- **which** – index of adjoint problem

*AmiVector* const &**getState**(*realtype* t) const

Access state solution at time t.

**Parameters** **t** – time

**Returns** x or interpolated solution dky

*AmiVector* const &**getDerivativeState**(*realtype* t) const

Access derivative state solution at time t.

**Parameters** **t** – time

**Returns** dx or interpolated solution dky

*AmiVectorArray* const &**getStateSensitivity**(*realtype* t) const

Access state sensitivity solution at time t.

**Parameters** **t** – time

**Returns** (interpolated) solution **sx**

*AmiVector* const &**getAdjointState**(int which, *realtype* t) const

Access adjoint solution at time t.

**Parameters**

- **which** – adjoint problem index
- **t** – time

**Returns** (interpolated) solution **xB**

*AmiVector* const &**getAdjointDerivativeState**(int which, *realtype* t) const

Access adjoint derivative solution at time t.

**Parameters**

- **which** – adjoint problem index
- **t** – time

**Returns** (interpolated) solution **dxB**

*AmiVector* const &**getAdjointQuadrature**(int which, *realtype* t) const

Access adjoint quadrature solution at time t.

**Parameters**

- **which** – adjoint problem index
- **t** – time

**Returns** (interpolated) solution **xQB**

*AmiVector* const &**getQuadrature**(*realtype* t) const

Access quadrature solution at time t.

**Parameters** **t** – time

**Returns** (interpolated) solution **xQ**

virtual void **reInit**(*realtype* t0, *AmiVector* const &yy0, *AmiVector* const &yp0) const = 0

Reinitializes the states in the solver after an event occurrence.

**Parameters**

- **t0** – reinitialization timepoint
- **yy0** – initial state variables
- **yp0** – initial derivative state variables (DAE only)

virtual void **sensReInit**(*AmiVectorArray* const &yyS0, *AmiVectorArray* const &ypS0) const = 0

Reinitializes the state sensitivities in the solver after an event occurrence.

**Parameters**

- **yyS0** – new state sensitivity
- **ypS0** – new derivative state sensitivities (DAE only)

virtual void **sensToggleOff**() const = 0

Switches off computation of state sensitivities without deallocating the memory for sensitivities.

virtual void **reInitB**(int which, *realtype* tB0, *AmiVector* const &yyB0, *AmiVector* const &ypB0) const = 0

Reinitializes the adjoint states after an event occurrence.

**Parameters**

- **which** – identifier of the backwards problem
- **tB0** – reinitialization timepoint
- **yyB0** – new adjoint state
- **ypB0** – new adjoint derivative state

virtual void **quadReInitB**(int which, *AmiVector* const &yQB0) const = 0

Reinitialize the adjoint states after an event occurrence.

**Parameters**

- **which** – identifier of the backwards problem
- **yQB0** – new adjoint quadrature state

*realtype* **gett**() const

current solver timepoint

**Returns** t

*realtype* **getCpuTime**() const

Reads out the CPU time needed for forward solve.

**Returns** cpu\_time

*realtype* **getCpuTimeB**() const

Reads out the CPU time needed for backward solve.

**Returns** cpu\_timeB

int **nx**() const

number of states with which the solver was initialized

**Returns** x.getLength()

int **nplist**() const

number of parameters with which the solver was initialized

**Returns** sx.getLength()

int **nquad**() const

number of quadratures with which the solver was initialized

**Returns** xQB.getLength()

inline bool **computingFSA**() const

check if FSA is being computed

**Returns** flag

inline bool **computingASA**() const

check if ASA is being computed

**Returns** flag

void **resetDiagnosis()** const

Resets vectors containing diagnosis information.

void **storeDiagnosis()** const

Stores diagnosis information from solver memory block for forward problem.

void **storeDiagnosisB**(int which) const

Stores diagnosis information from solver memory block for backward problem.

**Parameters** **which** – identifier of the backwards problem

inline std::vector<int> const &**getNumSteps()** const

Accessor ns.

**Returns** ns

inline std::vector<int> const &**getNumStepsB()** const

Accessor nsB.

**Returns** nsB

inline std::vector<int> const &**getNumRhsEvals()** const

Accessor nrhs.

**Returns** nrhs

inline std::vector<int> const &**getNumRhsEvalsB()** const

Accessor nrhsB.

**Returns** nrhsB

inline std::vector<int> const &**getNumErrTestFails()** const

Accessor netf.

**Returns** netf

inline std::vector<int> const &**getNumErrTestFailsB()** const

Accessor netfB.

**Returns** netfB

inline std::vector<int> const &**getNumNonlinSolvConvFails()** const

Accessor nnlscf.

**Returns** nnlscf

inline std::vector<int> const &**getNumNonlinSolvConvFailsB()** const

Accessor nnlscfB.

**Returns** nnlscfB

inline std::vector<int> const &**getLastOrder()** const

Accessor order.

**Returns** order

inline bool **getNewtonStepSteadyStateCheck()** const

Returns how convergence checks for steadystate computation are performed. If activated, convergence checks are limited to every 25 steps in the simulation solver to limit performance impact.

**Returns** boolean flag indicating newton step (true) or the right hand side (false)

inline bool **getSensiSteadyStateCheck**() const

Returns how convergence checks for steadystate computation are performed.

**Returns** boolean flag indicating state and sensitivity equations (true) or only state variables (false).

inline void **setNewtonStepSteadyStateCheck**(bool flag)

Sets how convergence checks for steadystate computation are performed.

**Parameters** **flag** – boolean flag to pick newton step (true) or the right hand side (false, default)

inline void **setSensiSteadyStateCheck**(bool flag)

Sets for which variables convergence checks for steadystate computation are performed.

**Parameters** **flag** – boolean flag to pick state and sensitivity equations (true, default) or only state variables (false).

## Public Members

*Logger* \***logger** = nullptr

logger

## Protected Functions

virtual void **setStopTime**(*realtype* tstop) const = 0

Sets a timepoint at which the simulation will be stopped.

**Parameters** **tstop** – timepoint until which simulation should be performed

virtual int **solve**(*realtype* tout, int itask) const = 0

Solves the forward problem until a predefined timepoint.

**Parameters**

- **tout** – timepoint until which simulation should be performed
- **itask** – task identifier, can be CV\_NORMAL or CV\_ONE\_STEP

**Returns** status flag indicating success of execution

virtual int **solveF**(*realtype* tout, int itask, int \*ncheckPtr) const = 0

Solves the forward problem until a predefined timepoint (adjoint only)

**Parameters**

- **tout** – timepoint until which simulation should be performed
- **itask** – task identifier, can be CV\_NORMAL or CV\_ONE\_STEP
- **ncheckPtr** – pointer to a number that counts the internal checkpoints

**Returns** status flag indicating success of execution

virtual void **reInitPostProcessF**(*realtype* tnext) const = 0

reInitPostProcessF postprocessing of the solver memory after a discontinuity in the forward problem

**Parameters** **tnext** – next timepoint (defines integration direction)

virtual void **reInitPostProcessB**(*realtype* tnext) const = 0

reInitPostProcessB postprocessing of the solver memory after a discontinuity in the backward problem

**Parameters** **tnext** – next timepoint (defines integration direction)

virtual void **getSens**() const = 0

extracts the state sensitivity at the current timepoint from solver memory and writes it to the sx member variable

virtual void **getB**(int which) const = 0

extracts the adjoint state at the current timepoint from solver memory and writes it to the xB member variable

**Parameters** **which** – index of the backwards problem

virtual void **getQuadB**(int which) const = 0

extracts the adjoint quadrature state at the current timepoint from solver memory and writes it to the xQB member variable

**Parameters** **which** – index of the backwards problem

virtual void **getQuad**(*realtype* &t) const = 0

extracts the quadrature at the current timepoint from solver memory and writes it to the xQ member variable

**Parameters** **t** – timepoint for quadrature extraction

virtual void **init**(*realtype* t0, *AmiVector* const &x0, *AmiVector* const &dx0) const = 0

Initializes the states at the specified initial timepoint.

**Parameters**

- **t0** – initial timepoint
- **x0** – initial states
- **dx0** – initial derivative states

virtual void **initSteadystate**(*realtype* t0, *AmiVector* const &x0, *AmiVector* const &dx0) const = 0

Initializes the states at the specified initial timepoint.

**Parameters**

- **t0** – initial timepoint
- **x0** – initial states
- **dx0** – initial derivative states

virtual void **sensInit1**(*AmiVectorArray* const &sx0, *AmiVectorArray* const &sdx0) const = 0

Initializes the forward sensitivities.

**Parameters**

- **sx0** – initial states sensitivities
- **sdx0** – initial derivative states sensitivities

virtual void **binit**(int which, *realtype* tf, *AmiVector* const &xB0, *AmiVector* const &dxB0) const = 0

Initialize the adjoint states at the specified final timepoint.

**Parameters**

- **which** – identifier of the backwards problem
- **tf** – final timepoint

- **xB0** – initial adjoint state
- **dxB0** – initial adjoint derivative state

virtual void **qbinit**(int which, *AmiVector* const &xQB0) const = 0

Initialize the quadrature states at the specified final timepoint.

**Parameters**

- **which** – identifier of the backwards problem
- **xQB0** – initial adjoint quadrature state

virtual void **rootInit**(int ne) const = 0

Initializes the rootfinding for events.

**Parameters** **ne** – number of different events

void **initializeNonLinearSolverSens**(*Model* const \*model) const

Initialize non-linear solver for sensitivities.

**Parameters** **model** – *Model* instance

virtual void **setDenseJacFn**() const = 0

Set the dense Jacobian function.

virtual void **setSparseJacFn**() const = 0

sets the sparse Jacobian function

virtual void **setBandJacFn**() const = 0

sets the banded Jacobian function

virtual void **setJacTimesVecFn**() const = 0

sets the Jacobian vector multiplication function

virtual void **setDenseJacFnB**(int which) const = 0

sets the dense Jacobian function

**Parameters** **which** – identifier of the backwards problem

virtual void **setSparseJacFnB**(int which) const = 0

sets the sparse Jacobian function

**Parameters** **which** – identifier of the backwards problem

virtual void **setBandJacFnB**(int which) const = 0

sets the banded Jacobian function

**Parameters** **which** – identifier of the backwards problem

virtual void **setJacTimesVecFnB**(int which) const = 0

sets the Jacobian vector multiplication function

**Parameters** **which** – identifier of the backwards problem

virtual void **setSparseJacFn\_ss**() const = 0

sets the sparse Jacobian function for backward steady state case

virtual void **allocateSolver**() const = 0

Create specifies solver method and initializes solver memory for the forward problem.



virtual void **setSStolerances**(double rtol, double atol) const = 0  
 sets scalar relative and absolute tolerances for the forward problem

**Parameters**

- **rtol** – relative tolerances
- **atol** – absolute tolerances

virtual void **setSensSStolerances**(double rtol, double const \*atol) const = 0  
 activates sets scalar relative and absolute tolerances for the sensitivity variables

**Parameters**

- **rtol** – relative tolerances
- **atol** – array of absolute tolerances for every sensitivity variable

virtual void **setSensErrCon**(bool error\_corr) const = 0  
 SetSensErrCon specifies whether error control is also enforced for sensitivities for the forward problem

**Parameters** **error\_corr** – activation flag

virtual void **setQuadErrConB**(int which, bool flag) const = 0  
 Specifies whether error control is also enforced for the backward quadrature problem.

**Parameters**

- **which** – identifier of the backwards problem
- **flag** – activation flag

virtual void **setQuadErrCon**(bool flag) const = 0  
 Specifies whether error control is also enforced for the forward quadrature problem.

**Parameters** **flag** – activation flag

virtual void **setErrHandlerFn**() const = 0  
 Attaches the error handler function (errMsgIdAndTxt) to the solver.

virtual void **setUserData**() const = 0  
 Attaches the user data to the forward problem.

virtual void **setUserDataB**(int which) const = 0  
 attaches the user data to the backward problem

**Parameters** **which** – identifier of the backwards problem

virtual void **setMaxNumSteps**(long int mxsteps) const = 0  
 specifies the maximum number of steps for the forward problem

---

**Note:** in contrast to the SUNDIALS method, this sets the overall maximum, not the maximum between output times.

---

**Parameters** **mxsteps** – number of steps

virtual void **setMaxNumStepsB**(int which, long int mxstepsB) const = 0  
specifies the maximum number of steps for the forward problem

---

**Note:** in contrast to the SUNDIALS method, this sets the overall maximum, not the maximum between output times.

---

**Parameters**

- **which** – identifier of the backwards problem
- **mxstepsB** – number of steps

virtual void **setStabLimDet**(int stldet) const = 0  
activates stability limit detection for the forward problem

**Parameters** **stldet** – flag for stability limit detection (TRUE or FALSE)

virtual void **setStabLimDetB**(int which, int stldet) const = 0  
activates stability limit detection for the backward problem

**Parameters**

- **which** – identifier of the backwards problem
- **stldet** – flag for stability limit detection (TRUE or FALSE)

virtual void **setId**(*Model* const \*model) const = 0  
specify algebraic/differential components (DAE only)

**Parameters** **model** – model specification

virtual void **setSuppressAlg**(bool flag) const = 0  
deactivates error control for algebraic components (DAE only)

**Parameters** **flag** – deactivation flag

virtual void **setSensParams**(*realtype* const \*p, *realtype* const \*pbar, int const \*plist) const = 0  
specifies the scaling and indexes for sensitivity computation

**Parameters**

- **p** – parameters
- **pbar** – parameter scaling constants
- **plist** – parameter index list

virtual void **getDky**(*realtype* t, int k) const = 0  
interpolates the (derivative of the) solution at the requested timepoint

**Parameters**

- **t** – timepoint
- **k** – derivative order

virtual void **getDkyB**(*realtype* t, int k, int which) const = 0  
interpolates the (derivative of the) solution at the requested timepoint

**Parameters**

- **t** – timepoint

- **k** – derivative order
- **which** – index of backward problem

virtual void **getSensDky**(*realtype* t, int k) const = 0

interpolates the (derivative of the) solution at the requested timepoint

#### Parameters

- **t** – timepoint
- **k** – derivative order

virtual void **getQuadDkyB**(*realtype* t, int k, int which) const = 0

interpolates the (derivative of the) solution at the requested timepoint

#### Parameters

- **t** – timepoint
- **k** – derivative order
- **which** – index of backward problem

virtual void **getQuadDky**(*realtype* t, int k) const = 0

interpolates the (derivative of the) solution at the requested timepoint

#### Parameters

- **t** – timepoint
- **k** – derivative order

virtual void **adjInit**() const = 0

initializes the adjoint problem

virtual void **quadInit**(*AmiVector* const &xQ0) const = 0

initializes the quadratures

**Parameters** **xQ0** – vector with initial values for xQ

virtual void **allocateSolverB**(int \*which) const = 0

Specifies solver method and initializes solver memory for the backward problem.

**Parameters** **which** – identifier of the backwards problem

virtual void **setSStolerancesB**(int which, *realtype* relTolB, *realtype* absTolB) const = 0

sets relative and absolute tolerances for the backward problem

#### Parameters

- **which** – identifier of the backwards problem
- **relTolB** – relative tolerances
- **absTolB** – absolute tolerances

virtual void **quadSStolerancesB**(int which, *realtype* reltolQB, *realtype* abstolQB) const = 0

sets relative and absolute tolerances for the quadrature backward problem

#### Parameters

- **which** – identifier of the backwards problem
- **reltolQB** – relative tolerances
- **abstolQB** – absolute tolerances

virtual void **quadSStolerances**(*realtype* reltolQB, *realtype* abstolQB) const = 0

sets relative and absolute tolerances for the quadrature problem

**Parameters**

- **reltolQB** – relative tolerances
- **abstolQB** – absolute tolerances

virtual void **getNumSteps**(void const \*ami\_mem, long int \*numsteps) const = 0

reports the number of solver steps

**Parameters**

- **ami\_mem** – pointer to the solver memory instance (can be from forward or backward problem)
- **numsteps** – output array

virtual void **getNumRhsEvals**(void const \*ami\_mem, long int \*numrhsevals) const = 0

reports the number of right hand evaluations

**Parameters**

- **ami\_mem** – pointer to the solver memory instance (can be from forward or backward problem)
- **numrhsevals** – output array

virtual void **getNumErrTestFails**(void const \*ami\_mem, long int \*numerrtestfails) const = 0

reports the number of local error test failures

**Parameters**

- **ami\_mem** – pointer to the solver memory instance (can be from forward or backward problem)
- **numerrtestfails** – output array

virtual void **getNumNonlinSolvConvFails**(void const \*ami\_mem, long int \*numnonlinsolvconvfails) const = 0

reports the number of nonlinear convergence failures

**Parameters**

- **ami\_mem** – pointer to the solver memory instance (can be from forward or backward problem)
- **numnonlinsolvconvfails** – output array

virtual void **getLastOrder**(void const \*ami\_mem, int \*order) const = 0

Reports the order of the integration method during the last internal step.

**Parameters**

- **ami\_mem** – pointer to the solver memory instance (can be from forward or backward problem)
- **order** – output array

void **initializeLinearSolver**(*Model* const \*model) const

Initializes and sets the linear solver for the forward problem.

**Parameters** **model** – pointer to the model object

void **initializeNonLinearSolver**() const  
 Sets the non-linear solver.

virtual void **setLinearSolver**() const = 0  
 Sets the linear solver for the forward problem.

virtual void **setLinearSolverB**(int which) const = 0  
 Sets the linear solver for the backward problem.  
**Parameters** **which** – index of the backward problem

virtual void **setNonLinearSolver**() const = 0  
 Set the non-linear solver for the forward problem.

virtual void **setNonLinearSolverB**(int which) const = 0  
 Set the non-linear solver for the backward problem.  
**Parameters** **which** – index of the backward problem

virtual void **setNonLinearSolverSens**() const = 0  
 Set the non-linear solver for sensitivities.

void **initializeLinearSolverB**(*Model* const \*model, int which) const  
 Initializes the linear solver for the backward problem.  
**Parameters**

- **model** – pointer to the model object
- **which** – index of the backward problem

void **initializeNonLinearSolverB**(int which) const  
 Initializes the non-linear solver for the backward problem.  
**Parameters** **which** – index of the backward problem

virtual *Model* const \***getModel**() const = 0  
 Accessor function to the model stored in the user data  
**Returns** user data model

bool **getInitDone**() const  
 checks whether memory for the forward problem has been allocated  
**Returns** proxy for solverMemory->(cv|ida)\_MallocDone

bool **getSensInitDone**() const  
 checks whether memory for forward sensitivities has been allocated  
**Returns** proxy for solverMemory->(cv|ida)\_SensMallocDone

bool **getAdjInitDone**() const  
 checks whether memory for forward interpolation has been allocated  
**Returns** proxy for solverMemory->(cv|ida)\_adjMallocDone

bool **getInitDoneB**(int which) const  
 checks whether memory for the backward problem has been allocated  
**Parameters** **which** – adjoint problem index  
**Returns** proxy for solverMemoryB->(cv|ida)\_MallocDone

bool **getQuadInitDoneB**(int which) const  
checks whether memory for backward quadratures has been allocated  
**Parameters** **which** – adjoint problem index  
**Returns** proxy for solverMemoryB->(cv|ida)\_QuadMallocDone

bool **getQuadInitDone**() const  
checks whether memory for quadratures has been allocated  
**Returns** proxy for solverMemory->(cv|ida)\_QuadMallocDone

virtual void **diag**() const = 0  
attaches a diagonal linear solver to the forward problem

virtual void **diagB**(int which) const = 0  
attaches a diagonal linear solver to the backward problem  
**Parameters** **which** – identifier of the backwards problem

void **resetMutableMemory**(int nx, int nplist, int nquad) const  
resets solverMemory and solverMemoryB  
**Parameters**

- **nx** – new number of state variables
- **nplist** – new number of sensitivity parameters
- **nquad** – new number of quadratures (only differs from nplist for higher order sensitivity computation)

virtual void **\*getAdjBmem**(void \*ami\_mem, int which) const = 0  
Retrieves the solver memory instance for the backward problem.  
**Parameters**

- **which** – identifier of the backwards problem
- **ami\_mem** – pointer to the forward solver memory instance

**Returns** A (void \*) pointer to the CVODES memory allocated for the backward problem.

void **applyTolerances**() const  
updates solver tolerances according to the currently specified member variables

void **applyTolerancesFSA**() const  
updates FSA solver tolerances according to the currently specified member variables

void **applyTolerancesASA**(int which) const  
updates ASA solver tolerances according to the currently specified member variables  
**Parameters** **which** – identifier of the backwards problem

void **applyQuadTolerancesASA**(int which) const  
updates ASA quadrature solver tolerances according to the currently specified member variables  
**Parameters** **which** – identifier of the backwards problem

void **applyQuadTolerances**() const  
updates quadrature solver tolerances according to the currently specified member variables

void **applySensitivityTolerances**() const  
 updates all sensitivity solver tolerances according to the currently specified member variables

void **setInitDone**() const  
 sets that memory for the forward problem has been allocated

void **setSensInitDone**() const  
 sets that memory for forward sensitivities has been allocated

void **setSensInitOff**() const  
 sets that memory for forward sensitivities has not been allocated

void **setAdjInitDone**() const  
 sets that memory for forward interpolation has been allocated

void **setInitDoneB**(int which) const  
 sets that memory for the backward problem has been allocated  
**Parameters** **which** – adjoint problem index

void **setQuadInitDoneB**(int which) const  
 sets that memory for backward quadratures has been allocated  
**Parameters** **which** – adjoint problem index

void **setQuadInitDone**() const  
 sets that memory for quadratures has been allocated

void **checkSensitivityMethod**(const *SensitivityMethod* sensi\_meth, bool preequilibration) const  
 Sets sensitivity method (for simulation or preequilibration)  
**Parameters**

- **sensi\_meth** – new value for sensi\_meth[\_preeq]
- **preequilibration** – flag indicating preequilibration or simulation

### Protected Attributes

mutable std::unique\_ptr<void, std::function<void(void\*)>> **solver\_memory\_**  
 pointer to solver memory block

mutable std::vector<std::unique\_ptr<void, std::function<void(void\*)>>> **solver\_memory\_B\_**  
 pointer to solver memory block

mutable *user\_data\_type* **user\_data**  
 Sundials user\_data

*InternalSensitivityMethod* **ism\_** = {*InternalSensitivityMethod::simultaneous*}

internal sensitivity method flag used to select the sensitivity solution method. Only applies for Forward Sensitivities.

*LinearMultistepMethod* **lmm\_** = {*LinearMultistepMethod::BDF*}

specifies the linear multistep method.

*NonlinearSolverIteration* **iter\_** = {*NonlinearSolverIteration::newton*}

specifies the type of nonlinear solver iteration

*InterpolationType* **interp\_type\_** = {*InterpolationType::polynomial*}

interpolation type for the forward problem solution which is then used for the backwards problem.

long int **maxsteps\_** = {10000}

maximum number of allowed integration steps

std::chrono::duration<double, std::ratio<1>> **maxtime\_** = {0}

Maximum CPU-time for integration in seconds

mutable *CpuTimer* **simulation\_timer\_**

Time at which solver timer was started

mutable std::unique\_ptr<*SUNLinSolWrapper*> **linear\_solver\_**

linear solver for the forward problem

mutable std::unique\_ptr<*SUNLinSolWrapper*> **linear\_solver\_B\_**

linear solver for the backward problem

mutable std::unique\_ptr<*SUNNonLinSolWrapper*> **non\_linear\_solver\_**

non-linear solver for the forward problem

mutable std::unique\_ptr<*SUNNonLinSolWrapper*> **non\_linear\_solver\_B\_**

non-linear solver for the backward problem

mutable std::unique\_ptr<*SUNNonLinSolWrapper*> **non\_linear\_solver\_sens\_**

non-linear solver for the sensitivities

mutable bool **solver\_was\_called\_F\_** = {false}

flag indicating whether the forward solver has been called

mutable bool **solver\_was\_called\_B\_** = {false}

flag indicating whether the backward solver has been called

mutable *AmiVector* **x\_** = {0}

state (dimension: nx\_solver)

mutable *AmiVector* **dky\_** = {0}

state interface variable (dimension: nx\_solver)

mutable *AmiVector* **dx\_** = {0}

state derivative dummy (dimension: nx\_solver)



mutable *AmiVectorArray* **sx\_** = {0, 0}  
state sensitivities interface variable (dimension: nx\_solver x nplist)

mutable *AmiVectorArray* **sdx\_** = {0, 0}  
state derivative sensitivities dummy (dimension: nx\_solver x nplist)

mutable *AmiVector* **xB\_** = {0}  
adjoint state interface variable (dimension: nx\_solver)

mutable *AmiVector* **dxB\_** = {0}  
adjoint derivative dummy variable (dimension: nx\_solver)

mutable *AmiVector* **xQB\_** = {0}  
adjoint quadrature interface variable (dimension: nJ x nplist)

mutable *AmiVector* **xQ\_** = {0}  
forward quadrature interface variable (dimension: nx\_solver)

mutable *realtype* **t\_** = {std::nan("")}  
integration time of the forward problem

mutable bool **force\_reinit\_postprocess\_F\_** = {false}  
flag to force reInitPostProcessF before next call to solve

mutable bool **force\_reinit\_postprocess\_B\_** = {false}  
flag to force reInitPostProcessB before next call to solveB

mutable bool **sens\_initialized\_** = {false}  
flag indicating whether sensInit1 was called

## Friends

template<class **Archive**>  
friend void **serialize**(*Archive* &ar, *Solver* &s, unsigned int version)  
Serialize *Solver* (see boost::serialization::serialize)

### Parameters

- **ar** – Archive to serialize to
- **s** – Data to serialize
- **version** – Version number

friend bool **operator==(***Solver* const &a, *Solver* const &b)  
Check equality of data members excluding solver memory.

### Parameters

- **a** –

- **b** –

**Returns**

## Class SteadystateProblem

- Defined in file\_include\_amici\_steadystateproblem.h

## Class Documentation

class **SteadystateProblem**

The *SteadystateProblem* class solves a steady-state problem using Newton's method and falls back to integration on failure.

### Public Functions

explicit **SteadystateProblem**(*Solver* const &solver, *Model* const &model)

constructor

#### Parameters

- **solver** – *Solver* instance
- **model** – *Model* instance

void **workSteadyStateProblem**(*Solver* const &solver, *Model* &model, int it)

Handles steady state computation in the forward case: tries to determine the steady state of the ODE system and computes steady state sensitivities if requested.

#### Parameters

- **solver** – pointer to the solver object
- **model** – pointer to the model object
- **it** – integer with the index of the current time step

void **workSteadyStateBackwardProblem**(*Solver* const &solver, *Model* &model, *BackwardProblem* const \*bwd)

Integrates over the adjoint state backward in time by solving a linear system of equations, which gives the analytical solution. Computes the gradient via adjoint steady state sensitivities

#### Parameters

- **solver** – pointer to the solver object
- **model** – pointer to the model object
- **bwd** – backward problem

inline *SimulationState* const &**getFinalSimulationState**() const

Returns the stored *SimulationState*.

**Returns** stored *SimulationState*

inline *AmiVector* const &**getEquilibrationQuadratures**() const

Returns the quadratures from pre- or postequilibration.

**Returns** xQB Vector with quadratures

inline *AmiVector* const &**getState**() const

Returns state at steadystate.

**Returns** x

inline *AmiVectorArray* const &**getStateSensitivity**() const

Returns state sensitivity at steadystate.

**Returns** sx

inline std::vector<*realtype*> const &**getDJydx**() const

Accessor for dJydx.

**Returns** dJydx

inline double **getCPUTime**() const

Accessor for run\_time of the forward problem.

**Returns** run\_time

inline double **getCPUTimeB**() const

Accessor for run\_time of the backward problem.

**Returns** run\_time

inline std::vector<*SteadyStateStatus*> const &**getSteadyStateStatus**() const

Accessor for steady\_state\_status.

**Returns** steady\_state\_status

inline *realtype* **getSteadyStateTime**() const

Get model time at which steadystate was found through simulation.

**Returns** t

inline *realtype* **getResidualNorm**() const

Accessor for wrms.

**Returns** wrms

inline std::vector<int> const &**getNumSteps**() const

Accessor for numsteps.

**Returns** numsteps

inline int **getNumStepsB**() const

Accessor for numstepsB.

**Returns** numstepsB

void **getAdjointUpdates**(*Model* &model, *ExpData* const &edata)

computes adjoint updates dJydx according to provided model and expdata

**Parameters**

- **model** – *Model* instance
- **edata** – experimental data

inline *AmiVector* const &**getAdjointState**() const

Return the adjoint state.

**Returns** xB adjoint state

inline *AmiVector* const &**getAdjointQuadrature**() const

Accessor for xQB.

**Returns** xQB

inline bool **hasQuadrature**() const

Accessor for hasQuadrature\_.

**Returns** hasQuadrature\_

bool **checkSteadyStateSuccess**() const

computes adjoint updates dJydx according to provided model and expdata

**Returns** convergence of steady state solver

## Class SUNLinSolBand

- Defined in file\_include\_amici\_sundials\_linsol\_wrapper.h

## Inheritance Relationships

### Base Type

- public amici::SUNLinSolWrapper (*Class SUNLinSolWrapper*)

## Class Documentation

class **SUNLinSolBand** : public amici::SUNLinSolWrapper

SUNDIALS band direct solver.

### Public Functions

**SUNLinSolBand**(N\_Vector x, SUNMatrix A)

Create solver using existing matrix A without taking ownership of A.

#### Parameters

- **x** – A template for cloning vectors needed within the solver.
- **A** – square matrix

**SUNLinSolBand**(*AmiVector* const &x, int ubw, int lbw)

Create new band solver and matrix A.

#### Parameters

- **x** – A template for cloning vectors needed within the solver.
- **ubw** – upper bandwidth of band matrix A

- **lbw** – lower bandwidth of band matrix A

virtual SUNMatrix **getMatrix**() const override

Get the matrix A (matrix solvers only).

**Returns** A

## Class SUNLinSolDense

- Defined in file\_include\_amici\_sundials\_linsol\_wrapper.h

## Inheritance Relationships

### Base Type

- public amici::SUNLinSolWrapper (*Class SUNLinSolWrapper*)

## Class Documentation

class **SUNLinSolDense** : public amici::SUNLinSolWrapper  
SUNDIALS dense direct solver.

### Public Functions

explicit **SUNLinSolDense**(*AmiVector* const &x)

Create dense solver.

**Parameters** **x** – A template for cloning vectors needed within the solver.

virtual SUNMatrix **getMatrix**() const override

Get the matrix A (matrix solvers only).

**Returns** A

## Class SUNLinSolKLU

- Defined in file\_include\_amici\_sundials\_linsol\_wrapper.h

## Inheritance Relationships

### Base Type

- public amici::SUNLinSolWrapper (*Class SUNLinSolWrapper*)

## Class Documentation

class **SUNLinSolKLU** : public amici::SUNLinSolWrapper  
SUNDIALS KLU sparse direct solver.

### Public Types

enum class **StateOrdering**

KLU state reordering (different from SuperLUMT ordering!)

*Values:*

enumerator **AMD**

enumerator **COLAMD**

enumerator **natural**

### Public Functions

**SUNLinSolKLU**(N\_Vector x, SUNMatrix A)

Create KLU solver with given matrix.

#### Parameters

- **x** – A template for cloning vectors needed within the solver.
- **A** – sparse matrix

**SUNLinSolKLU**(AmiVector const &x, int nnz, int sparsetype, StateOrdering ordering)

Create KLU solver and matrix to operate on.

#### Parameters

- **x** – A template for cloning vectors needed within the solver.
- **nnz** – Number of non-zeros in matrix A
- **sparsetype** – Sparse matrix type (CSC\_MAT, CSR\_MAT)
- **ordering** –

virtual SUNMatrix **getMatrix**() const override

Get the matrix A (matrix solvers only).

#### Returns

A

void **reInit**(int nnz, int reinit\_type)

Reinitializes memory and flags for a new factorization (symbolic and numeric) to be conducted at the next solver setup call.

For more details see sunlinsol/sunlinsol\_klu.h

#### Parameters

- **nnz** – Number of non-zeros

- **reinit\_type** – SUNKLU\_REINIT\_FULL or SUNKLU\_REINIT\_PARTIAL

void **setOrdering**(*StateOrdering* ordering)

Sets the ordering used by KLU for reducing fill in the linear solve.

**Parameters** **ordering** –

## Class SUNLinSolPCG

- Defined in file\_include\_amici\_sundials\_linsol\_wrapper.h

## Inheritance Relationships

### Base Type

- public amici::SUNLinSolWrapper (*Class SUNLinSolWrapper*)

## Class Documentation

class **SUNLinSolPCG** : public amici::SUNLinSolWrapper

SUNDIALS scaled preconditioned CG (Conjugate Gradient method) (PCG) solver.

### Public Functions

**SUNLinSolPCG**(N\_Vector y, int pretype, int maxl)

Create PCG solver.

#### Parameters

- **y** –
- **pretype** – Preconditioner type (PREC\_NONE, PREC\_LEFT, PREC\_RIGHT, PREC\_BOTH)
- **maxl** – Maximum number of solver iterations

int **setATimes**(void \*A\_data, ATimesFn ATimes)

Sets the function pointer for ATimes (see sundials/sundials\_linearsolver.h).

#### Parameters

- **A\_data** –
- **ATimes** –

#### Returns

int **setPreconditioner**(void \*P\_data, PSetupFn Pset, PSolveFn Psol)

Sets function pointers for PSetup and PSolve routines inside of iterative linear solver objects (see sundials/sundials\_linearsolver.h).

#### Parameters

- **P\_data** –

- **Pset** –
- **Psol** –

**Returns**

int **setScalingVectors**(N\_Vector s, N\_Vector nul)

Sets pointers to left/right scaling vectors for the linear system solve (see sundials/sundials\_linearsolver.h).

**Parameters**

- **s** –
- **nul** –

**Returns**

int **getNumIters**() const

Returns the number of linear iterations performed in the last ‘Solve’ call.

**Returns** Number of iterations

*realtype* **getResNorm**() const

Returns the final residual norm from the last ‘Solve’ call.

**Returns** residual norm

N\_Vector **getResid**() const

Get preconditioned initial residual (see sundials/sundials\_linearsolver.h).

**Returns**

## Class SUNLinSolSPBCGS

- Defined in file\_include\_amici\_sundials\_linsol\_wrapper.h

## Inheritance Relationships

### Base Type

- public amici::SUNLinSolWrapper (*Class SUNLinSolWrapper*)

## Class Documentation

class **SUNLinSolSPBCGS** : public amici::SUNLinSolWrapper

SUNDIALS scaled preconditioned Bi-CGStab (Bi-Conjugate Gradient Stable method) (SPBCGS) solver.



## Public Functions

explicit **SUNLinSolSPBCGS**(N\_Vector x, int pretype = PREC\_NONE, int maxl = SUNSPBCGS\_MAXL\_DEFAULT)

*SUNLinSolSPBCGS.*

### Parameters

- **x** – A template for cloning vectors needed within the solver.
- **pretype** – Preconditioner type (PREC\_NONE, PREC\_LEFT, PREC\_RIGHT, PREC\_BOTH)
- **maxl** – Maximum number of solver iterations

explicit **SUNLinSolSPBCGS**(*AmiVector* const &x, int pretype = PREC\_NONE, int maxl = SUNSPBCGS\_MAXL\_DEFAULT)

*SUNLinSolSPBCGS.*

### Parameters

- **x** – A template for cloning vectors needed within the solver.
- **pretype** – Preconditioner type (PREC\_NONE, PREC\_LEFT, PREC\_RIGHT, PREC\_BOTH)
- **maxl** – Maximum number of solver iterations

int **setATimes**(void \*A\_data, ATimesFn ATimes)

Sets the function pointer for ATimes (see sundials/sundials\_linearsolver.h).

### Parameters

- **A\_data** –
- **ATimes** –

### Returns

int **setPreconditioner**(void \*P\_data, PSetupFn Pset, PSolveFn Psol)

Sets function pointers for PSetup and PSolve routines inside of iterative linear solver objects (see sundials/sundials\_linearsolver.h).

### Parameters

- **P\_data** –
- **Pset** –
- **Psol** –

### Returns

int **setScalingVectors**(N\_Vector s, N\_Vector nul)

Sets pointers to left/right scaling vectors for the linear system solve (see sundials/sundials\_linearsolver.h).

### Parameters

- **s** –
- **nul** –

### Returns

int **getNumIters()** const

Returns the number of linear iterations performed in the last ‘Solve’ call.

**Returns** Number of iterations

*realtype* **getResNorm()** const

Returns the final residual norm from the last ‘Solve’ call.

**Returns** residual norm

N\_Vector **getResid()** const

Get preconditioned initial residual (see sundials/sundials\_linearsolver.h).

**Returns**

## Class SUNLinSolSPFGMR

- Defined in file\_include\_amici\_sundials\_linsol\_wrapper.h

## Inheritance Relationships

### Base Type

- public amici::SUNLinSolWrapper (*Class SUNLinSolWrapper*)

## Class Documentation

class **SUNLinSolSPFGMR** : public amici::SUNLinSolWrapper

SUNDIALS scaled preconditioned FGMRES (Flexible Generalized Minimal Residual method) (SPFGMR) solver.

### Public Functions

**SUNLinSolSPFGMR**(*AmiVector* const &x, int pretype, int maxl)

*SUNLinSolSPFGMR*.

#### Parameters

- **x** – A template for cloning vectors needed within the solver.
- **pretype** – Preconditioner type (PREC\_NONE, PREC\_LEFT, PREC\_RIGHT, PREC\_BOTH)
- **maxl** – Maximum number of solver iterations

int **setATimes**(void \*A\_data, ATimesFn ATimes)

Sets the function pointer for ATimes (see sundials/sundials\_linearsolver.h).

#### Parameters

- **A\_data** –
- **ATimes** –

#### Returns

int **setPreconditioner**(void \*P\_data, PSetupFn Pset, PSolveFn Psol)

Sets function pointers for PSetup and PSolve routines inside of iterative linear solver objects (see sundials/sundials\_linearsolver.h).

**Parameters**

- **P\_data** –
- **Pset** –
- **Psol** –

**Returns**

int **setScalingVectors**(N\_Vector s, N\_Vector nul)

Sets pointers to left/right scaling vectors for the linear system solve (see sundials/sundials\_linearsolver.h).

**Parameters**

- **s** –
- **nul** –

**Returns**

int **getNumIters**() const

Returns the number of linear iterations performed in the last ‘Solve’ call.

**Returns** Number of iterations

*realtype* **getResNorm**() const

Returns the final residual norm from the last ‘Solve’ call.

**Returns** residual norm

N\_Vector **getResid**() const

Get preconditioned initial residual (see sundials/sundials\_linearsolver.h).

**Returns**

## Class SUNLinSolSPGMR

- Defined in file\_include\_amici\_sundials\_linsol\_wrapper.h

## Inheritance Relationships

### Base Type

- public amici::SUNLinSolWrapper (*Class SUNLinSolWrapper*)

## Class Documentation

class **SUNLinSolSPGMR** : public amici::SUNLinSolWrapper

SUNDIALS scaled preconditioned GMRES (Generalized Minimal Residual method) solver (SPGMR).

### Public Functions

explicit **SUNLinSolSPGMR**(AmiVector const &x, int pretype = PREC\_NONE, int maxl = SUNSPGMR\_MAXL\_DEFAULT)

Create SPGMR solver.

#### Parameters

- **x** – A template for cloning vectors needed within the solver.
- **pretype** – Preconditioner type (PREC\_NONE, PREC\_LEFT, PREC\_RIGHT, PREC\_BOTH)
- **maxl** – Maximum number of solver iterations

int **setATimes**(void \*A\_data, ATimesFn ATimes)

Sets the function pointer for ATimes (see sundials/sundials\_linearsolver.h).

#### Parameters

- **A\_data** –
- **ATimes** –

#### Returns

int **setPreconditioner**(void \*P\_data, PSetupFn Pset, PSolveFn Psol)

Sets function pointers for Pset and Psolve routines inside of iterative linear solver objects (see sundials/sundials\_linearsolver.h).

#### Parameters

- **P\_data** –
- **Pset** –
- **Psol** –

#### Returns

int **setScalingVectors**(N\_Vector s, N\_Vector nul)

Sets pointers to left/right scaling vectors for the linear system solve (see sundials/sundials\_linearsolver.h).

#### Parameters

- **s** –
- **nul** –

#### Returns

int **getNumIters**() const

Returns the number of linear iterations performed in the last ‘Solve’ call.

**Returns** Number of iterations

*realtype* **getResNorm()** const

Returns the final residual norm from the last ‘Solve’ call.

**Returns** residual norm

N\_Vector **getResid()** const

Get preconditioned initial residual (see sundials/sundials\_linearsolver.h).

**Returns**

## Class SUNLinSolSPTFQMR

- Defined in file\_include\_amici\_sundials\_linsol\_wrapper.h

## Inheritance Relationships

### Base Type

- public amici::SUNLinSolWrapper (*Class SUNLinSolWrapper*)

## Class Documentation

class **SUNLinSolSPTFQMR** : public amici::SUNLinSolWrapper

SUNDIALS scaled preconditioned TFQMR (Transpose-Free Quasi-Minimal Residual method) (SPTFQMR) solver.

### Public Functions

explicit **SUNLinSolSPTFQMR**(N\_Vector x, int pretype = PREC\_NONE, int maxl = SUNSPTFQMR\_MAXL\_DEFAULT)

Create SPTFQMR solver.

#### Parameters

- **x** – A template for cloning vectors needed within the solver.
- **pretype** – Preconditioner type (PREC\_NONE, PREC\_LEFT, PREC\_RIGHT, PREC\_BOTH)
- **maxl** – Maximum number of solver iterations

explicit **SUNLinSolSPTFQMR**(AmiVector const &x, int pretype = PREC\_NONE, int maxl = SUNSPTFQMR\_MAXL\_DEFAULT)

Create SPTFQMR solver.

#### Parameters

- **x** – A template for cloning vectors needed within the solver.
- **pretype** – Preconditioner type (PREC\_NONE, PREC\_LEFT, PREC\_RIGHT, PREC\_BOTH)
- **maxl** – Maximum number of solver iterations

int **setATimes**(void \*A\_data, ATimesFn ATimes)

Sets the function pointer for ATimes (see sundials/sundials\_linearsolver.h).

**Parameters**

- **A\_data** –
- **ATimes** –

**Returns**

int **setPreconditioner**(void \*P\_data, PSetupFn Pset, PSolveFn Psol)

Sets function pointers for PSetup and PSolve routines inside of iterative linear solver objects (see sundials/sundials\_linearsolver.h).

**Parameters**

- **P\_data** –
- **Pset** –
- **Psol** –

**Returns**

int **setScalingVectors**(N\_Vector s, N\_Vector nul)

Sets pointers to left/right scaling vectors for the linear system solve (see sundials/sundials\_linearsolver.h).

**Parameters**

- **s** –
- **nul** –

**Returns**

int **getNumIters**() const

Returns the number of linear iterations performed in the last ‘Solve’ call.

**Returns** Number of iterations

*realtype* **getResNorm**() const

Returns the final residual norm from the last ‘Solve’ call.

**Returns** residual norm

N\_Vector **getResid**() const

Get preconditioned initial residual (see sundials/sundials\_linearsolver.h).

**Returns**

## Class SUNLinSolWrapper

- Defined in file\_include\_amici\_sundials\_linsol\_wrapper.h

## Inheritance Relationships

### Derived Types

- public amici::SUNLinSolBand (*Class SUNLinSolBand*)
- public amici::SUNLinSolDense (*Class SUNLinSolDense*)
- public amici::SUNLinSolKLU (*Class SUNLinSolKLU*)
- public amici::SUNLinSolPCG (*Class SUNLinSolPCG*)
- public amici::SUNLinSolSPBCGS (*Class SUNLinSolSPBCGS*)
- public amici::SUNLinSolSPFGMR (*Class SUNLinSolSPFGMR*)
- public amici::SUNLinSolSPGMR (*Class SUNLinSolSPGMR*)
- public amici::SUNLinSolSPTFQMR (*Class SUNLinSolSPTFQMR*)

### Class Documentation

#### class **SUNLinSolWrapper**

A RAII wrapper for SUNLinearSolver structs.

For details on member functions see documentation in sunlinSol/sundials\_linearsolver.h.

Subclassed by *amici::SUNLinSolBand*, *amici::SUNLinSolDense*, *amici::SUNLinSolKLU*, *amici::SUNLinSolPCG*, *amici::SUNLinSolSPBCGS*, *amici::SUNLinSolSPFGMR*, *amici::SUNLinSolSPGMR*, *amici::SUNLinSolSPTFQMR*

#### Public Functions

**SUNLinSolWrapper**() = default

explicit **SUNLinSolWrapper**(SUNLinearSolver linsol)

Wrap existing SUNLinearSolver.

**Parameters** *linsol* –

virtual ~**SUNLinSolWrapper**()

**SUNLinSolWrapper**(*SUNLinSolWrapper* const &other) = delete

Copy constructor.

**Parameters** *other* –

**SUNLinSolWrapper**(*SUNLinSolWrapper* &&other) noexcept

Move constructor.

**Parameters** *other* –

*SUNLinSolWrapper* &**operator**=(*SUNLinSolWrapper* const &other) = delete

Copy assignment.

**Parameters** *other* –

**Returns**

*SUNLinSolWrapper* &**operator**=(*SUNLinSolWrapper* &&other) noexcept

Move assignment.

**Parameters** *other* –

**Returns**

SUNLinearSolver **get**() const

Returns the wrapped SUNLinSol.

**Returns** SUNLinearSolver

SUNLinearSolver\_Type **getType**() const

Returns an identifier for the linear solver type.

**Returns**

void **setup**(SUNMatrix A) const

Performs any linear solver setup needed, based on an updated system matrix A.

**Parameters** *A* –

void **setup**(*SUNMatrixWrapper* const &A) const

Performs any linear solver setup needed, based on an updated system matrix A.

**Parameters** *A* –

int **Solve**(SUNMatrix A, N\_Vector x, N\_Vector b, *realtype* tol) const

Solves a linear system  $A \cdot x = b$ .

**Parameters**

- *A* –
- *x* – A template for cloning vectors needed within the solver.
- *b* –
- *tol* – Tolerance (weighted 2-norm), iterative solvers only

**Returns** error flag

long int **getLastFlag**() const

Returns the last error flag encountered within the linear solver.

**Returns** error flag

int **space**(long int \*lenrwLS, long int \*leniwLS) const

Returns the integer and real workspace sizes for the linear solver.

**Parameters**

- *lenrwLS* – output argument for size of real workspace
- *leniwLS* – output argument for size of integer workspace

**Returns** workspace size

virtual SUNMatrix **getMatrix**() const

Get the matrix A (matrix solvers only).

**Returns** A



## Protected Functions

int **initialize()**

Performs linear solver initialization (assumes that all solver-specific options have been set).

**Returns** error code

## Protected Attributes

SUNLinearSolver **solver\_** = {nullptr}

Wrapped solver

## Class SUNMatrixWrapper

- Defined in file `_include_amici_sundials_matrix_wrapper.h`

## Class Documentation

class **SUNMatrixWrapper**

A RAI wrapper for SUNMatrix structs.

This can create dense, sparse, or banded matrices using the respective constructor.

## Public Functions

**SUNMatrixWrapper()** = default

**SUNMatrixWrapper**(sunindextype M, sunindextype N, sunindextype NNZ, int sparsetype)

Create sparse matrix. See SUNSparseMatrix in `sunmatrix_sparse.h`.

### Parameters

- **M** – Number of rows
- **N** – Number of columns
- **NNZ** – Number of nonzeros
- **sparsetype** – Sparse type

**SUNMatrixWrapper**(sunindextype M, sunindextype N)

Create dense matrix. See SUNDenseMatrix in `sunmatrix_dense.h`.

### Parameters

- **M** – Number of rows
- **N** – Number of columns

**SUNMatrixWrapper**(sunindextype M, sunindextype ubw, sunindextype lbw)

Create banded matrix. See SUNBandMatrix in `sunmatrix_band.h`.

### Parameters

- **M** – Number of rows and columns

- **ubw** – Upper bandwidth
- **lbw** – Lower bandwidth

**SUNMatrixWrapper**(*SUNMatrixWrapper* const &A, *realtype* droptol, int sparsetype)

Create sparse matrix from dense or banded matrix. See SUNSparseFromDenseMatrix and SUNSparseFromBandMatrix in sunmatrix\_sparse.h.

**Parameters**

- **A** – Wrapper for dense matrix
- **droptol** – tolerance for dropping entries
- **sparsetype** – Sparse type

explicit **SUNMatrixWrapper**(SUNMatrix mat)

Wrap existing SUNMatrix.

**Parameters** **mat** –

**~SUNMatrixWrapper**()

**SUNMatrixWrapper**(*SUNMatrixWrapper* const &other)

Copy constructor.

**Parameters** **other** –

**SUNMatrixWrapper**(*SUNMatrixWrapper* &&other)

Move constructor.

**Parameters** **other** –

*SUNMatrixWrapper* &**operator=**(*SUNMatrixWrapper* const &other)

Copy assignment.

**Parameters** **other** –

**Returns**

*SUNMatrixWrapper* &**operator=**(*SUNMatrixWrapper* &&other)

Move assignment.

**Parameters** **other** –

**Returns**

void **reallocate**(sunindextype nnz)

Reallocate space for sparse matrix according to specified nnz.

**Parameters** **nnz** – new number of nonzero entries

void **realloc**()

Reallocate space for sparse matrix to used space according to last entry in indexptrs.

SUNMatrix **get**() const

Get the wrapped SUNMatrix.

---

**Note:** Even though the returned `matrix_` pointer is const qualified, `matrix_>content` will not be const. This is a shortcoming in the underlying C library, which we cannot address and it is not intended that any of those values are modified externally. If `matrix_>content` is manipulated, `cpp:meth:SUNMatrixWrapper:refresh` needs to be called.

---

**Returns** raw SunMatrix object

inline sunindextype **rows()** const

Get the number of rows.

**Returns** number of rows

inline sunindextype **columns()** const

Get the number of columns.

**Returns** number of columns

sunindextype **num\_nonzeros()** const

Get the number of specified non-zero elements (sparse matrices only)

---

**Note:** value will be 0 before indexptrs are set.

---

**Returns** number of nonzero entries

sunindextype **num\_indexptrs()** const

Get the number of indexptrs that can be specified (sparse matrices only)

**Returns** number of indexptrs

sunindextype **capacity()** const

Get the number of allocated data elements.

**Returns** number of allocated entries

*realtype* **\*data()**

Get raw data of a sparse matrix.

**Returns** pointer to first data entry

*realtype* const **\*data()** const

Get const raw data of a sparse matrix.

**Returns** pointer to first data entry

inline *realtype* **get\_data**(sunindextype idx) const

Get data of a sparse matrix.

**Parameters** **idx** – data index

**Returns** idx-th data entry

inline *realtype* **get\_data**(sunindextype irow, sunindextype icol) const

Get data entry for a dense matrix.

**Parameters**

- **irow** – row
- **icol** – col

**Returns** A(irow,icol)

inline void **set\_data**(sunindextype idx, *realtype* data)

Set data entry for a sparse matrix.

**Parameters**

- **idx** – data index
- **data** – data for idx-th entry

inline void **set\_data**(sunindextype irow, sunindextype icol, *realtype* data)

Set data entry for a dense matrix.

**Parameters**

- **irow** – row
- **icol** – col
- **data** – data for idx-th entry

inline sunindextype **get\_indexval**(sunindextype idx) const

Get the index value of a sparse matrix.

**Parameters** **idx** – data index

**Returns** row (CSC) or column (CSR) for idx-th data entry

inline void **set\_indexval**(sunindextype idx, sunindextype val)

Set the index value of a sparse matrix.

**Parameters**

- **idx** – data index
- **val** – row (CSC) or column (CSR) for idx-th data entry

inline void **set\_indexvals**(const gsl::span<sunindextype const> vals)

Set the index values of a sparse matrix.

**Parameters** **vals** – rows (CSC) or columns (CSR) for data entries

inline sunindextype **get\_indexptr**(sunindextype ptr\_idx) const

Get the index pointer of a sparse matrix.

**Parameters** **ptr\_idx** – pointer index

**Returns** index where the ptr\_idx-th column (CSC) or row (CSR) starts

inline void **set\_indexptr**(sunindextype ptr\_idx, sunindextype ptr)

Set the index pointer of a sparse matrix.

**Parameters**

- **ptr\_idx** – pointer index
- **ptr** – data-index where the ptr\_idx-th column (CSC) or row (CSR) starts

inline void **set\_indexptrs**(const gsl::span<sunindextype const> ptrs)

Set the index pointers of a sparse matrix.

**Parameters** **ptrs** – starting data-indices where the columns (CSC) or rows (CSR) start

int **sparsetype**() const

Get the type of sparse matrix.

**Returns** matrix type

void **scale**(*realtype* a)

multiply with a scalar (in-place)

**Parameters** **a** – scalar value to multiply matrix

void **multiply**(N\_Vector c, *const\_N\_Vector* b, *realtype* alpha = 1.0) const

N\_Vector interface for multiply.

**Parameters**

- **c** – output vector, may already contain values
- **b** – multiplication vector
- **alpha** – scalar coefficient for matrix

inline void **multiply**(*AmiVector* &c, *AmiVector* const &b, *realtype* alpha = 1.0) const

*AmiVector* interface for multiply.

**Parameters**

- **c** – output vector, may already contain values
- **b** – multiplication vector
- **alpha** – scalar coefficient for matrix

void **multiply**(gsl::span<*realtype*> c, gsl::span<*realtype* const> b, const *realtype* alpha = 1.0) const

Perform matrix vector multiplication  $c += \alpha * A * b$ .

**Parameters**

- **c** – output vector, may already contain values
- **b** – multiplication vector
- **alpha** – scalar coefficient

void **multiply**(N\_Vector c, *const\_N\_Vector* b, gsl::span<int const> cols, bool transpose) const

Perform reordered matrix vector multiplication  $c += A[:,cols]*b$ .

**Parameters**

- **c** – output vector, may already contain values
- **b** – multiplication vector
- **cols** – int vector for column reordering
- **transpose** – bool transpose A before multiplication

void **multiply**(gsl::span<*realtype*> c, gsl::span<*realtype* const> b, gsl::span<int const> cols, bool transpose) const

Perform reordered matrix vector multiplication  $c += A[:,cols]*b$ .

**Parameters**

- **c** – output vector, may already contain values
- **b** – multiplication vector
- **cols** – int vector for column reordering

- **transpose** – bool transpose A before multiplication

void **sparse\_multiply**(*SUNMatrixWrapper* &C, *SUNMatrixWrapper* const &B) const

Perform matrix matrix multiplication  $C = A * B$  for sparse A, B, C.

---

**Note:** will overwrite existing data, indexptrs, indexvals for C, but will use preallocated space for these vars

---

#### Parameters

- **C** – output matrix,
- **B** – multiplication matrix

void **sparse\_add**(*SUNMatrixWrapper* const &A, *realtype* alpha, *SUNMatrixWrapper* const &B, *realtype* beta)

Perform sparse matrix matrix addition  $C = \alpha * A + \beta * B$ .

---

**Note:** will overwrite existing data, indexptrs, indexvals for C, but will use preallocated space for these vars

---

#### Parameters

- **A** – addition matrix
- **alpha** – scalar A
- **B** – addition matrix
- **beta** – scalar B

void **sparse\_sum**(std::vector<*SUNMatrixWrapper*> const &mats)

Perform matrix-matrix addition  $A = \text{sum}(\text{mats}(0) \dots \text{mats}(\text{len}(\text{mats})))$

---

**Note:** will overwrite existing data, indexptrs, indexvals for A, but will use preallocated space for these vars

---

**Parameters** **mats** – vector of sparse matrices

sunindextype **scatter**(const sunindextype k, const *realtype* beta, sunindextype \*w, gsl::span<*realtype*> x, const sunindextype mark, *SUNMatrixWrapper* \*C, sunindextype nnz) const

Compute  $x = x + \beta * A(:,k)$ , where x is a dense vector and  $A(:,k)$  is sparse, and update the sparsity pattern for  $C(:,j)$  if applicable.

This function currently has two purposes:

- perform parts of sparse matrix-matrix multiplication  $C(:,j) = A(:,k) * B(k,j)$  enabled by passing  $\beta = B(k,j)$ ,  $x = C(:,j)$ ,  $C = C$ ,  $w = \text{sparsity of } C(:,j) \text{ from } B(k, 0 \dots j-1)$ ,  $\text{nnz} = \text{nnz}(C(:, 0 \dots j-1))$
- add the k-th column of the sparse matrix A multiplied by beta to the dense vector x. enabled by passing  $\beta = *$ ,  $x = x$ ,  $C = \text{nullptr}$ ,  $w = \text{nullptr}$ ,  $\text{nnz} = *$

#### Parameters

- **k** – column index

- **beta** – scaling factor
- **w** – index workspace, ( $w[i] < \text{mark}$ ) indicates non-zeros of  $C(i,j)$  (dimension:  $m$ ), if this is a nullptr, sparsity pattern of  $C$  will not be updated (if applicable).
- **x** – dense output vector (dimension:  $m$ )
- **mark** – marker for  $w$  to indicate nonzero pattern
- **C** – sparse output matrix, if this is a nullptr, sparsity pattern of  $C$  will not be updated
- **nnz** – number of nonzeros that were already written to  $C$

**Returns** updated number of nonzeros in  $C$

void **transpose**(*SUNMatrixWrapper* & $C$ , const *realtype*  $\alpha$ , sunindextype  $\text{blocksize}$ ) const

Compute transpose  $A'$  of sparse matrix  $A$  and writes it to the matrix  $C = \alpha * A'$ .

**Parameters**

- **C** – output matrix (sparse or dense)
- **alpha** – scalar multiplier
- **blocksize** – blocksize for transposition. For full matrix transpose set to  $\text{ncols}/\text{nrows}$

void **to\_dense**(*SUNMatrixWrapper* & $D$ ) const

Writes a sparse matrix  $A$  to a dense matrix  $D$ .

**Parameters** **D** – dense output matrix

void **to\_diag**(N\_Vector  $v$ ) const

Writes the diagonal of sparse matrix  $A$  to a dense vector  $v$ .

**Parameters** **v** – dense output vector

void **zero**()

Set to 0.0, for sparse matrices also resets  $\text{indexptr}/\text{indexvals}$ .

inline SUNMatrix\_ID **matrix\_id**() const

Get matrix id.

**Returns** SUNMatrix\_ID

void **refresh**()

Update internal cache, needs to be called after external manipulation of  $\text{matrix\_} \rightarrow \text{content}$ .

### Class **SUNNonLinSolFixedPoint**

- Defined in `file_include_amici_sundials_linsol_wrapper.h`

## Inheritance Relationships

### Base Type

- `public amici::SUNNonLinSolWrapper` (*Class SUNNonLinSolWrapper*)

### Class Documentation

class **SUNNonLinSolFixedPoint** : public amici::SUNNonLinSolWrapper  
 SUNDIALS Fixed point non-linear solver to solve  $G(y) = y$ .

#### Public Functions

explicit **SUNNonLinSolFixedPoint**(*const\_N\_Vector* x, int m = 0)  
 Create fixed-point solver.

##### Parameters

- **x** – template for cloning vectors needed within the solver.
- **m** – number of acceleration vectors to use

**SUNNonLinSolFixedPoint**(int count, *const\_N\_Vector* x, int m = 0)  
 Create fixed-point solver for use with sensitivity analysis.

##### Parameters

- **count** – Number of vectors in the nonlinear solve. When integrating a system containing  $N_s$  sensitivities the value of count is:
  - $N_s + 1$  if using a simultaneous corrector approach.
  - $N_s$  if using a staggered corrector approach.
- **x** – template for cloning vectors needed within the solver.
- **m** – number of acceleration vectors to use

int **getSysFn**(SUNNonlinSolSysFn \*SysFn) const  
 Get function to evaluate the fixed point function  $G(y) = y$ .

##### Parameters SysFn –

##### Returns

### Class SUNNonLinSolNewton

- Defined in file\_include\_amici\_sundials\_linsol\_wrapper.h



## Inheritance Relationships

### Base Type

- `public amici::SUNNonLinSolWrapper` (*Class `SUNNonLinSolWrapper`*)

### Class Documentation

class **SUNNonLinSolNewton** : public amici::*SUNNonLinSolWrapper*

SUNDIALS Newton non-linear solver to solve  $F(y) = 0$ .

#### Public Functions

explicit **SUNNonLinSolNewton**(N\_Vector x)

Create Newton solver.

**Parameters** **x** – A template for cloning vectors needed within the solver.

**SUNNonLinSolNewton**(int count, N\_Vector x)

Create Newton solver for enabled sensitivity analysis.

**Parameters**

- **count** – Number of vectors in the nonlinear solve. When integrating a system containing  $N_s$  sensitivities the value of count is:
  - $N_s+1$  if using a simultaneous corrector approach.
  - $N_s$  if using a staggered corrector approach.
- **x** – A template for cloning vectors needed within the solver.

int **getSysFn**(SUNNonlinSolSysFn \*SysFn) const

Get function to evaluate the nonlinear residual function  $F(y) = 0$ .

**Parameters** **SysFn** –

**Returns**

### Class `SUNNonLinSolWrapper`

- Defined in file `include/amici_sundials_linsol_wrapper.h`

## Inheritance Relationships

### Derived Types

- `public amici::SUNNonLinSolFixedPoint` (*Class `SUNNonLinSolFixedPoint`*)
- `public amici::SUNNonLinSolNewton` (*Class `SUNNonLinSolNewton`*)

## Class Documentation

### class **SUNNonLinSolWrapper**

A RAI wrapper for SUNNonLinearSolver structs which solve the nonlinear system  $F(y) = 0$  or  $G(y) = y$ .

Subclassed by *amici::SUNNonLinSolFixedPoint*, *amici::SUNNonLinSolNewton*

### Public Functions

explicit **SUNNonLinSolWrapper**(SUNNonlinearSolver sol)

*SUNNonLinSolWrapper* from existing SUNNonlinearSolver.

**Parameters** *sol* –

virtual **~SUNNonLinSolWrapper**()

**SUNNonLinSolWrapper**(*SUNNonLinSolWrapper* const &other) = delete

Copy constructor.

**Parameters** *other* –

**SUNNonLinSolWrapper**(*SUNNonLinSolWrapper* &&other) noexcept

Move constructor.

**Parameters** *other* –

*SUNNonLinSolWrapper* &**operator=**(*SUNNonLinSolWrapper* const &other) = delete

Copy assignment.

**Parameters** *other* –

**Returns**

*SUNNonLinSolWrapper* &**operator=**(*SUNNonLinSolWrapper* &&other) noexcept

Move assignment.

**Parameters** *other* –

**Returns**

SUNNonlinearSolver **get**() const

Get the wrapped SUNNonlinearSolver.

**Returns** SUNNonlinearSolver

SUNNonlinearSolver\_Type **getType**() const

Get type ID of the solver.

**Returns**

int **setup**(N\_Vector y, void \*mem)

Setup solver.

**Parameters**

- *y* – the initial iteration passed to the nonlinear solver.
- *mem* – the sundials integrator memory structure.

**Returns**

int **Solve**(N\_Vector y0, N\_Vector y, N\_Vector w, *realtype* tol, bool callLSetup, void \*mem)

Solve the nonlinear system  $F(y) = 0$  or  $G(y) = y$ .

#### Parameters

- **y0** – the initial iterate for the nonlinear solve. This must remain unchanged throughout the solution process.
- **y** – the solution to the nonlinear system
- **w** – the solution error weight vector used for computing weighted error norms.
- **tol** – the requested solution tolerance in the weighted root-mean- squared norm.
- **callLSetup** – a flag indicating that the integrator recommends for the linear solver setup function to be called.
- **mem** – the sundials integrator memory structure.

#### Returns

int **setSysFn**(SUNNonlinSolSysFn SysFn)

Set function to evaluate the nonlinear residual function  $F(y) = 0$  or the fixed point function  $G(y) = y$ .

#### Parameters SysFn –

#### Returns

int **setLSetupFn**(SUNNonlinSolLSetupFn SetupFn)

Set linear solver setup function.

#### Parameters SetupFn –

#### Returns

int **setLSolveFn**(SUNNonlinSolLSolveFn SolveFn)

Set linear solver solve function.

#### Parameters SolveFn –

#### Returns

int **setConvTestFn**(SUNNonlinSolConvTestFn CTestFn, void \*ctest\_data)

Set function to test for convergence.

#### Parameters

- **CTestFn** –
- **ctest\_data** –

#### Returns

int **setMaxIters**(int maxiters)

Set maximum number of non-linear iterations.

#### Parameters maxiters –

#### Returns

long int **getNumIters**() const

getNumIters

#### Returns

int **getCurIter**() const  
 getCurIter

#### Returns

long int **getNumConvFails**() const  
 getNumConvFails

#### Returns

### Protected Functions

void **initialize**()  
 initialize

### Protected Attributes

SUNNonlinearSolver **solver** = nullptr  
 the wrapper solver

## Enums

### Enum BLASLayout

- Defined in file\_include\_amici\_defines.h

### Enum Documentation

enum class amici::BLASLayout  
 BLAS Matrix Layout, affects dgemm and gemv calls  
*Values:*

enumerator **rowMajor**

enumerator **colMajor**

### Enum BLASTranspose

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::**BLASTranspose**

BLAS Matrix Transposition, affects dgemm and gemv calls

*Values:*

enumerator **noTrans**

enumerator **trans**

enumerator **conjTrans**

## Enum FixedParameterContext

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::**FixedParameterContext**

fixedParameter to be used in condition context

*Values:*

enumerator **simulation**

enumerator **preequilibration**

enumerator **presimulation**

## Enum InternalSensitivityMethod

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::**InternalSensitivityMethod**

CVODES/IDAS forward sensitivity computation method

*Values:*

enumerator **simultaneous**

enumerator **staggered**

enumerator **staggered1**

## Enum InterpolationType

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::InterpolationType

CVODES/IDAS state interpolation for adjoint sensitivity analysis

*Values:*

enumerator **hermite**

enumerator **polynomial**

## Enum LinearMultistepMethod

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::LinearMultistepMethod

CVODES/IDAS linear multistep method

*Values:*

enumerator **adams**

enumerator **BDF**

## Enum LinearSolver

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::LinearSolver  
linear solvers for CVODES/IDAS  
*Values:*  
  
enumerator **dense**  
  
enumerator **band**  
  
enumerator **LAPACKDense**  
  
enumerator **LAPACKBand**  
  
enumerator **diag**  
  
enumerator **SPGMR**  
  
enumerator **SPBCG**  
  
enumerator **SPTFQMR**  
  
enumerator **KLU**  
  
enumerator **SuperLUMT**

## Enum LogSeverity

- Defined in file\_include\_amici\_logging.h

## Enum Documentation

enum class amici::LogSeverity  
Severity levels for logging.  
*Values:*  
  
enumerator **error**  
  
enumerator **warning**  
  
enumerator **debug**

## Enum ModelQuantity

- Defined in file\_include\_amici\_model.h

## Enum Documentation

enum class amici::ModelQuantity

Describes the various model quantities.

*Values:*

enumerator **J**

enumerator **JB**

enumerator **Jv**

enumerator **JvB**

enumerator **JDiag**

enumerator **sx**

enumerator **sy**

enumerator **sz**

enumerator **srz**

enumerator **ssigmay**

enumerator **ssigmaz**

enumerator **xdot**

enumerator **sxdot**

enumerator **xBdot**

enumerator **x0\_rdata**

enumerator **x0**



enumerator **x\_rdata**

enumerator **x**

enumerator **dwdw**

enumerator **dwdx**

enumerator **dwdp**

enumerator **y**

enumerator **dydp**

enumerator **dydx**

enumerator **w**

enumerator **root**

enumerator **qBdot**

enumerator **qBdot\_ss**

enumerator **xBdot\_ss**

enumerator **JSparseB\_ss**

enumerator **deltax**

enumerator **deltasx**

enumerator **deltaxB**

enumerator **k**

enumerator **p**

enumerator **ts**

enumerator **dJydy**

enumerator **dJydy\_matlab**

enumerator **deltaqB**

enumerator **dsigmaydp**

enumerator **dsigmaydy**

enumerator **dsigmazdp**

enumerator **dJydsigma**

enumerator **dJydx**

enumerator **dzdx**

enumerator **dzdp**

enumerator **dJrzdsigma**

enumerator **dJrzdz**

enumerator **dJrzdx**

enumerator **dJzdsigma**

enumerator **dJzdz**

enumerator **dJzdx**

enumerator **drzdp**

enumerator **drzdx**

## Enum **NewtonDampingFactorMode**

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::**NewtonDampingFactorMode**

Damping factor flag for the Newton method

*Values:*

enumerator **off**

enumerator **on**

## Enum NonlinearSolverIteration

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::**NonlinearSolverIteration**

CVODES/IDAS Nonlinear Iteration method

*Values:*

enumerator **functional**

enumerator **fixedpoint**  
deprecated

enumerator **newton**

## Enum ObservableScaling

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::**ObservableScaling**

modes for observable scaling

*Values:*

enumerator **lin**

enumerator **log**

enumerator **log10**

## Enum ParameterScaling

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::ParameterScaling

modes for parameter transformations

*Values:*

enumerator **none**

enumerator **ln**

enumerator **log10**

## Enum RDataReporting

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::RDataReporting

*Values:*

enumerator **full**

enumerator **residuals**

enumerator **likelihood**

## Enum SecondOrderMode

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::SecondOrderMode  
modes for second order sensitivity analysis  
*Values:*  
  
enumerator **none**  
  
enumerator **full**  
  
enumerator **directional**

## Enum SensitivityMethod

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::SensitivityMethod  
methods for sensitivity computation  
*Values:*  
  
enumerator **none**  
Don't compute sensitivities.  
  
enumerator **forward**  
Forward sensitivity analysis.  
  
enumerator **adjoint**  
Adjoint sensitivity analysis.

## Enum SensitivityOrder

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::SensitivityOrder

orders of sensitivity analysis

*Values:*

enumerator **none**

Don't compute sensitivities.

enumerator **first**

First-order sensitivities.

enumerator **second**

Second-order sensitivities.

## Enum SplineBoundaryCondition

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::SplineBoundaryCondition

boundary conditions for splines

*Values:*

enumerator **given**

enumerator **zeroDerivative**

enumerator **natural**

enumerator **naturalZeroDerivative**

enumerator **periodic**

## Enum SplineExtrapolation

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::SplineExtrapolation

extrapolation methods for splines

*Values:*

enumerator **noExtrapolation**

enumerator **constant**

enumerator **linear**

enumerator **polynomial**

enumerator **periodic**

## Enum SteadyStateComputationMode

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::SteadyStateComputationMode

Steady-state computation mode in steadyStateProblem

*Values:*

enumerator **newtonOnly**

enumerator **integrationOnly**

enumerator **integrateIfNewtonFails**

## Enum SteadyStateContext

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::SteadyStateContext

Context for which the sensitivity flag should be computed

*Values:*

enumerator **newtonSensi**

enumerator **sensiStorage**

enumerator **solverCreation**

## Enum SteadyStateSensitivityMode

- Defined in file\_include\_amici\_defines.h

## Enum Documentation

enum class amici::SteadyStateSensitivityMode

Sensitivity computation mode in steadyStateProblem

*Values:*

enumerator **newtonOnly**

enumerator **integrationOnly**

enumerator **integrateIfNewtonFails**

## Enum SteadyStateStatus

- Defined in file\_include\_amici\_defines.h



## Enum Documentation

enum class amici::SteadyStateStatus

State in which the steady state computation finished

*Values:*

enumerator **failed\_too\_long\_simulation**

enumerator **failed\_damping**

enumerator **failed\_factorization**

enumerator **failed\_convergence**

enumerator **failed**

enumerator **not\_run**

enumerator **success**

## Functions

**Template Function** amici::addSlice(const gsl::span<T const>, gsl::span<T>)

- Defined in file\_include\_amici\_misc.h

## Function Documentation

template<class T>

void amici::addSlice(const gsl::span<T const> slice, gsl::span<T> buffer)

local helper function to add the computed slice to provided buffer (span)

### Parameters

- **slice** – computed value
- **buffer** – buffer to which values are to be added

### Template Function amici::addSlice(std::vector<T> const&, gsl::span<T>)

- Defined in file\_include\_amici\_misc.h

#### Function Documentation

template<class T>  
void amici::addSlice(std::vector<T> const &s, gsl::span<T> b)  
local helper function to add the computed slice to provided buffer (vector/span)

##### Parameters

- **s** – computed value
- **b** – buffer to which values are to be written

### Function amici::amici\_daxpy

- Defined in file\_include\_amici\_cblas.h

#### Function Documentation

void amici::amici\_daxpy(int n, double alpha, double const \*x, int incx, double \*y, int incy)  
Compute  $y = \alpha x + y$ .

##### Parameters

- **n** – number of elements in y
- **alpha** – scalar coefficient of x
- **x** – vector of length n\*incx
- **incx** – x stride
- **y** – vector of length n\*incy
- **incy** – y stride

### Function amici::amici\_dgemm

- Defined in file\_include\_amici\_cblas.h

#### Function Documentation

void amici::amici\_dgemm(*BLASLayout* layout, *BLASTranspose* TransA, *BLASTranspose* TransB, int M, int N, int K, double alpha, double const \*A, int lda, double const \*B, int ldb, double beta, double \*C, int ldc)

CBLAS matrix matrix multiplication (dgemm)

This routines computes  $C = \alpha A * B + \beta C$  with A: [MxK] B:[KxN] C:[MxN]

##### Parameters

- **layout** – memory layout.
- **TransA** – flag indicating whether A should be transposed before multiplication
- **TransB** – flag indicating whether B should be transposed before multiplication
- **M** – number of rows in A/C
- **N** – number of columns in B/C
- **K** – number of rows in B, number of columns in A
- **alpha** – coefficient alpha
- **A** – matrix A
- **lda** – leading dimension of A ( $\geq M$  or  $\geq K$ )
- **B** – matrix B
- **ldb** – leading dimension of B ( $\geq K$  or  $\geq N$ )
- **beta** – coefficient beta
- **C** – matrix C
- **ldc** – leading dimension of C ( $\geq M$  or  $\geq N$ )

### Function amici::amici\_dgemv

- Defined in file\_include\_amici\_cblas.h

### Function Documentation

void amici::amici\_dgemv(*BLASLayout* layout, *BLASTranspose* TransA, int M, int N, double alpha, double const \*A, int lda, double const \*X, int incX, double beta, double \*Y, int incY)

CBLAS matrix vector multiplication (dgemv).

Computes  $y = \alpha * A * x + \beta * y$  with A: [MxN] x:[Nx1] y:[Mx1]

#### Parameters

- **layout** – Matrix layout, column major or row major.
- **TransA** – flag indicating whether A should be transposed before multiplication
- **M** – number of rows in A
- **N** – number of columns in A
- **alpha** – coefficient alpha
- **A** – matrix A
- **lda** – leading dimension / stride of A ( $\geq N$  if row-major,  $\geq M$  if col-major)
- **X** – vector X
- **incX** – increment for entries of X
- **beta** – coefficient beta
- **Y** – vector Y
- **incY** – increment for entries of Y

## Function amici::backtraceString

- Defined in file\_include\_amici\_misc.h

### Function Documentation

std::string amici::backtraceString(int maxFrames, int const first\_frame = 0)

Returns the current backtrace as std::string.

#### Parameters

- **maxFrames** – Number of frames to include
- **first\_frame** – Index of first frame to include

**Returns** Backtrace

## Template Function amici::checkBufferSize

- Defined in file\_include\_amici\_misc.h

### Function Documentation

template<class T>

void amici::checkBufferSize(gsl::span<T> buffer, typename gsl::span<T>::index\_type expected\_size)

local helper to check whether the provided buffer has the expected size

#### Parameters

- **buffer** – buffer to which values are to be written
- **expected\_size** – expected size of the buffer

## Function amici::checkSigmaPositivity(std::vector<realtype> const&, char const \*)

- Defined in file\_include\_amici\_edata.h

### Function Documentation

void amici::checkSigmaPositivity(std::vector<realtype> const &sigmaVector, char const \*vectorName)

checks input vector of sigmas for not strictly positive values

#### Parameters

- **sigmaVector** – vector input to be checked
- **vectorName** – name of the input

**Function amici::checkSigmaPositivity(realtype, char const \*)**

- Defined in file\_include\_amici\_edata.h

**Function Documentation**

void amici::checkSigmaPositivity(*realtype* sigma, char const \*sigmaName)

checks input scalar sigma for not strictly positive value

**Parameters**

- **sigma** – input to be checked
- **sigmaName** – name of the input

**Template Function amici::deserializeFromChar**

- Defined in file\_include\_amici\_serialization.h

**Function Documentation**

template<typename T>

*T* amici::deserializeFromChar(char const \*buffer, int size)

Deserialize object that has been serialized using serializeToChar.

**Parameters**

- **buffer** – serialized object
- **size** – length of buffer

**Returns** The deserialized object

**Template Function amici::deserializeFromString**

- Defined in file\_include\_amici\_serialization.h

**Function Documentation**

template<typename T>

*T* amici::deserializeFromString(std::string const &serialized)

Deserialize object that has been serialized using serializeToString.

**Parameters** **serialized** – serialized object

**Returns** The deserialized object

## Function amici::dotProd

- Defined in file\_include\_amici\_vector.h

### Function Documentation

inline *realtype* amici::dotProd(*AmiVector* const &x, *AmiVector* const &y)

Compute dot product of x and y.

#### Parameters

- **x** – vector
- **y** – vector

**Returns** dot product of x and y

## Function amici::getScaledParameter

- Defined in file\_include\_amici\_misc.h

### Function Documentation

double amici::getScaledParameter(double unscaledParameter, *ParameterScaling* scaling)

Apply parameter scaling according to scaling

#### Parameters

- **unscaledParameter** –
- **scaling** – parameter scaling

**Returns** Scaled parameter

## Function amici::getUnscaledParameter

- Defined in file\_include\_amici\_misc.h

### Function Documentation

double amici::getUnscaledParameter(double scaledParameter, *ParameterScaling* scaling)

Remove parameter scaling according to scaling

#### Parameters

- **scaledParameter** – scaled parameter
- **scaling** – parameter scaling

**Returns** Unscaled parameter

**Function amici::hdf5::attributeExists(H5::H5File const&, std::string const&, std::string const&)**

- Defined in file\_include\_amici\_hdf5.h

**Function Documentation**

```
bool amici::hdf5::attributeExists(H5::H5File const &file, std::string const &optionsObject, std::string const &attributeName)
```

Check whether an attribute with the given name exists on the given dataset.

**Parameters**

- **file** – The HDF5 file object
- **optionsObject** – Dataset of which attributes should be checked
- **attributeName** – Name of the attribute of interest

**Returns** true if attribute exists, false otherwise

**Function amici::hdf5::attributeExists(H5::H5Object const&, std::string const&)**

- Defined in file\_include\_amici\_hdf5.h

**Function Documentation**

```
bool amici::hdf5::attributeExists(H5::H5Object const &object, std::string const &attributeName)
```

Check whether an attribute with the given name exists on the given object.

**Parameters**

- **object** – An HDF5 object
- **attributeName** – Name of the attribute of interest

**Returns** true if attribute exists, false otherwise

**Function amici::hdf5::createAndWriteDouble1DDataset**

- Defined in file\_include\_amici\_hdf5.h

**Function Documentation**

```
void amici::hdf5::createAndWriteDouble1DDataset(H5::H5File const &file, std::string const &datasetName, gsl::span<double const> buffer)
```

Create and write to 1-dimensional native double dataset.

**Parameters**

- **file** – HDF5 file object
- **datasetName** – Name of dataset to create
- **buffer** – Data to write to dataset

## Function `amici::hdf5::createAndWriteDouble2DDataset`

- Defined in `file_include_amici_hdf5.h`

### Function Documentation

```
void amici::hdf5::createAndWriteDouble2DDataset(H5::H5File const &file, std::string const &datasetName,  
                                                gsl::span<double const> buffer, hsize_t m, hsize_t n)
```

Create and write to 2-dimensional native double dataset.

#### Parameters

- **file** – HDF5 file object
- **datasetName** – Name of dataset to create
- **buffer** – Flattened data to write to dataset (assuming row-major)
- **m** – Number of rows in buffer
- **n** – Number of columns buffer

## Function `amici::hdf5::createAndWriteDouble3DDataset`

- Defined in `file_include_amici_hdf5.h`

### Function Documentation

```
void amici::hdf5::createAndWriteDouble3DDataset(H5::H5File const &file, std::string const &datasetName,  
                                                gsl::span<double const> buffer, hsize_t m, hsize_t n,  
                                                hsize_t o)
```

Create and write to 3-dimensional native double dataset.

#### Parameters

- **file** – HDF5 file object
- **datasetName** – Name of dataset to create
- **buffer** – Flattened data to write to dataset (assuming row-major)
- **m** – Length of first dimension in buffer
- **n** – Length of first dimension in buffer
- **o** – Length of first dimension in buffer



### Function amici::hdf5::createAndWriteInt1DDataset

- Defined in file\_include\_amici\_hdf5.h

#### Function Documentation

```
void amici::hdf5::createAndWriteInt1DDataset(H5::H5File const &file, std::string const &datasetName,  
                                              gsl::span<int const> buffer)
```

Create and write to 1-dimensional native integer dataset.

##### Parameters

- **file** – HDF5 file object
- **datasetName** – Name of dataset to create
- **buffer** – Data to write to dataset

### Function amici::hdf5::createAndWriteInt2DDataset

- Defined in file\_include\_amici\_hdf5.h

#### Function Documentation

```
void amici::hdf5::createAndWriteInt2DDataset(H5::H5File const &file, std::string const &datasetName,  
                                              gsl::span<int const> buffer, hsize_t m, hsize_t n)
```

Create and write to 2-dimensional native integer dataset.

##### Parameters

- **file** – HDF5 file object
- **datasetName** – Name of dataset to create
- **buffer** – Flattened data to write to dataset (assuming row-major)
- **m** – Number of rows in buffer
- **n** – Number of columns buffer

### Function amici::hdf5::createGroup

- Defined in file\_include\_amici\_hdf5.h

## Function Documentation

`void amici::hdf5::createGroup(const H5::H5File &file, std::string const &groupPath, bool recursively = true)`

Create the given group and possibly parents.

### Parameters

- **file** – HDF5 file to write to
- **groupPath** – Path to the group to be created
- **recursively** – Create intermediary groups

## Function `amici::hdf5::createOrOpenForWriting`

- Defined in `file_include_amici_hdf5.h`

## Function Documentation

`H5::H5File amici::hdf5::createOrOpenForWriting(std::string const &hdf5filename)`

Open the given file for writing.

Append if exists, create if not.

**Parameters** **hdf5filename** – File to open

**Returns** File object

## Function `amici::hdf5::getDoubleDataset1D`

- Defined in `file_include_amici_hdf5.h`

## Function Documentation

`std::vector<double> amici::hdf5::getDoubleDataset1D(const H5::H5File &file, std::string const &name)`

Read 1-dimensional native double dataset from HDF5 file.

### Parameters

- **file** – HDF5 file object
- **name** – Name of dataset to read

**Returns** Data read

### Function `amici::hdf5::getDoubleDataset2D`

- Defined in `file_include_amici_hdf5.h`

#### Function Documentation

```
std::vector<double> amici::hdf5::getDoubleDataset2D(const H5::H5File &file, std::string const &name,
                                                    hsize_t &m, hsize_t &n)
```

Read 2-dimensional native double dataset from HDF5 file.

##### Parameters

- **file** – HDF5 file object
- **name** – Name of dataset to read
- **m** – Number of rows in the dataset
- **n** – Number of columns in the dataset

**Returns** Flattened data (row-major)

### Function `amici::hdf5::getDoubleDataset3D`

- Defined in `file_include_amici_hdf5.h`

#### Function Documentation

```
std::vector<double> amici::hdf5::getDoubleDataset3D(const H5::H5File &file, std::string const &name,
                                                    hsize_t &m, hsize_t &n, hsize_t &o)
```

Read 3-dimensional native double dataset from HDF5 file.

##### Parameters

- **file** – HDF5 file object
- **name** – Name of dataset to read
- **m** – Length of first dimension in dataset
- **n** – Length of first dimension in dataset
- **o** – Length of first dimension in dataset

**Returns** Flattened data (row-major)

### Function `amici::hdf5::getDoubleScalarAttribute`

- Defined in `file_include_amici_hdf5.h`

## Function Documentation

`double amici::hdf5::getDoubleScalarAttribute(const H5::H5File &file, std::string const &optionsObject, std::string const &attributeName)`

Read scalar native double attribute from HDF5 object.

### Parameters

- **file** – HDF5 file
- **optionsObject** – Object to read attribute from
- **attributeName** – Name of attribute to read

**Returns** Attribute value

## Function `amici::hdf5::getIntDataset1D`

- Defined in `file_include_amici_hdf5.h`

## Function Documentation

`std::vector<int> amici::hdf5::getIntDataset1D(const H5::H5File &file, std::string const &name)`

Read 1-dimensional native integer dataset from HDF5 file.

### Parameters

- **file** – HDF5 file object
- **name** – Name of dataset to read

**Returns** Data read

## Function `amici::hdf5::getIntScalarAttribute`

- Defined in `file_include_amici_hdf5.h`

## Function Documentation

`int amici::hdf5::getIntScalarAttribute(const H5::H5File &file, std::string const &optionsObject, std::string const &attributeName)`

Read scalar native integer attribute from HDF5 object.

### Parameters

- **file** – HDF5 file
- **optionsObject** – Object to read attribute from
- **attributeName** – Name of attribute to read

**Returns** Attribute value

## Function amici::hdf5::getStringAttribute

- Defined in file\_include\_amici\_hdf5.h

### Function Documentation

std::string amici::hdf5::getStringAttribute(H5::H5File const &file, std::string const &optionsObject, std::string const &attributeName)

Read string attribute from HDF5 object.

#### Parameters

- **file** – HDF5 file
- **optionsObject** – Object to read attribute from
- **attributeName** – Name of attribute to read

**Returns** Attribute value

## Function amici::hdf5::locationExists(std::string const&, std::string const&)

- Defined in file\_include\_amici\_hdf5.h

### Function Documentation

bool amici::hdf5::locationExists(std::string const &filename, std::string const &location)

Check if the given location (group, link or dataset) exists in the given file.

#### Parameters

- **filename** – HDF5 filename
- **location** – Location to test for

**Returns** true if exists, false otherwise

## Function amici::hdf5::locationExists(H5::H5File const&, std::string const&)

- Defined in file\_include\_amici\_hdf5.h

### Function Documentation

bool amici::hdf5::locationExists(H5::H5File const &file, std::string const &location)

Check if the given location (group, link or dataset) exists in the given file.

#### Parameters

- **file** – HDF5 file object
- **location** – Location to test for

**Returns** true if exists, false otherwise

## Function `amici::hdf5::readModelDataFromHDF5(std::string const&, Model&, std::string const&)`

- Defined in `file_include_amici_hdf5.h`

### Function Documentation

`void amici::hdf5::readModelDataFromHDF5`(`std::string const &hdf5file`, *Model* &model, `std::string const &datasetPath`)

Read model data from HDF5 file.

#### Parameters

- **hdf5file** – Name of HDF5 file
- **model** – *Model* to set data on
- **datasetPath** – Path inside the HDF5 file

## Function `amici::hdf5::readModelDataFromHDF5(H5::H5File const&, Model&, std::string const&)`

- Defined in `file_include_amici_hdf5.h`

### Function Documentation

`void amici::hdf5::readModelDataFromHDF5`(`H5::H5File const &file`, *Model* &model, `std::string const &datasetPath`)

Read model data from HDF5 file.

#### Parameters

- **file** – HDF5 file handle to read from
- **model** – *Model* to set data on
- **datasetPath** – Path inside the HDF5 file

## Function `amici::hdf5::readSimulationExpData`

- Defined in `file_include_amici_hdf5.h`

### Function Documentation

`std::unique_ptr<ExpData> amici::hdf5::readSimulationExpData`(`std::string const &hdf5Filename`, `std::string const &hdf5Root`, *Model* const &model)

Read AMICI *ExpData* data from HDF5 file.

#### Parameters

- **hdf5Filename** – Name of HDF5 file
- **hdf5Root** – Path inside the HDF5 file to object having *ExpData*
- **model** – The model for which data is to be read

**Returns** *ExpData* created from data in the given location

**Function** `amici::hdf5::readSolverSettingsFromHDF5(const H5::H5File&, Solver&, std::string const&)`

- Defined in `file_include_amici_hdf5.h`

## Function Documentation

`void amici::hdf5::readSolverSettingsFromHDF5(const H5::H5File &file, Solver &solver, std::string const &datasetPath)`

Read solver options from HDF5 file.

### Parameters

- **file** – HDF5 file to read from
- **solver** – *Solver* to set options on
- **datasetPath** – Path inside the HDF5 file

**Function** `amici::hdf5::readSolverSettingsFromHDF5(std::string const&, Solver&, std::string const&)`

- Defined in `file_include_amici_hdf5.h`

## Function Documentation

`void amici::hdf5::readSolverSettingsFromHDF5(std::string const &hdffile, Solver &solver, std::string const &datasetPath)`

Read solver options from HDF5 file.

### Parameters

- **hdffile** – Name of HDF5 file
- **solver** – *Solver* to set options on
- **datasetPath** – Path inside the HDF5 file

**Function** `amici::hdf5::writeReturnData(ReturnData const&, H5::H5File const&, std::string const&)`

- Defined in `file_include_amici_hdf5.h`

## Function Documentation

void amici::hdf5::writeReturnData(*ReturnData* const &rdata, H5::H5File const &file, std::string const &hdf5Location)

Write *ReturnData* to HDF5 file.

### Parameters

- **rdata** – Data to write
- **file** – HDF5 file to write to
- **hdf5Location** – Full dataset path inside the HDF5 file (will be created)

## Function amici::hdf5::writeReturnData(ReturnData const&, std::string const&, std::string const&)

- Defined in file\_include\_amici\_hdf5.h

## Function Documentation

void amici::hdf5::writeReturnData(*ReturnData* const &rdata, std::string const &hdf5Filename, std::string const &hdf5Location)

Write *ReturnData* to HDF5 file.

### Parameters

- **rdata** – Data to write
- **hdf5Filename** – Filename of HDF5 file
- **hdf5Location** – Full dataset path inside the HDF5 file (will be created)

## Function amici::hdf5::writeReturnDataDiagnosis

- Defined in file\_include\_amici\_hdf5.h

## Function Documentation

void amici::hdf5::writeReturnDataDiagnosis(*ReturnData* const &rdata, H5::H5File const &file, std::string const &hdf5Location)

Write *ReturnData* diagnosis data to HDF5 file.

### Parameters

- **rdata** – Data to write
- **file** – HDF5 file to write to
- **hdf5Location** – Full dataset path inside the HDF5 file (will be created)



## Function amici::hdf5::writeSimulationExpData

- Defined in file\_include\_amici\_hdf5.h

### Function Documentation

void amici::hdf5::writeSimulationExpData(*ExpData* const &edata, H5::H5File const &file, std::string const &hdf5Location)

Write AMICI experimental data to HDF5 file.

#### Parameters

- **edata** – The experimental data which is to be written
- **file** – Name of HDF5 file
- **hdf5Location** – Path inside the HDF5 file to object having *ExpData*

## Function amici::hdf5::writeSolverSettingsToHDF5(Solver const&, std::string const&, std::string const&)

- Defined in file\_include\_amici\_hdf5.h

### Function Documentation

void amici::hdf5::writeSolverSettingsToHDF5(*Solver* const &solver, std::string const &hdf5Filename, std::string const &hdf5Location)

Write solver options to HDF5 file.

#### Parameters

- **hdf5Filename** – Name of HDF5 file to write to
- **solver** – *Solver* to write options from
- **hdf5Location** – Path inside the HDF5 file

## Function amici::hdf5::writeSolverSettingsToHDF5(Solver const&, H5::H5File const&, std::string const&)

- Defined in file\_include\_amici\_hdf5.h

### Function Documentation

void amici::hdf5::writeSolverSettingsToHDF5(*Solver* const &solver, H5::H5File const &file, std::string const &hdf5Location)

Write solver options to HDF5 file.

#### Parameters

- **file** – File to read from
- **solver** – *Solver* to write options from

- **hdf5Location** – Path inside the HDF5 file

### Template Function amici::is\_equal

- Defined in file\_include\_amici\_misc.h

### Function Documentation

```
template<class T>
bool amici::is_equal(T const &a, T const &b)
```

Check if two spans are equal, treating NaNs in the same position as equal.

#### Parameters

- **a** –
- **b** –

**Returns** Whether the contents of the two spans are equal.

### Function amici::linearSum

- Defined in file\_include\_amici\_vector.h

### Function Documentation

```
inline void amici::linearSum(realtype a, AmiVector const &x, realtype b, AmiVector const &y, AmiVector &z)
```

Computes  $z = a*x + b*y$ .

#### Parameters

- **a** – coefficient for x
- **x** – a vector
- **b** – coefficient for y
- **y** – another vector with same size as x
- **z** – result vector of same size as x and y

### Function amici::N\_VGetArrayPointerConst

- Defined in file\_include\_amici\_vector.h

## Function Documentation

inline *realttype* const \*amici::N\_VGetArrayPointerConst(const *N\_Vector* x)

## Function amici::operator==(ExpData const&, ExpData const&)

- Defined in file\_include\_amici\_edata.h

## Function Documentation

inline bool amici::operator==(ExpData const &lhs, ExpData const &rhs)

Equality operator.

### Parameters

- **lhs** – some object
- **rhs** – another object

**Returns** true, if both arguments are equal; false otherwise.

## Function amici::operator==(Model const&, Model const&)

- Defined in file\_include\_amici\_model.h

## Function Documentation

bool amici::operator==(Model const &a, Model const &b)

### Parameters

- **a** – First model instance
- **b** – Second model instance

**Returns** Equality

## Function amici::operator==(ModelDimensions const&, ModelDimensions const&)

- Defined in file\_include\_amici\_model.h

## Function Documentation

bool amici::operator==(ModelDimensions const &a, ModelDimensions const &b)

## Function amici::operator==(ModelState const&, ModelState const&)

- Defined in file\_include\_amici\_model\_state.h

### Function Documentation

inline bool amici::operator==(ModelState const &a, ModelState const &b)

## Function amici::operator==(SimulationParameters const&, SimulationParameters const&)

- Defined in file\_include\_amici\_simulation\_parameters.h

### Function Documentation

bool amici::operator==(SimulationParameters const &a, SimulationParameters const &b)

## Function amici::operator==(Solver const&, Solver const&)

- Defined in file\_include\_amici\_solver.h

### Function Documentation

bool amici::operator==(Solver const &a, Solver const &b)

#### Parameters

- **a** –
- **b** –

#### Returns

## Function amici::printfToString

- Defined in file\_include\_amici\_misc.h

### Function Documentation

std::string amici::printfToString(char const \*fmt, va\_list ap)

Format printf-style arguments to std::string.

#### Parameters

- **fmt** – Format string
- **ap** – Argument list pointer

**Returns** Formatted String

## Function amici::regexErrorToString

- Defined in file\_include\_amici\_misc.h

### Function Documentation

std::string amici::regexErrorToString(std::regex\_constants::error\_type err\_type)

Convert std::regex\_constants::error\_type to string.

**Parameters** **err\_type** – error type

**Returns** Error type as string

## Function amici::runAmiciSimulation

- Defined in file\_include\_amici\_amici.h

### Function Documentation

std::unique\_ptr<ReturnData> amici::runAmiciSimulation(*Solver* &solver, *ExpData* const \*edata, *Model* &model, bool rethrow = false)

Core integration routine. Initializes the solver and runs the forward and backward problem.

#### Parameters

- **solver** – *Solver* instance
- **edata** – pointer to experimental data object
- **model** – model specification object
- **rethrow** – rethrow integration exceptions?

**Returns** rdata pointer to return data object

## Function amici::runAmiciSimulations

- Defined in file\_include\_amici\_amici.h

### Function Documentation

std::vector<std::unique\_ptr<ReturnData>> amici::runAmiciSimulations(*Solver* const &solver, std::vector<*ExpData*\*> const &edatas, *Model* const &model, bool failfast, int num\_threads)

Same as runAmiciSimulation, but for multiple *ExpData* instances. When compiled with OpenMP support, this function runs multi-threaded.

#### Parameters

- **solver** – *Solver* instance
- **edatas** – experimental data objects

- **model** – model specification object
- **failfast** – flag to allow early termination
- **num\_threads** – number of threads for parallel execution

**Returns** vector of pointers to return data objects

### Function `amici::scaleParameters`

- Defined in `file_include_amici_misc.h`

### Function Documentation

```
void amici::scaleParameters(gsl::span<realtype> const& bufferUnscaled, gsl::span<ParameterScaling> const&
                             pscale, gsl::span<realtype> bufferScaled)
```

Apply parameter scaling according to `scaling`

#### Parameters

- **bufferUnscaled** –
- **pscale** – parameter scaling
- **bufferScaled** – destination

### Template Function `amici::serializeToChar`

- Defined in `file_include_amici_serialization.h`

### Function Documentation

```
template<typename T>
char *amici::serializeToChar(T const &data, int *size)
```

Serialize object to char array.

#### Parameters

- **data** – input object
- **size** – maximum char length

**Returns** The object serialized as char

### Template Function `amici::serializeToStdVec`

- Defined in `file_include_amici_serialization.h`

## Function Documentation

```
template<typename T>  
std::vector<char> amici::serializeToStdVec(T const &data)
```

Serialize object to std::vector<char>

**Parameters** *data* – input object

**Returns** The object serialized as std::vector<char>

## Template Function amici::serializeToString

- Defined in file\_include\_amici\_serialization.h

## Function Documentation

```
template<typename T>  
std::string amici::serializeToString(T const &data)
```

Serialize object to string.

**Parameters** *data* – input object

**Returns** The object serialized as string

## Function amici::simulation\_status\_to\_str

- Defined in file\_include\_amici\_amici.h

## Function Documentation

```
std::string amici::simulation_status_to_str(int status)
```

Get the string representation of the given simulation status code (see [ReturnData::status](#)).

**Parameters** *status* – Status code

**Returns** Name of the variable representing this status code.

## Template Function amici::slice(std::vector<T>&, int, unsigned)

- Defined in file\_include\_amici\_misc.h

## Function Documentation

template<class **T**>  
gsl::span<**T**> amici::slice(std::vector<**T**> &data, int index, unsigned size)  
creates a slice from existing data

### Parameters

- **data** – to be sliced
- **index** – slice index
- **size** – slice size

**Returns** span of the slice

## Template Function amici::slice(std::vector<**T**> const&, int, unsigned)

- Defined in file\_include\_amici\_misc.h

## Function Documentation

template<class **T**>  
gsl::span<**T** const> amici::slice(std::vector<**T**> const &data, int index, unsigned size)  
creates a constant slice from existing constant data

### Parameters

- **data** – to be sliced
- **index** – slice index
- **size** – slice size

**Returns** span of the slice

## Function amici::unravel\_index(size\_t, size\_t)

- Defined in file\_include\_amici\_misc.h

## Function Documentation

auto amici::unravel\_index(size\_t flat\_idx, size\_t num\_cols) -> std::pair<size\_t, size\_t>  
Convert a flat index to a pair of row/column indices, assuming row-major order.

### Parameters

- **flat\_idx** – flat index
- **num\_cols** – number of columns of referred to matrix

**Returns** row index, column index



**Function amici::unravel\_index(sunindextype, SUNMatrix)**

- Defined in file\_include\_amici\_sundials\_matrix\_wrapper.h

**Function Documentation**

auto amici::unravel\_index(sunindextype i, SUNMatrix m) -> std::pair<sunindextype, sunindextype>

Convert a flat index to a pair of row/column indices.

**Parameters**

- **i** – flat index
- **m** – referred to matrix

**Returns** row index, column index

**Function amici::unscaleParameters**

- Defined in file\_include\_amici\_misc.h

**Function Documentation**

void amici::unscaleParameters(gsl::span<realtype> const& bufferScaled, gsl::span<ParameterScaling> const& pscale, gsl::span<realtype> bufferUnscaled)

Remove parameter scaling according to the parameter scaling in pscale.

All vectors must be of same length.

**Parameters**

- **bufferScaled** – scaled parameters
- **pscale** – parameter scaling
- **bufferUnscaled** – unscaled parameters are written to the array

**Function amici::wrapErrHandlerFn**

- Defined in file\_include\_amici\_solver.h

**Function Documentation**

void amici::wrapErrHandlerFn(int error\_code, char const \*module, char const \*function, char \*msg, void \*eh\_data)

Extracts diagnosis information from solver memory block and passes them to the specified output function.

**Parameters**

- **error\_code** – error identifier
- **module** – name of the module in which the error occurred
- **function** – name of the function in which the error occurred

- **msg** – error message
- **eh\_data** – *amici::Solver* as void\*

### Template Function `amici::writeSlice(const gsl::span<T const>, gsl::span<T>)`

- Defined in `file_include_amici_misc.h`

### Function Documentation

```
template<class T>
void amici::writeSlice(const gsl::span<T const> slice, gsl::span<T> buffer)
    local helper function to write computed slice to provided buffer (span)
```

#### Parameters

- **slice** – computed value
- **buffer** – buffer to which values are to be written

### Template Function `amici::writeSlice(std::vector<T> const&, std::vector<T>&)`

- Defined in `file_include_amici_misc.h`

### Function Documentation

```
template<class T>
void amici::writeSlice(std::vector<T> const &s, std::vector<T> &b)
    local helper function to write computed slice to provided buffer (vector)
```

#### Parameters

- **s** – computed value
- **b** – buffer to which values are to be written

### Template Function `amici::writeSlice(std::vector<T> const&, gsl::span<T>)`

- Defined in `file_include_amici_misc.h`

### Function Documentation

```
template<class T>
void amici::writeSlice(std::vector<T> const &s, gsl::span<T> b)
    local helper function to write computed slice to provided buffer (vector/span)
```

#### Parameters

- **s** – computed value
- **b** – buffer to which values are to be written

**Function amici::writeSlice(AmiVector const&, gsl::span<realtype>)**

- Defined in file\_include\_amici\_misc.h

**Function Documentation**

void amici::writeSlice(*AmiVector* const &s, gsl::span<*realtype*> b)  
local helper function to write computed slice to provided buffer (AmiVector/span)

**Parameters**

- **s** – computed value
- **b** – buffer to which values are to be written

**Template Function boost::serialization::archiveVector**

- Defined in file\_include\_amici\_serialization.h

**Function Documentation**

template<class **Archive**, typename T>  
void boost::serialization::archiveVector(*Archive* &ar, T \*\*p, int size)  
Serialize a raw array to a boost archive.

**Parameters**

- **ar** – archive
- **p** – Pointer to array
- **size** – Size of p

**Template Function boost::serialization::serialize(Archive&, amici::Model&, unsigned int)**

- Defined in file\_include\_amici\_model.h

**Function Documentation**

template<class **Archive**>  
void boost::serialization::serialize(*Archive* &ar, amici::Model &m, unsigned int version)

### Template Function `boost::serialization::serialize(Archive&, amici::ReturnData&, unsigned int)`

- Defined in `file_include_amici_rdata.h`

#### Function Documentation

```
template<class Archive>
void boost::serialization::serialize(Archive &ar, amici::ReturnData &r, unsigned int version)
```

### Template Function `boost::serialization::serialize(Archive&, amici::Solver&, unsigned int)`

- Defined in `file_include_amici_solver.h`

#### Function Documentation

```
template<class Archive>
void boost::serialization::serialize(Archive &ar, amici::Solver &s, unsigned int version)
```

### Template Function `boost::serialization::serialize(Archive&, amici::CNodeSolver&, unsigned int)`

- Defined in `file_include_amici_solver_cnodes.h`

#### Function Documentation

```
template<class Archive>
void boost::serialization::serialize(Archive &ar, amici::CNodeSolver &s, unsigned int version)
```

### Template Function `boost::serialization::serialize(Archive&, amici::IDASolver&, unsigned int)`

- Defined in `file_include_amici_solver_idas.h`

#### Function Documentation

```
template<class Archive>
void boost::serialization::serialize(Archive &ar, amici::IDASolver &s, unsigned int version)
```

### Function `gsl::make_span(SUNMatrix)`

- Defined in `file_include_amici_sundials_matrix_wrapper.h`

#### Function Documentation

`inline span<realtype> gsl::make_span(SUNMatrix m)`

Create span from SUNMatrix.

**Parameters** `m` – SUNMatrix

**Returns** Created span

### Function `gsl::make_span(N_Vector)`

- Defined in `file_include_amici_vector.h`

#### Function Documentation

`inline span<realtype> gsl::make_span(N_Vector nv)`

Create span from N\_Vector.

**Parameters** `nv` –

**Returns**

#### Variables

### Variable `amici::AMICI_CONV_FAILURE`

- Defined in `file_include_amici_defines.h`

#### Variable Documentation

`constexpr int amici::AMICI_CONV_FAILURE = -4`

### Variable `amici::AMICI_DAMPING_FACTOR_ERROR`

- Defined in `file_include_amici_defines.h`

## Variable Documentation

constexpr int amici::AMICI\_DAMPING\_FACTOR\_ERROR = -86

## Variable amici::AMICI\_DATA\_RETURN

- Defined in file\_include\_amici\_defines.h

## Variable Documentation

constexpr int amici::AMICI\_DATA\_RETURN = 1

## Variable amici::AMICI\_ERR\_FAILURE

- Defined in file\_include\_amici\_defines.h

## Variable Documentation

constexpr int amici::AMICI\_ERR\_FAILURE = -3

## Variable amici::AMICI\_ERROR

- Defined in file\_include\_amici\_defines.h

## Variable Documentation

constexpr int amici::AMICI\_ERROR = -99

## Variable amici::AMICI\_FIRST\_RHSFUNC\_ERR

- Defined in file\_include\_amici\_defines.h

## Variable Documentation

constexpr int amici::AMICI\_FIRST\_RHSFUNC\_ERR = -9

**Variable amici::AMICI\_ILL\_INPUT**

- Defined in file\_include\_amici\_defines.h

**Variable Documentation**

```
constexpr int amici::AMICI_ILL_INPUT = -22
```

**Variable amici::AMICI\_MAX\_TIME\_EXCEEDED**

- Defined in file\_include\_amici\_defines.h

**Variable Documentation**

```
constexpr int amici::AMICI_MAX_TIME_EXCEEDED = -1000
```

**Variable amici::AMICI\_NO\_STEADY\_STATE**

- Defined in file\_include\_amici\_defines.h

**Variable Documentation**

```
constexpr int amici::AMICI_NO_STEADY_STATE = -81
```

**Variable amici::AMICI\_NORMAL**

- Defined in file\_include\_amici\_defines.h

**Variable Documentation**

```
constexpr int amici::AMICI_NORMAL = 1
```

**Variable amici::AMICI\_NOT\_IMPLEMENTED**

- Defined in file\_include\_amici\_defines.h

## Variable Documentation

constexpr int amici::AMICI\_NOT\_IMPLEMENTED = -999

## Variable amici::AMICI\_NOT\_RUN

- Defined in file\_include\_amici\_defines.h

## Variable Documentation

constexpr int amici::AMICI\_NOT\_RUN = -1001

## Variable amici::AMICI\_ONE\_STEP

- Defined in file\_include\_amici\_defines.h

## Variable Documentation

constexpr int amici::AMICI\_ONE\_STEP = 2

## Variable amici::AMICI\_ONEOUTPUT

- Defined in file\_include\_amici\_defines.h

## Variable Documentation

constexpr int amici::AMICI\_ONEOUTPUT = 5

## Variable amici::AMICI\_PREEQUILIBRATE

- Defined in file\_include\_amici\_defines.h

## Variable Documentation

constexpr int amici::AMICI\_PREEQUILIBRATE = -1



**Variable amici::AMICI\_RECOVERABLE\_ERROR**

- Defined in file\_include\_amici\_defines.h

**Variable Documentation**

```
constexpr int amici::AMICI_RECOVERABLE_ERROR = 1
```

**Variable amici::AMICI\_RHSFUNC\_FAIL**

- Defined in file\_include\_amici\_defines.h

**Variable Documentation**

```
constexpr int amici::AMICI_RHSFUNC_FAIL = -8
```

**Variable amici::AMICI\_ROOT\_RETURN**

- Defined in file\_include\_amici\_defines.h

**Variable Documentation**

```
constexpr int amici::AMICI_ROOT_RETURN = 2
```

**Variable amici::AMICI\_SINGULAR\_JACOBIAN**

- Defined in file\_include\_amici\_defines.h

**Variable Documentation**

```
constexpr int amici::AMICI_SINGULAR_JACOBIAN = -809
```

**Variable amici::AMICI\_SUCCESS**

- Defined in file\_include\_amici\_defines.h

## Variable Documentation

constexpr int amici::AMICI\_SUCCESS = 0

## Variable amici::AMICI\_TOO\_MUCH\_ACC

- Defined in file\_include\_amici\_defines.h

## Variable Documentation

constexpr int amici::AMICI\_TOO\_MUCH\_ACC = -2

## Variable amici::AMICI\_TOO\_MUCH\_WORK

- Defined in file\_include\_amici\_defines.h

## Variable Documentation

constexpr int amici::AMICI\_TOO\_MUCH\_WORK = -1

## Variable amici::AMICI\_UNRECOVERABLE\_ERROR

- Defined in file\_include\_amici\_defines.h

## Variable Documentation

constexpr int amici::AMICI\_UNRECOVERABLE\_ERROR = -10

## Variable amici::model\_quantity\_to\_str

- Defined in file\_include\_amici\_model.h

## Variable Documentation

const std::map<*ModelQuantity*, std::string> amici::model\_quantity\_to\_str

### Variable amici::pi

- Defined in file\_include\_amici\_defines.h

### Variable Documentation

constexpr double amici::pi = 3.14159265358979323846

### Defines

#### Define \_USE\_MATH\_DEFINES

- Defined in file\_include\_amici\_defines.h

### Define Documentation

**\_USE\_MATH\_DEFINES**

#### Define AMICI\_H5\_RESTORE\_ERROR\_HANDLER

- Defined in file\_include\_amici\_hdf5.h

### Define Documentation

**AMICI\_H5\_RESTORE\_ERROR\_HANDLER**

#### Define AMICI\_H5\_SAVE\_ERROR\_HANDLER

- Defined in file\_include\_amici\_hdf5.h

### Define Documentation

**AMICI\_H5\_SAVE\_ERROR\_HANDLER**

## Define AMICI\_VERSION

- Defined in file\_include\_amici\_version.in.h

## Define Documentation

**AMICI\_VERSION**

## Define M\_1\_PI

- Defined in file\_include\_amici\_defines.h

## Define Documentation

**M\_1\_PI**

## Define M\_2\_PI

- Defined in file\_include\_amici\_defines.h

## Define Documentation

**M\_2\_PI**

## Define M\_2\_SQRTPI

- Defined in file\_include\_amici\_defines.h

## Define Documentation

**M\_2\_SQRTPI**

## Define M\_E

- Defined in file\_include\_amici\_defines.h

## Define Documentation

**M\_E**

## Define M\_LN10

- Defined in file\_include\_amici\_defines.h

## Define Documentation

**M\_LN10**

## Define M\_LN2

- Defined in file\_include\_amici\_defines.h

## Define Documentation

**M\_LN2**

## Define M\_LOG10E

- Defined in file\_include\_amici\_defines.h

## Define Documentation

**M\_LOG10E**

## Define M\_LOG2E

- Defined in file\_include\_amici\_defines.h

## Define Documentation

**M\_LOG2E**

## Define M\_PI

- Defined in file\_include\_amici\_defines.h

## Define Documentation

**M\_PI**

## Define M\_PI\_2

- Defined in file\_include\_amici\_defines.h

## Define Documentation

**M\_PI\_2**

## Define M\_PI\_4

- Defined in file\_include\_amici\_defines.h

## Define Documentation

**M\_PI\_4**

## Define M\_SQRT1\_2

- Defined in file\_include\_amici\_defines.h

## Define Documentation

**M\_SQRT1\_2**

## Define M\_SQRT2

- Defined in file\_include\_amici\_defines.h

## Define Documentation

**M\_SQRT2**

## Typedefs

### Typedef amici::const\_N\_Vector

- Defined in file\_include\_amici\_vector.h

## Typedef Documentation

using amici::const\_N\_Vector = std::add\_const\_t<typename std::remove\_pointer\_t<N\_Vector>>\*

Since const N\_Vector is not what we want

### Typedef amici::realtype

- Defined in file\_include\_amici\_defines.h

## Typedef Documentation

using amici::realtype = double

defines variable type for simulation variables (determines numerical accuracy)





## MATLAB INTERFACE

### 12.1 Installing the AMICI MATLAB toolbox

To use AMICI from MATLAB, start MATLAB and add the `AMICI/matlab` directory to the MATLAB path. To add all toolbox directories to the MATLAB path, execute the matlab script:

```
installAMICI.m
```

To store the installation for further MATLAB session, the path can be saved via:

```
savepath
```

For the compilation of `.mex` files, MATLAB needs to be configured with a working C++ compiler. The C++ compiler needs to be installed and configured via:

```
mex -setup c++
```

For a list of supported compilers we refer to the respective MathWorks [documentation](#).

### 12.2 Using AMICI's MATLAB interface

In the following we will give a detailed overview how to specify models in MATLAB and how to call the generated simulation files.

---

**Note:** The MATLAB interface requires the MathWorks [Symbolic Math Toolbox](#) for model import (but not for model simulation).

The Symbolic Math Toolbox requirement can be circumvented by performing model import using the Python interface. The resulting code can then be used from Matlab (see [Compiling a Python-generated model](#)).

---

**Warning:** Due to changes in the Symbolic Math Toolbox, the last MATLAB release with working AMICI model import is R2017b (see <https://github.com/AMICI-dev/AMICI/issues/307>).

## 12.2.1 Specifying models in Matlab

This guide will guide the user on how to specify models in MATLAB. For example implementations see the examples in the `matlab/examples` directory.

### Header

The model definition needs to be defined as a function which returns a `struct` with all symbolic definitions and options.

```
function [model] = example_model_syms()
```

### Options

Set the options by specifying the respective field of the model struct

```
model.(fieldname) = value
```

The options specify default options for simulation, parametrisation and compilation. All of these options are optional.

field	description	default
.param	default parametrisation 'log'/'log10'/'lin'	'lin'
.debug	flag to compile with debug symbols	false
.forward	flag to activate forward sensitivities	true
.adjoint	flag to activate adjoint sensitivities	true

When set to false, the fields `forward` and `adjoint` will speed up the time required to compile the model but also disable the respective sensitivity computation.

### States

Create the respective symbolic variables. The name of the symbolic variable can be chosen arbitrarily.

```
syms state1 state2 state3
```

Create the state vector containing all states:

```
model.sym.x = [ state1 state2 state3 ];
```

### Parameters

Create the respective symbolic variables. The name of the symbolic variable can be chosen arbitrarily. Sensitivities will be derived for all *parameters*.

```
syms param1 param2 param3 param4 param5 param6
```

Create the parameters vector

```
model.sym.p = [ param1 param2 param3 param4 param5 param6 ];
```

## Constants

Create the respective symbolic variables. The name of the symbolic variable can be chosen arbitrarily. Sensitivities with respect to *constants* will not be derived.

```
syms const1 const2
```

Create the constants vector

```
model.sym.k = [ const1 const2 ];
```

## Differential equations

For time-dependent differential equations you can specify a symbolic variable for time. This **needs** to be denoted by `t`.

```
syms t
```

Specify the right hand side of the differential equation `f` or `xdot`

```
model.sym.xdot(1) = [ const1 - param1*state1 ];
model.sym.xdot(2) = [ +param2*state1 + dirac(t-param3) - const2*state2 ];
model.sym.xdot(3) = [ param4*state2 ];
```

or

```
model.sym.f(1) = [ const1 - param1*state1 ];
model.sym.f(2) = [ +param2*state1 + dirac(t-param3) - const2*state2 ];
model.sym.f(3) = [ param4*state2 ];
```

The specification of `f` or `xdot` may depend on states, parameters and constants.

For DAEs also specify the mass matrix.

```
model.sym.M = [1, 0, 0;...
               0, 1, 0;...
               0, 0, 0];
```

The specification of `M` may depend on parameters and constants.

For ODEs the integrator will solve the equation  $\dot{x} = f$  and for DAEs the equations  $M \cdot \dot{x} = f$ . AMICI will decide whether to use CVODES (for ODEs) or IDAS (for DAEs) based on whether the mass matrix is defined or not.

In the definition of the differential equation you can use certain symbolic functions. For a full list of available functions see `src/symbolic_functions.cpp`.

Dirac functions can be used to cause a jump in the respective states at the specified time-point. This is typically used to model injections, or other external stimuli. Spline functions can be used to model time/state dependent response with unknown time/state dependence.

## Initial Conditions

Specify the initial conditions. These may depend on parameters on constants and must have the same size as `x`.

```
model.sym.x0 = [ param4, 0, 0 ];
```

## Observables

Specify the observables. These may depend on parameters and constants.

```
model.sym.y(1) = state1 + state2;  
model.sym.y(2) = state3 - state2;
```

In the definition of the observable you can use certain symbolic functions. For a full list of available functions see `src/symbolic_functions.cpp`. Dirac functions in observables will have no effect.

## Events

Specifying events is optional. Events are specified in terms of a trigger function, a bolus function and an output function. The roots of the trigger function defines the occurrences of the event. The bolus function defines the change in the state on event occurrences. The output function defines the expression which is evaluated and reported by the simulation routine on every event occurrence. The user can create events by constructing a vector of objects of the class [amievent](#).

```
model.sym.event(1) = amievent(state1 - state2,0,[]);
```

Events may depend on states, parameters and constants but *not* on observables.

For more details about event support see:

Fröhlich, F., Theis, F. J., Rädler, J. O., & Hasenauer, J. (2017). Parameter estimation for dynamical systems with discrete events and logical operations. *Bioinformatics*, 33(7), 1049-1056. doi:10.1093/bioinformatics/btw764.

## Standard deviation

Specifying standard deviations is optional. It only has an effect when computing adjoint sensitivities. It allows the user to specify standard deviations of experimental data for observables and events.

Standard deviation for observable data is denoted by `sigma_y`

```
model.sym.sigma_y(1) = param5;
```

Standard deviation for event data is denoted by `sigma_t`

```
model.sym.sigma_t(1) = param6;
```

Both `sigma_y` and `sigma_t` can either be a scalar or of the same dimension as the observables / events function. They can depend on time and parameters but must not depend on the states or observables. The values provided in `sigma_y` and `sigma_t` will only be used if the value in `D.Sigma_Y` or `D.Sigma_T` in the user-provided data struct is NaN. See simulation for details.

## Objective Function

By default, AMICI assumes a normal noise model and uses the corresponding negative log-likelihood

$$J = 1/2 * \text{sum}(((y_i(t) - my_{ti})/\sigma_{y_i})^2 + \log(2 * \pi * \sigma_{y_i}^2))$$

as objective function. A user provided objective function can be specified in

```
model.sym.Jy
```

As reference see the default specification of `this.sym.Jy` in `amimodel.makeSyms`.

### 12.2.2 SBML

AMICI can also import SBML models using the command `SBML2AMICI`. This will generate a model specification as described above, which may be edited by the user to apply further changes.

### 12.2.3 Model Compilation

The model can then be compiled by calling `amiwrap.m`:

```
amiwrap(modelname, 'example_model_syms', dir, o2flag)
```

Here `modelname` should be a string defining the name of the model, `dir` should be a string containing the path to the directory in which simulation files should be placed and `o2flag` is a flag indicating whether second order sensitivities should also be compiled. The user should make sure that the previously defined function `example_model_syms` is in the user path. Alternatively, the user can also call the function `example_model_syms`

```
[model] = example_model_syms()
```

and subsequently provide the generated struct to `amiwrap(...)`, instead of providing the symbolic function:

```
amiwrap(modelname, model, dir, o2flag)
```

In a similar fashion, the user could also generate multiple models and pass them directly to `amiwrap(...)` without generating respective model definition scripts.

### Compiling a Python-generated model

For better performance or to avoid the Symbolic Math Toolbox requirement, it might be desirable to import a model in Python and compile the resulting code into a mex file. For Python model import, consult the respective section of the Python documentation. Once the import succeeded, there will be a `compileMexFile.m` script inside the newly created model directory which can be invoked to compile the mex file. This mex file and `simulate_*.m` can be used as if fully created by matlab.

## Using Python-AMICI model import from Matlab

With recent matlab versions it is possible to use the AMICI python package from within Matlab. This not quite comfortable yet, but it is possible.

Here for proof of concept:

- Install the python package as described in the documentation
- Ensure pyversion shows the correct python version (3.6 or 3.7)
- Then, from within the AMICI matlab/ directory:

```
sbml_importer = py.amici.SbmlImporter('../python/examples/example_steadystate/model_
↪steadystate_scaled.xml')
sbml_importer.sbml2amici('steadystate', 'steadystate_example_from_python')
model = py.steadystate.getModel()
solver = model.getSolver()
model.setTimepoints(linspace(0, 50, 51))
rdata = py.amici.runAmiciSimulation(model, solver)
result = struct(py.dict(rdata.items()))
t = double(py.array.array('d', result.ts))
x = double(py.array.array('d', result.x.flatten()))
x = reshape(x, flip(double(py.array.array('d', result.x.shape))))
plot(t, x)
```

### 12.2.4 Model simulation

After the call to `amiwrap(...)` two files will be placed in the specified directory. One is a `_modelname_.mex` and the other is `simulate_*modelname*.m`. The mex file should never be called directly. Instead the MATLAB script, which acts as a wrapper around the .mex simulation file should be used.

The `simulate_ _modelname_.m` itself carries extensive documentation on how to call the function, what it returns and what additional options can be specified. In the following we will give a short overview of possible function calls.

#### Integration

Define a time vector:

```
t = linspace(0,10,100)
```

Generate a parameter vector:

```
theta = ones(6,1);
```

Generate a constants vector:

```
kappa = ones(2,1);
```

Integrate:

```
sol = simulate_modelname(t,theta,kappa,[],options)
```

The integration status will be indicated by the `sol.status` flag. Negative values indicated failed integration. The states will then be available as `sol.x`. The observables will then be available as `sol.y`. The event outputs will then

be available as `sol.z`. If no event occurred there will be an event at the end of the considered interval with the final value of the root function is stored in `sol.rz`.

Alternatively the integration can also be called via

```
[status,t,x,y] = simulate_modelname(t,theta,kappa,[],options)
```

The integration status will be indicated by the flag `status`. Negative values indicated failed integration. The states will then be available as `x`. The observables will then be available as `y`. No event output will be given.

## Forward Sensitivities

Set the sensitivity computation to forward sensitivities and integrate:

```
options.sensi = 1;
options.sensi_meth = 'forward';
sol = simulate_modelname(t,theta,kappa,[],options)
```

The integration status will be indicated by the `sol.status` flag. Negative values indicate failed integration. The states will be available as `sol.x`, with the derivative with respect to the parameters in `sol.sx`. The observables will be available as `sol.y`, with the derivative with respect to the parameters in `sol.sy`. The event outputs will be available as `sol.z`, with the derivative with respect to the parameters in `sol.sz`. If no event occurred there will be an event at the end of the considered interval with the final value of the root function stored in `sol.rz`, with the derivative with respect to the parameters in `sol.srz`.

Alternatively the integration can also be called via

```
[status,t,x,y,sx,sy] = simulate_modelname(t,theta,kappa,[],options)
```

The integration status will be indicated by the `status` flag. Negative values indicate failed integration. The states will then be available as `x`, with derivative with respect to the parameters in `sx`. The observables will then be available as `y`, with derivative with respect to the parameters in `sy`. No event output will be given.

## Adjoint sensitivities

Set the sensitivity computation to adjoint sensitivities:

```
options.sensi = 1;
options.sensi_meth = 'adjoint';
```

Define Experimental Data:

```
D.Y = [NaN(1,2)],ones(length(t)-1,2)];
D.Sigma_Y = [0.1*ones(length(t)-1,2),NaN(1,2)];
D.T = ones(1,1);
D.Sigma_T = NaN;
```

The NaN values in `Sigma_Y` and `Sigma_T` will be replaced by the specification in `model.sym.sigma_y` and `model.sym.sigma_t`. Data points with NaN value will be completely ignored.

Integrate:

```
sol = simulate_modelname(t,theta,kappa,D,options)
```

The integration status will be indicated by the `sol.status` flag. Negative values indicate failed integration. The log-likelihood will then be available as `sol.llh` and the derivative with respect to the parameters in `sol.sllh`. Note that for adjoint sensitivities no state, observable and event sensitivities will be available. Yet this approach can be expected to be significantly faster for systems with a large number of parameters.

### Steady-state sensitivities

This will compute state sensitivities according to the formula

$$s_k^x = - \left( \frac{\partial f}{\partial x} \right)^{-1} \frac{\partial f}{\partial \theta_k}$$

In the current implementation this formulation does not allow for conservation laws as this would result in a singular Jacobian.

Set the final timepoint as infinity, this will indicate the solver to compute the steadystate:

```
t = Inf;
```

Set the sensitivity computation to steady state sensitivities:

```
options.sensi = 1;
```

Integrate:

```
sol = simulate_modelname(t,theta,kappa,D,options)
```

The states will be available as `sol.x`, with the derivative with respect to the parameters in `sol.sx`. The observables will be available as `sol.y`, with the derivative with respect to the parameters in `sol.sy`. Notice that for steady state sensitivities no event sensitivities will be available. For the accuracy of the computed derivatives it is essential that the system is sufficiently close to a steady state. This can be checked by examining the right hand side of the system at the final time-point via `sol.diagnosis.xdot`.

## 12.3 FAQ

**Q:** My model fails to build.

**A:** Remove the corresponding model directory located in `AMICI/models/yourmodelname` and compile again.

---

**Q:** It still does not compile.

**A:** Remove the directory `AMICI/models/mexext` and compile again.

---

**Q:** It still does not compile.

**A:** Make an [issue](#) and we will have a look.

---

**Q:** My Python-generated model does not compile from MATLAB.

**A:** Try building any of the available examples before. If this succeeds, retry building the original model. Some dependencies might not be built correctly when using only the `compileMexFile.m` script.

---



**Q:** I get an out of memory error while compiling my model on a Windows machine.

**A:** This may be due to an old compiler version. See [issue #161](#) for instructions on how to install a new compiler.

---

**Q:** How are events interpreted in a DAE context?

**A:** Currently we only support impulse free events. Also sensitivities have never been tested. Proceed with care and create an [issue](#) if any problems arise!

---

**Q:** The simulation/sensitivities I get are incorrect.

**A:** There are some known issues, especially with adjoint sensitivities, events and DAEs. If your particular problem is not featured in the [issues](#) list, please add it!

## 12.4 AMICI Matlab API

AMICI Matlab library functions

### 12.4.1 Class Hierarchy

### 12.4.2 File Hierarchy

### 12.4.3 Full API

#### Namespaces

#### Namespace matlab

##### Contents

- [Namespaces](#)

#### Namespaces

- [Namespace matlab::mixin](#)

#### Namespace matlab::mixin

#### Classes and Structs

#### Class amidata

- Defined in file\_matlab\_@amidata\_amidata.m

## Inheritance Relationships

### Base Type

- public handle

### Class Documentation

#### **amidata : public handle**

AMIDATA provides a data container to pass experimental data to the simulation routine for likelihood computation. when any of the properties are updated, the class automatically checks consistency of dimension and updates related properties and initialises them with NaNs.

### Public Functions

#### **amidata::amidata(matlabtypesubstitute varargin)**

amidata creates an amidata container for experimental data with specified dimensions amidata.

AMIDATA(amidata) creates a copy of the input container

AMIDATA(struct) tries to creates an amidata container from the input struct. the struct should have the following

AMIDATA(nt,ny,nz,ne,nk) constructs an empty data container with in the provided dimensions intialised with NaNs

**fields** t [nt,1] Y [nt,ny] Sigma\_Y [nt,ny] Z [ne,nz] Sigma\_Z [ne,nz] condition [nk,1] conditionPreequilibration [nk,1] if some fields are missing the function will try to initialise them with NaNs with consistent dimensions

**param varargin**

### Public Members

**matlabtypesubstitute nt = 0**

number of timepoints

**Default:** 0

---

**Note:** This property has custom functionality when its value is changed.

---

**matlabtypesubstitute ny = 0**

number of observables

**Default:** 0

---

**Note:** This property has custom functionality when its value is changed.

---

**matlabtypesubstitute nz = 0**

number of event observables

**Default:** 0

---

**Note:** This property has custom functionality when its value is changed.

---

**matlabtypesubstitute ne = 0**

number of events

**Default:** 0

---

**Note:** This property has custom functionality when its value is changed.

---

**matlabtypesubstitute nk = 0**

number of conditions/constants

**Default:** 0

---

**Note:** This property has custom functionality when its value is changed.

---

**matlabtypesubstitute t = double.empty("")**

timepoints of observations

**Default:** double.empty("")

---

**Note:** This property has custom functionality when its value is changed.

---

**matlabtypesubstitute Y = double.empty("")**

observations

**Default:** double.empty("")

---

**Note:** This property has custom functionality when its value is changed.

---

**matlabtypesubstitute Sigma\_Y = double.empty("")**

standard deviation of observations

**Default:** double.empty("")

---

**Note:** This property has custom functionality when its value is changed.

---

**matlabtypesubstitute Z** = double.empty("")

event observations

**Default:** double.empty("")

---

**Note:** This property has custom functionality when its value is changed.

---

**matlabtypesubstitute Sigma\_Z** = double.empty("")

standard deviation of event observations

**Default:** double.empty("")

---

**Note:** This property has custom functionality when its value is changed.

---

**matlabtypesubstitute condition** = double.empty("")

experimental condition

**Default:** double.empty("")

---

**Note:** This property has custom functionality when its value is changed.

---

**matlabtypesubstitute conditionPreequilibration** = double.empty("")

experimental condition for preequilibration

**Default:** double.empty("")

---

**Note:** This property has custom functionality when its value is changed.

---

**matlabtypesubstitute reinitializeStates** = false

reinitialize states based on fixed parameters after preeq.?

**Default:** false

## Class amievent

- Defined in file\_matlab\_@amievent\_amievent.m

## Class Documentation

### class amievent

AMIEVENT defines events which later on will be transformed into appropriate C code.

### Public Functions

**amievent::amievent(matlabtypesubstitute trigger, matlabtypesubstitute bolus, matlabtypesubstitute z)**

amievent constructs an amievent object from the provided input.

**param trigger** trigger function, the event will be triggered on at all roots of this function

**param bolus** the bolus that will be added to all states on every occurrence of the event

**param z** the event output that will be reported on every occurrence of the event

**mlhsInnerSubst<::amievent > amievent::setHflag(matlabtypesubstitute hflag)**

setHflag sets the hflag property.

**param hflag** value for the hflag property, type double

**retval this** updated event definition object

### Public Members

**::symbolic trigger = sym.empty("")**

the trigger function activates the event on every zero crossing

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** sym.empty("")

---

**::symbolic bolus = sym.empty("")**

the bolus function defines the change in states that is applied on every event occurrence

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** sym.empty("")

---

```
::symbolic z = sym.empty("")
```

output function for the event

---

**Note:** This property has non-standard access specifiers: `SetAccess = Private`, `GetAccess = Public`

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** `sym.empty("")`

---

```
matlabtypesubstitute hflag = logical.empty("")
```

flag indicating that a heaviside function is present, this helps to speed up symbolic computations

---

**Note:** This property has non-standard access specifiers: `SetAccess = Private`, `GetAccess = Public`

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** `logical.empty("")`

---

## Class amifun

- Defined in file\_matlab\_@amifun\_amifun.m

## Class Documentation

### class amifun

AMIFUN defines functions which later on will be transformed into appropriate C code.

### Public Functions

```
amifun::amifun(matlabtypesubstitute funstr, matlabtypesubstitute model)
```

amievent constructs an amifun object from the provided input.

**param funstr** name of the requested function

**param model** amimodel object which carries all symbolic definitions to construct the function

```
noret::substitute amifun::writeCcode_sensi(::amimodel model, ::fileid fid)
```

writeCcode\_sensi is a wrapper for writeCcode which loops over parameters and reduces overhead by check nonzero values

**param model** model definition object

**param fid** file id in which the final expression is written

**retval fid** void

**noret::substitute amifun::writeCcode(::amimodel model, ::fileid fid)**

writeCcode is a wrapper for gccode which initialises data and reduces overhead by check nonzero values

**param model** model definition object

**param fid** file id in which the final expression is written

**retval fid** void

**noret::substitute amifun::writeMcode(::amimodel model)**

writeMcode generates matlab evaluable code for specific model functions

**param model** model definition object

**retval model** void

**mlhsInnerSubst<::amifun > amifun::gccode(::amimodel model, ::fileid fid)**

gccode transforms symbolic expressions into c code and writes the respective expression into a specified file

**param model** model definition object

**param fid** file id in which the expression should be written

**retval this** function definition object

**mlhsInnerSubst<::amifun > amifun::getDeps(::amimodel model)**

getDeps populates the sensiflag for the requested function

**param model** model definition object

**retval this** updated function definition object

**mlhsInnerSubst<::amifun > amifun::getArgs(::amimodel model)**

getFArgs populates the fargstr property with the argument string of the respective model function (if applicable). model functions are not wrapped versions of functions which have a model specific name and for which the call is solver specific.

**param model** model definition object

**retval this** updated function definition object

**mlhsInnerSubst<::amifun > amifun::getNVecs()**

getfunargs populates the nvecs property with the names of the N\_Vector elements which are required in the execution of the function (if applicable). the information is directly extracted from the argument string

**retval this** updated function definition object

**mlhsInnerSubst<::amifun > amifun::getCVar()**

getCVar populates the cvar property

**retval this** updated function definition object

**mlhsInnerSubst<::amifun > amifun::getSensiFlag()**

getSensiFlag populates the sensiflag property

**retval this** updated function definition object

```
mlhsSubst< mlhsInnerSubst<::amifun >,mlhsInnerSubst<::amimodel > > amifun::getSyms(:  
:amimodel model)
```

getSyms computes the symbolic expression for the requested function

**param model** model definition object

**retval this** updated function definition object

**retval model** updated model definition object

## Public Members

```
::symbolic sym = sym("[]")
```

symbolic definition struct

**Default:** sym("")

```
::symbolic sym_noopt = sym("[]")
```

symbolic definition which was not optimized (no dependencies on w)

**Default:** sym("")

```
::symbolic strsym = sym("[]")
```

short symbolic string which can be used for the reuse of precomputed values

**Default:** sym("")

```
::symbolic strsym_old = sym("[]")
```

short symbolic string which can be used for the reuse of old values

**Default:** sym("")

```
::char funstr = char.empty("")
```

name of the model

**Default:** char.empty("")

```
::char cvar = char.empty("")
```

name of the c variable

**Default:** char.empty("")

```
::char argstr = char.empty("")
```

argument string (solver specific)

**Default:** char.empty("")

```
::cell deps = cell.empty("")
```

dependencies on other functions

**Default:** cell.empty("")



**matlabtypesubstitute nvecs** = `cell.empty("")`

nvec dependencies

**Default:** `cell.empty("")`

**matlabtypesubstitute sensiflag** = `logical.empty("")`

indicates whether the function is a sensitivity or derivative with respect to parameters

**Default:** `logical.empty("")`

## Class amimodel

- Defined in file\_matlab\_@amimodel\_amimodel.m

## Inheritance Relationships

### Base Type

- public handle

## Class Documentation

### **amimodel** : public handle

AMIMODEL carries all model definitions including functions and events.

### Public Functions

**amimodel::amimodel(::string symfun, ::string modelname)**

amimodel initializes the model object based on the provided symfun and modelname

**param symfun** this is the string to the function which generates the modelstruct. You can also directly pass the struct here

**param modelname** name of the model

**noret::substitute amimodel::updateRHS(matlabtypesubstitute xdot)**

updateRHS updates the private fun property .fun.xdot.sym (right hand side of the differential equation)

**param xdot** new right hand side of the differential equation

**retval xdot** void

**noret::substitute amimodel::updateModelName(matlabtypesubstitute modelname)**

updateModelName updates the modelname

**param modelname** new modelname

**retval modelname** void

**noret::substitute amimodel::updateWrapPath(matlabtypesubstitute wrap\_path)**

updateModelName updates the modelName

**param wrap\_path** new wrap\_path

**retval wrap\_path** void

**noret::substitute amimodel::parseModel()**

parseModel parses the model definition and computes all necessary symbolic expressions.

**retval void**

**noret::substitute amimodel::generateC()**

generateC generates the c files which will be used in the compilation.

**retval void**

**noret::substitute amimodel::generateRebuildM()**

generateRebuildM generates a Matlab script for recompilation of this model

**retval void**

**noret::substitute amimodel::compileC()**

compileC compiles the mex simulation file

**retval void**

**noret::substitute amimodel::generateM(::amimodel amimodelo2)**

generateM generates the matlab wrapper for the compiled C files.

**param amimodelo2** this struct must contain all necessary symbolic definitions for second order sensitivities

**retval amimodelo2** void

**noret::substitute amimodel::getFun(::struct HTable, ::string funstr)**

getFun generates symbolic expressions for the requested function.

**param HTable** struct with hashes of symbolic definition from the previous compilation

**param funstr** function for which symbolic expressions should be computed

**retval funstr** void

**noret::substitute amimodel::makeEvents()**

makeEvents extracts discontinuities from the model right hand side and converts them into events

**retval void**

**noret::substitute amimodel::makeSyms()**

makeSyms extracts symbolic definition from the user provided model and checks them for consistency

**retval void**

**mlhsInnerSubst<:bool > amimodel::checkDeps(:struct HTable, :cell deps)**

checkDeps checks the dependencies of functions and populates sym fields if necessary

**param HTable** struct with reference hashes of functions in its fields

**param deps** cell array with containing a list of dependencies

**retval cflag** boolean indicating whether any of the dependencies have changed with respect to the hashes stored in HTable

**mlhsInnerSubst<:struct > amimodel::loadOldHashes()**

loadOldHashes loads information from a previous compilation of the model.

**retval HTable** struct with hashes of symbolic definition from the previous compilation

**mlhsInnerSubst< matlabtypesubstitute > amimodel::augmento2()**

augmento2 augments the system equation to also include equations for sensitivity equation. This will enable us to compute second order sensitivities in a forward-adjoint or forward-forward approach later on.

**retval this** augmented system which contains symbolic definition of the original system and its sensitivities

**mlhsInnerSubst<:amimodel > amimodel::augmento2vec()**

augmento2 augments the system equation to also include equations for sensitivity equation. This will enable us to compute second order sensitivities in a forward-adjoint or forward-forward approach later on.

**retval modelo2vec** augmented system which contains symbolic definition of the original system and its sensitivities

## Public Members

**::struct sym = struct.empty("")**

symbolic definition struct

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** struct.empty("")

---

**::struct fun = struct.empty("")**

struct which stores information for which functions c code needs to be generated

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** struct.empty("")

---

**::amievent event = amievent.empty("")**

struct which stores information for which functions c code needs to be generated

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** amievent.empty("")

---

**::string modelname = char.empty("")**

name of the model

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** char.empty("")

---

**::struct HTable = struct.empty("")**

struct that contains hash values for the symbolic model definitions

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** struct.empty("")

---

**::bool debug = false**

flag indicating whether debugging symbols should be compiled

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** false

---

**::bool adjoint = true**

flag indicating whether adjoint sensitivities should be enabled

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** true

---

**::bool forward = true**

flag indicating whether forward sensitivities should be enabled

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** true

---

**::double t0 = 0**

default initial time

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** 0

---

**::string wtype = char.empty("")**

type of wrapper (cvides/idas)

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** char.empty("")

---

**::int nx = double.empty("")**

number of states

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** double.empty("")

---

**::int nxtrue = double.empty("")**

number of original states for second order sensitivities

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** double.empty("")

---

**::int ny = double.empty("")**

number of observables

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** double.empty("")

---

**::int nytrue = double.empty("")**

number of original observables for second order sensitivities

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** double.empty("")

---

**::int np = double.empty("")**

number of parameters

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** double.empty("")

---

**::int nk = double.empty("")**

number of constants

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** double.empty("")

---

---

```
::int ng = double.empty("")
```

number of objective functions

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** double.empty("")

---

```
::int nevent = double.empty("")
```

number of events

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** double.empty("")

---

```
::int nz = double.empty("")
```

number of event outputs

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** double.empty("")

---

```
::int nztrue = double.empty("")
```

number of original event outputs for second order sensitivities

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** double.empty("")

---

```
::*int id = double.empty("")
```

flag for DAEs

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** `double.empty("")`

---

**::int ubw** = `double.empty("")`

upper Jacobian bandwidth

---

**Note:** This property has non-standard access specifiers: `SetAccess = Private`, `GetAccess = Public`

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** `double.empty("")`

---

**::int lbw** = `double.empty("")`

lower Jacobian bandwidth

---

**Note:** This property has non-standard access specifiers: `SetAccess = Private`, `GetAccess = Public`

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** `double.empty("")`

---

**::int nnz** = `double.empty("")`

number of nonzero entries in Jacobian

---

**Note:** This property has non-standard access specifiers: `SetAccess = Private`, `GetAccess = Public`

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** `double.empty("")`

---

**::\*int sparseidx** = `double.empty("")`

dataindexes of sparse Jacobian

---

**Note:** This property has non-standard access specifiers: `SetAccess = Private`, `GetAccess = Public`

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** `double.empty("")`

---

**::\*int rowvals** = `double.empty("")`

rowindexes of sparse Jacobian



---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** double.empty("")

---

**::\*int colptrs = double.empty("")**  
columnindexes of sparse Jacobian

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** double.empty("")

---

**::\*int sparseidxB = double.empty("")**  
dataindexes of sparse Jacobian

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** double.empty("")

---

**::\*int rowvalsB = double.empty("")**  
rowindexes of sparse Jacobian

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** double.empty("")

---

**::\*int colptrsB = double.empty("")**  
columnindexes of sparse Jacobian

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** double.empty("")

---

**::\*cell funs** = `cell.empty("")`

cell array of functions to be compiled

---

**Note:** This property has non-standard access specifiers: `SetAccess = Private`, `GetAccess = Public`

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** `cell.empty("")`

---

**::\*cell mfun**s = `cell.empty("")`

cell array of matlab functions to be compiled

---

**Note:** This property has non-standard access specifiers: `SetAccess = Private`, `GetAccess = Public`

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** `cell.empty("")`

---

**::string coptim** = `"-O3"`

optimisation flag for compilation

---

**Note:** This property has non-standard access specifiers: `SetAccess = Private`, `GetAccess = Public`

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** `"-O3"`

---

**::string param** = `"lin"`

default parametrisation

---

**Note:** This property has non-standard access specifiers: `SetAccess = Private`, `GetAccess = Public`

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** `"lin"`

---

**matlabtypesubstitute wrap\_path** = `char.empty("")`

path to wrapper

---

**Note:** This property has non-standard access specifiers: `SetAccess = Private`, `GetAccess = Public`

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** `char.empty("")`

---

**matlabtypesubstitute recompile = false**

flag to enforce recompilation of the model

---

**Note:** This property has non-standard access specifiers: `SetAccess = Private`, `GetAccess = Public`

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** `false`

---

**matlabtypesubstitute cfun = struct.empty("")**

storage for flags determining recompilation of individual functions

---

**Note:** This property has non-standard access specifiers: `SetAccess = Private`, `GetAccess = Public`

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** `struct.empty("")`

---

**matlabtypesubstitute o2flag = 0**

flag which identifies augmented models 0 indicates no augmentation 1 indicates augmentation by first order sensitivities (yields second order sensitivities) 2 indicates augmentation by one linear combination of first order sensitivities (yields hessian-vector product)

---

**Note:** This property has non-standard access specifiers: `SetAccess = Private`, `GetAccess = Public`

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** `0`

---

**matlabtypesubstitute z2event = double.empty("")**

vector that maps outputs to events

**Default:** `double.empty("")`

**matlabtypesubstitute splineflag = false**

flag indicating whether the model contains spline functions

**Default:** `false`

**matlabtypesubstitute minflag = false**

flag indicating whether the model contains min functions

**Default:** `false`

**matlabtypesubstitute maxflag = false**

flag indicating whether the model contains max functions

**Default:** false

**::int nw = 0**

number of derived variables w, w is used for code optimization to reduce the number of frequently occurring expressions

**Default:** 0

**::int ndwdx = 0**

number of derivatives of derived variables w, dwdx

**Default:** 0

**::int ndwdp = 0**

number of derivatives of derived variables w, dwdp

**Default:** 0

## Public Static Functions

**noret::substitute amimodel::compileAndLinkModel(matlabtypesubstitute modelname, matlabtypesubstitute modelSourceFolder, matlabtypesubstitute coptim, matlabtypesubstitute debug, matlabtypesubstitute funs, matlabtypesubstitute cfun)**

compileAndLinkModel compiles the mex simulation file. It does not check if the model files have changed since generating C++ code or whether all files are still present. Use only if you know what you are doing. The safer alternative is rerunning [amiwrap\(\)](#).

**param modelname** name of the model as specified for [amiwrap\(\)](#)

**param modelSourceFolder** path to model source directory

**param coptim** optimization flags

**param debug** enable debugging

**param funs** array with names of the model functions, will be guessed from source files if left empty

**param cfun** struct indicating which files should be recompiled

**retval cfun** void

**noret::substitute amimodel::generateMatlabWrapper(matlabtypesubstitute nx, matlabtypesubstitute ny, matlabtypesubstitute np, matlabtypesubstitute nk, matlabtypesubstitute nz, matlabtypesubstitute o2flag, ::amimodel amimodelo2, matlabtypesubstitute wrapperFilename, matlabtypesubstitute modelname, matlabtypesubstitute pscale, matlabtypesubstitute forward, matlabtypesubstitute adjoint)**

generateMatlabWrapper generates the matlab wrapper for the compiled C files.

**param nx** number of states

**param ny** number of observables

**param np** number of parameters  
**param nk** number of fixed parameters  
**param nz** number of events  
**param o2flag** o2flag  
**param amimodelo2** this struct must contain all necessary symbolic definitions for second order sensitivities  
**param wrapperFilename** output filename  
**param modelname** name of the model  
**param pscale** default parameter scaling  
**param forward** has forward sensitivity equations  
**param adjoint** has adjoint sensitivity equations  
**retval adjoint** void

## Class amioption

- Defined in file\_matlab\_@amioption\_amioption.m

## Inheritance Relationships

### Base Type

- public matlab::mixin::CustomDisplay

## Class Documentation

**amioption : public matlab::mixin::CustomDisplay**

AMIOPTION provides an option container to pass simulation parameters to the simulation routine.

### Public Functions

**amioption::amioption(matlabtypesubstitute varargin)**

amioptions Construct a new amioptions object `OPTS = amioption()` creates a set of options with each option set to its default value.

`OPTS = amioption(PARAM, VAL, ...)` creates a set of options with the named parameters altered with the specified values.

`OPTS = amioption(OLDOPTS, PARAM, VAL, ...)` creates a copy of OLDOPTS with the named parameters altered with the specified value

Note: to see the parameters, check the documentation page for amioption

**param varargin** input to construct amioption object, see function function description

## Public Members

**matlabtypesubstitute atol = 1e-16**

absolute integration tolerance

**Default:** 1e-16

**matlabtypesubstitute rtol = 1e-8**

relative integration tolerance

**Default:** 1e-8

**matlabtypesubstitute maxsteps = 1e4**

maximum number of integration steps

**Default:** 1e4

**matlabtypesubstitute quad\_atol = 1e-12**

absolute quadrature tolerance

**Default:** 1e-12

**matlabtypesubstitute quad\_rtol = 1e-8**

relative quadrature tolerance

**Default:** 1e-8

**matlabtypesubstitute maxstepsB = 0**

maximum number of integration steps

**Default:** 0

**matlabtypesubstitute ss\_atol = 1e-16**

absolute steady state tolerance

**Default:** 1e-16

**matlabtypesubstitute ss\_rtol = 1e-8**

relative steady state tolerance

**Default:** 1e-8

**matlabtypesubstitute sens\_ind = double.empty("")**

index of parameters for which the sensitivities are computed

**Default:** double.empty("")

**matlabtypesubstitute tstart = 0**

starting time of the simulation

**Default:** 0

**matlabtypesubstitute lmm = 2**

linear multistep method.

**Default:** 2

**matlabtypesubstitute iter = 2**

iteration method for linear multistep.

**Default:** 2

**matlabtypesubstitute linsol = 9**

linear solver

**Default:** 9

**matlabtypesubstitute stldet = true**

stability detection flag

**Default:** true

**matlabtypesubstitute interpType = 1**

interpolation type

**Default:** 1

**matlabtypesubstitute ism = 1**

forward sensitivity mode

**Default:** 1

**matlabtypesubstitute sensi\_meth = 1**

sensitivity method

**Default:** 1

---

**Note:** This property has custom functionality when its value is changed.

---

**matlabtypesubstitute sensi\_meth\_preeq = 1**

sensitivity method for preequilibration

**Default:** 1

**matlabtypesubstitute sensi = 0**

sensitivity order

**Default:** 0

---

**Note:** This property has custom functionality when its value is changed.

---

**matlabtypesubstitute nmaxevent = 10**

number of reported events

**Default:** 10

**matlabtypesubstitute ordering = 0**

reordering of states

**Default:** 0

**matlabtypesubstitute ss = 0**

steady state sensitivity flag

**Default:** 0

**matlabtypesubstitute x0 = double.empty("")**

custom initial state

**Default:** double.empty("")

**matlabtypesubstitute sx0 = double.empty("")**

custom initial sensitivity

**Default:** double.empty("")

**matlabtypesubstitute newton\_maxsteps = 40**

newton solver: maximum newton steps

**Default:** 40

---

**Note:** This property has custom functionality when its value is changed.

---

**matlabtypesubstitute z2event = double.empty("")**

mapping of event outputs to events

**Default:** double.empty("")

**matlabtypesubstitute pscale = "[]"**

parameter scaling Single value or vector matching sens\_ind. Valid options are "log", "log10" and "lin" for log, log10 or unscaled parameters p. Use [] for default as specified in the model (fallback: lin).

**Default:** "[]"

---

**Note:** This property has custom functionality when its value is changed.

---

**matlabtypesubstitute steadyStateSensitivityMode = 0**

Mode for computing sensitivities ({0: Newton}, 1: Simulation)

**Default:** 0



## Public Static Functions

```
mlhsInnerSubst< matlabtypesubstitute > amioption::
getIntegerPScale(matlabtypesubstitute pscaleString)
```

pscaleInt converts a parameter scaling string into the corresponding integer representation

```
param pscaleString parameter scaling string
retval pscaleString int
```

## Class amised

- Defined in file\_matlab\_@amised\_amised.m

## Inheritance Relationships

### Base Type

- public handle

## Class Documentation

**amised : public handle**

AMISED is a container for SED-ML objects.

## Public Functions

```
amised::amised(matlabtypesubstitute sedname)
```

amised reads in an SEDML document using the JAVA binding of libSEDML

```
param sedname name/path of the SEDML document
```

## Public Members

```
matlabtypesubstitute model = struct("'event',[],'sym',[]")
```

amimodel from the specified model

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---



---

**Note:** Matlab documentation of property attributes. **Default:** struct("'event',[],'sym',[]")

---

**matlabtypesubstitute modelname** = {""}

cell array of model identifiers

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** {""}

---

**matlabtypesubstitute sedml** = struct.empty("")

stores the struct tree from the xml definition

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** struct.empty("")

---

**matlabtypesubstitute outputcount** = "[]"

count the number of outputs per model

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** ""

---

**matlabtypesubstitute varidx** = "[]"

indexes for dataGenerators

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** ""

---

**matlabtypesubstitute varsym** = sym("[]")

symbolic expressions for variables

---

**Note:** This property has non-standard access specifiers: SetAccess = Private, GetAccess = Public

---

---

**Note:** [Matlab documentation of property attributes](#). **Default:** `sym("[]")`

---

**matlabtypesubstitute datasym** = `sym("[]")`

symbolic expressions for data

---

**Note:** This property has non-standard access specifiers: `SetAccess = Private`, `GetAccess = Public`

---



---

**Note:** [Matlab documentation of property attributes](#). **Default:** `sym("[]")`

---

## Class optsym

- Defined in file\_matlab\_@optsym\_optsym.m

## Inheritance Relationships

### Base Type

- public sym

## Class Documentation

### optsym : public sym

OPTSYM is an auxiliary class to gain access to the private symbolic property `s` which is necessary to be able to call `symobj::optimize` on it.

### Public Functions

**optsym::optsym(::sym symbol)**

optsym converts the symbolic object into a optsym object

**param symbol** symbolic object

**mlhsInnerSubst<::sym > optsym::getoptimized()**

getoptimized calls `symobj::optimize` on the optsym object

**retval out** optimized symbolic object

## Functions

### Function `am_and`

- Defined in `file_matlab_symbolic_am_and.m`

#### Function Documentation

**mlhsInnerSubst**< **matlabtypesubstitute** > **am\_and**(::sym a, ::sym b)

`am_and` is the amici implementation of the symbolic and function

**param a** first input parameter

**param b** second input parameter

**retval fun** logical value, negative for false, positive for true

### Function `am_eq`

- Defined in `file_matlab_symbolic_am_eq.m`

#### Function Documentation

**mlhsInnerSubst**< **matlabtypesubstitute** > **am\_eq**(**matlabtypesubstitute** varargin)

`am_eq` is currently a placeholder that simply produces an error message

**param varargin** elements for chain of equalities

**retval fun** logical value, negative for false, positive for true

### Function `am_ge`

- Defined in `file_matlab_symbolic_am_ge.m`

#### Function Documentation

**mlhsInnerSubst**< **matlabtypesubstitute** > **am\_ge**(::sym varargin)

`am_ge` is the amici implementation of the n-ary mathml greaterorequal function this is an n-ary function, for more than 2 input parameters it will check whether `and(varargin{1} >= varargin{2}, varargin{2} >= varargin{3}, ...)`

**param varargin** chain of input parameters

**retval fun** `a >= b` logical value, negative for false, positive for true

## Function am\_gt

- Defined in file\_matlab\_symbolic\_am\_gt.m

## Function Documentation

**mlhsInnerSubst< matlabtypesubstitute > am\_gt(::sym varargin)**

am\_gt is the amici implementation of the n-ary mathml greaterthan function this is an n-ary function, for more than 2 input parameters it will check whether and(varargin{1} > varargin{2}, varargin{2} > varargin{3},...)

**param varargin** chain of input parameters

**retval fun** a > b logical value, negative for false, positive for true

## Function am\_if

- Defined in file\_matlab\_symbolic\_am\_if.m

## Function Documentation

**mlhsInnerSubst< matlabtypesubstitute > am\_if(::sym condition, ::sym truepart, ::sym falsepart)**

am\_if is the amici implementation of the symbolic if function

**param condition** logical value

**param truepart** value if condition is true

**param falsepart** value if condition is false

**retval fun** if condition is true truepart, else falsepart

## Function am\_le

- Defined in file\_matlab\_symbolic\_am\_le.m

## Function Documentation

**mlhsInnerSubst< matlabtypesubstitute > am\_le(::sym varargin)**

am\_le is the amici implementation of the n-ary mathml lessorequal function this is an n-ary function, for more than 2 input parameters it will check whether and(varargin{1} <= varargin{2}, varargin{2} <= varargin{3},...)

**param varargin** chain of input parameters

**retval fun** a <= b logical value, negative for false, positive for true

## Function am\_lt

- Defined in file\_matlab\_symbolic\_am\_lt.m

## Function Documentation

**mlhsInnerSubst< matlabtypesubstitute > am\_lt(::sym varargin)**

am\_lt is the amici implementation of the n-ary mathml lessthan function this is an n-ary function, for more than 2 input parameters it will check whether and(varargin{1} < varargin{2}, varargin{2} < varargin{3},...)

**param varargin** chain of input parameters

**retval fun** a < b logical value, negative for false, positive for true

## Function am\_max

- Defined in file\_matlab\_symbolic\_am\_max.m

## Function Documentation

**mlhsInnerSubst< matlabtypesubstitute > am\_max(::sym a, ::sym b)**

am\_max is the amici implementation of the symbolic max function

**param a** first input parameter

**param b** second input parameter

**retval fun** maximum of a and b

## Function am\_min

- Defined in file\_matlab\_symbolic\_am\_min.m

## Function Documentation

**mlhsInnerSubst< matlabtypesubstitute > am\_min(::sym a, ::sym b)**

am\_min is the amici implementation of the symbolic min function

**param a** first input parameter

**param b** second input parameter

**retval fun** minimum of a and b

## Function `am_or`

- Defined in `file_matlab_symbolic_am_or.m`

### Function Documentation

`mlhsInnerSubst< matlabtypesubstitute > am_or(::sym a, ::sym b)`

`am_or` is the amici implementation of the symbolic or function

**param a** first input parameter

**param b** second input parameter

**retval fun** logical value, negative for false, positive for true

## Function `am_piecewise`

- Defined in `file_matlab_symbolic_am_piecewise.m`

### Function Documentation

`mlhsInnerSubst< matlabtypesubstitute > am_piecewise(matlabtypesubstitute piece, matlabtypesubstitute condition, matlabtypesubstitute default)`

`am_piecewise` is the amici implementation of the mathml piecewise function

**param piece** value if condition is true

**param condition** logical value

**param default** value if condition is false

**retval fun** return value, piece if condition is true, default if not

## Function `am_spline`

- Defined in `file_matlab_symbolic_am_spline.m`

### Function Documentation

`mlhsInnerSubst< matlabtypesubstitute > am_spline(matlabtypesubstitute varargin)`

## Function `am_spline_pos`

- Defined in file `_matlab_symbolic_am_spline_pos.m`

## Function Documentation

`mlhsInnerSubst< matlabtypesubstitute > am_spline_pos(matlabtypesubstitute varargin)`

## Function `am_stepfun`

- Defined in file `_matlab_symbolic_am_stepfun.m`

## Function Documentation

`mlhsInnerSubst< matlabtypesubstitute > am_stepfun(::sym t, matlabtypesubstitute tstart, matlabtypesubstitute vstart, matlabtypesubstitute tend, matlabtypesubstitute vend)`

`am_stepfun` is the amici implementation of the step function

**param t** input variable

**param tstart** input variable value at which the step starts

**param vstart** value during the step

**param tend** input variable value at which the step end

**param vend** value after the step

**retval fun** 0 before tstart, vstart between tstart and tend and vend after tend

## Function `am_xor`

- Defined in file `_matlab_symbolic_am_xor.m`

## Function Documentation

`mlhsInnerSubst< matlabtypesubstitute > am_xor(::sym a, ::sym b)`

`am_xor` is the amici implementation of the symbolic exclusive or function

**param a** first input parameter

**param b** second input parameter

**retval fun** logical value, negative for false, positive for true



## Function AMICI2D2D

- Defined in file\_matlab\_AMICI2D2D.m

## Function Documentation

**noret::substitute AMICI2D2D(matlabtypesubstitute filename,  
matlabtypesubstitute modelname)**

## Function amiwrap

- Defined in file\_matlab\_amiwrap.m

## Function Documentation

**noret::substitute amiwrap(matlabtypesubstitute varargin)**

AMIWRAP generates c++ mex files for the simulation of systems of differential equations via CVODES and IDAS.

**param varargin**

```
amiwrap ( modelname, symfun, tdir, o2flag )
```

*Required Parameters for varargin:*

- modelname specifies the name of the model which will be later used for the naming of the simulation file
- symfun specifies a function which executes model definition
- tdir target directory where the simulation file should be placed **Default:** \$AMICI-CIDIR/models/modelname
- o2flag boolean whether second order sensitivities should be enabled **Default:** false

**retval o2flag** void

## Function installAMICI

- Defined in file\_matlab\_installAMICI.m

## Function Documentation

`noret::substitute installAMICI()`

## Function SBML2AMICI

- Defined in file\_matlab\_SBML2AMICI.m

## Function Documentation

`noret::substitute SBML2AMICI(matlabtypesubstitute filename,  
matlabtypesubstitute modelname)`

SBML2AMICI generates AMICI model definition files from SBML.

**param filename** name of the SBML file (withouth extension)

**param modelname** name of the model, this will define the name of the output file (default: input filename)

**retval modelname** void

## 12.5 AMICI developer's guide

This document contains information for AMICI developers, not too relevant to regular users.

### 12.5.1 Branches / releases

AMICI roughly follows the [GitFlow](#). All new contributions are merged into `develop`. These changes are regularly merged into `master` as new releases. For release versioning we are trying to follow [semantic versioning](#). New releases are created on GitHub and are automatically deployed to [Zenodo](#) for archiving and to obtain a digital object identifier (DOI) to make them citable. Furthermore, our [CI pipeline](#) will automatically create and deploy a new release on [PyPI](#).

We try to keep a clean git history. Therefore, feature pull requests are squash-merged to `develop`. Merging of release branches to master is done via merge commits.

### 12.5.2 When starting to work on some issue

When starting to work on some Github issue, please assign yourself to let other developers know that you are working on it to avoid duplicate work. If the respective issue is not completely clear, it is generally a good idea to ask for clarification before starting to work on it.

If you want to work on something new, please create a Github issue first.

### 12.5.3 Code contributions

When making code contributions, please follow our style guide and the process described below:

- Check if you agree to release your contribution under the conditions provided in LICENSE. By opening a pull requests you confirm us that you do agree.
- Start a new branch from `develop` (on your fork, or at the main repository if you have access)
- Implement your changes
- Submit a pull request to the `develop` branch
- Ensure all tests pass
- When adding new functionality, please also provide test cases (see `tests/cpp/`, `python/tests/`, and `documentation/CI.md`)
- Write meaningful commit messages
- Run all tests to ensure nothing was broken ([more details](#))
  - Run `scripts/buildAll.sh && scripts/run-cpp-tests.sh`.
  - If you made changes to the Matlab or C++ code and have a Matlab license, please also run `tests/cpp/wrapTestModels.m` and `tests/testModels.m`
  - If you made changes to the Python or C++ code, run `make python-tests` in `build`
- When all tests are passing and you think your code is ready to merge, request a code review (see also our [code review guideline](#))
- Wait for feedback. If you do not receive feedback to your pull request within a week, please give us a friendly reminder.

### Style/compatibility guide

#### General

- All files and functions should come with file-level and function-level documentation.
- All new functionality should be covered by unit or integration tests. Runtime of those tests should be kept as short as possible.

#### Python

- In terms of Python compatibility, we follow numpy's [NEP 29](#).
- For the Python code we want to follow [PEP8](#). Although this is not the case for all existing code, any new contributions should do so. We use [black](#) for code formatting.  
 To run black as pre-commit hook, install the [pre-commit](#) package (e.g. `pip install pre-commit`), and enable AMICI-hooks by running `pre-commit install` from within the AMICI directory.
- We use Python [type hints](#) for all functions (but not for class attributes, since they are not supported by the current Python doxygen filter). In Python code, type hints should be used instead of doxygen `@type`.

For function docstrings, follow this format:

```
"""One-line description.

Possible a more detailed description

Arguments:
    Argument1: This needs to start on the same line, otherwise the current
               doxygen filter will fail.

Returns:
    Return value

Raises:
    SomeError in case of some error.
"""
```

## C++

- We use C++17
- We want to maintain compatibility with g++, clang, and the Intel C++ compiler
- For code formatting, we use clang-format and cmake-format. They can be invoked by `make clang-format cmake-format` from the CMake build directory.

## Matlab

*To be defined*

### 12.5.4 Further topics

#### AMICI documentation

This file describes how the AMICI documentation is organized and compiled.

#### Building documentation

##### Multi-interface documentation

AMICI documentation hosted at [Read the Docs \(RTD\)](#) is generated using [Sphinx](#) and related packages. The legacy GitHub Pages URL <https://amici-dev.github.io/AMICI/> is set up as a redirect to RTD.

The main configuration file is `documentation/conf.py` and the documentation is generated using `scripts/run-sphinx.sh`. The documentation is written to `documentation/_build/`.

The documentation comprises:

- reStructuredText / Markdown files from `documentation/`
- Python API documentation of native Python modules
- Python API documentation of Python generated via SWIG (doxygen-style comments translated to docstrings by SWIG)

- C++ API documentation (doxygen -> exhale -> breathe -> sphinx)
- Matlab API documentation (mtocpp -> doxygen -> exhale -> breathe -> sphinx)

### Doxygen-only (legacy)

(Parts of the) AMICI documentation can also be directly created using [doxygen](#) directly. It combines Markdown files from the root directory, from `documentation/` and in-source documentation from the C++ and Matlab source files.

The documentation is generated by running

```
scripts/run-doxygen.sh
```

The resulting HTML and PDF documentation will be created in `doc/`. `scripts/run-doxygen.sh` also checks for any missing in-source documentation.

### Doxygen configuration

The main doxygen configuration file is located in `matlab/mtoc/config/Doxyfile.template`. Edit this file for inclusion or exclusion of additional files.

### Matlab documentation

Matlab documentation is processed by `mtoc++`. This is configured in `matlab/mtoc/config`.

### Python documentation

Python documentation is processed by doxygen and doxypy using the script and filters in `scripts/`.

### Writing documentation

#### Out-of-source documentation

Out-of-source documentation files should be written in reStructuredText if intended for Read the Docs or in Markdown if intended for rendering on GitHub. Files to be included in the Sphinx/RTD documentation live in `documentation/`. Graphics for documentation are kept in `documentation/gfx/`.

#### When using Markdown

- Note that there are some incompatibilities of GitHub Markdown and Doxygen Markdown. Ideally documentation should be written in a format compatible with both. This affects for example images links which currently cause trouble in Doxygen.
- Where possible, relative links are preferred over absolute links. However, they should work with both Github and Doxygen and ideally with local files for offline use.
- Please stick to the limit of 80 characters per line for readability of raw Markdown files where possible.

However, note that some Markdown interpreters can handle line breaks within links and headings, whereas others cannot. Here, compatibility is preferred over linebreaks.

- Avoid trailing whitespace

## Maintaining the list of publications

We want to maintain a list of publications / projects using AMICI. This is located at `documentation/references.md`. This file is created by `documentation/recreate_reference_list.py` based on the bibtex file `documentation/amici_refs.bib`.

After any changes to `documentation/amici_refs.bib`, please run

```
documentation/recreate_reference_list.py
```

(requires `biblib`)

## Code review guide

A guide for reviewing code and having your code reviewed by others.

### Everyone

- Don't be too protective of your code
- Accept that, to a large extent, coding decisions are a matter of personal preference
- Don't get personal
- Ask for clarification
- Avoid strong language
- Try to understand your counterpart's perspective
- Clarify how strong you feel about each discussion point

## Reviewing code

- If there are no objective advantages, don't force your style on others
- Ask questions instead of making demands
- Assume the author gave his best
- Mind the scope (many things are nice to have, but might be out of scope of the current change - open a new issue)
- The goal is "good enough", not "perfect"
- Be constructive
- You do not always have to request changes

## Having your code reviewed

- Don't take it personal - the review is on the code, not on you
- Code reviews take time, appreciate the reviewer's comments
- Assume the reviewer did his best (but might still be wrong)
- Keep code changes small (e.g. separate wide reformatting from actual code changes to facility review)
- If the reviewer does not understand your code, probably many others won't either

## Checklist

- ☐ Adherence to project-specific style guide
- ☐ The code is self-explanatory
- ☐ The code is concise / expressive
- ☐ Meaningful identifiers are used
- ☐ Corner-cases are covered, cases not covered fail loudly
- ☐ The code can be expected to scale well (enough)
- ☐ The code is well documented (e.g., input, operation, output), but without trivial comments
- ☐ The code is **SOLID**
- ☐ New code is added in the most meaningful place (i.e. matches the current architecture)
- ☐ No magic numbers
- ☐ No hard-coded values that should be user inputs
- ☐ No dead code left
- ☐ The changes make sense
- ☐ The changes are not obviously degrading performance
- ☐ There is no duplicated code
- ☐ The API is convenient
- ☐ Code block length and complexity is adequate
- ☐ Spelling okay
- ☐ The code is tested

## Continuous integration (CI) and tests

AMICI uses a continuous integration pipeline running via <https://github.com/features/actions>. This includes the following steps:

- Checking existence and format of documentation
- Static code analysis (<http://cppcheck.sourceforge.net/>)
- Unit and integration tests
- Memory leak detection

More details are provided in the sections below.

The CI scripts and tests can be found in `tests/` and `scripts/`. Some of the tests are integrated with CMake, see `make help` in the build directory.

### C++ unit and integration tests

To run C++ tests, build AMICI with `make` or `scripts/buildAll.sh`, then run `scripts/run-cpp-tests.sh`.

### Python unit and integration tests

To run Python tests, run `../scripts/run-python-tests.sh` from anywhere (assumes build directory is `build/`) or run `make python-tests` in your build directory.

### SBML Test Suite

We test Python-AMICI SBML support using the test cases from the semantic [SBML Test Suite](#). When making changes to the model import functions, make sure to run these tests.

To run the SBML Test Suite test cases, the easiest way is:

1. Running `scripts/installAmiciSource.sh` which creates a virtual Python environment and performs a development installation of AMICI from the current repository. (This needs to be run only once or after AMICI model generation or C++ changes).
2. Running `scripts/run-SBMLTestsuite.sh`. This will download the test cases if necessary and run them all. A subset of test cases can be selected with an optional argument (e.g. `scripts/run-SBMLTestsuite.sh 1, 3-6, 8`, to run cases 1, 3, 4, 5, 6 and 8).

Once the test cases are available locally, for debugging it might be easier to directly use `pytest` with `tests/testSBMLSuite.py`.

### Matlab tests (not included in CI pipeline)

To execute the Matlab test suite, run `tests/testModels.m`.

### Model simulation integration tests

Many of our integration tests are model simulations. The simulation results obtained from the Python and C++ are compared to results saved in an HDF5 file (`tests/cpp/expectedResults.h5`). Settings and data for the test simulations are also specified in this file.

**Note:** The C++ code for the models is included in the repository under `models/`. This code is to be updated whenever `amici::Model` changes.



## Regenerating C++ code of the test models

Regeneration of the model code has to be done whenever `amici::Model` or the Matlab model import routines change.


This is done with

```
tests/cpp/wrapTestModels.m
```

**Note:** This is currently only possible from Matlab < R2018a. This should change as soon as 1) all second-order sensitivity code is ported to C++/Python, 2) a non-SBML import exists for Python and 3) support for events has been added for Python.

## Regenerating expected results

To update test results, run `make test` in the build directory, replace `tests/cpp/expectedResults.h5` by `tests/cpp/writeResults.h5.bak` [ONLY DO THIS AFTER TRIPLE CHECKING CORRECTNESS OF RESULTS] Before replacing the test results, confirm that only expected datasets have changed, e.g. using

```
h5diff -v --relative 1e-8 tests/cpp/expectedResults.h5 tests/cpp/writeResults.h5.bak |  less
```

## Adding/Updating tests

To add new tests add a new corresponding python script (see, e.g., `./tests/generateTestConfig/example_dirac.py`) and add it to and run `tests/generateTestConfigurationForExamples.sh`. Then regenerate the expected test results (see above).

## Debugging AMICI

This document contains some information on how to debug any issues in AMICI, in particular for C++ Python extensions.

### Caveman debugging / printf-debugging

The simplest approach may often be adding some print-statements to the code, as this does not require any special tools.

Note that after each change of the C++ files, the AMICI extension *as well as the model extension* (if any model functions are called), need to be recompiled. The simplest and safest approach would be re-installation of the amici package and re-import of the model. As this can be very time-consuming, the following shortcut is possible, assuming you are using a development installation (`pip install -e .`):

```
# rebuild the amici base extension, from within the amici root directory
# (note that this only recompiles the amici source files, NOT third-party
# dependencies such as sundials):
cd python/sdist/
python setup.py build_ext --build-lib .

# rebuild the model, from within the model package directory:
python setup.py build_ext --force --build-lib .
```

Note: Be careful when working interactively, Python may not pick up any changes in already imported modules. The safest is to start a new Python process after any changes.

### Using a proper debugger

Debugging with with [gdb](<https://www.sourceware.org/gdb/>) is most convenient with a minimal reproducible example that is directly invoked from gdb. For example:

```
# start gdb
gdb --args python -m pytest ../tests/test_sbml_import.py::test_nosensi

# inside gdb, set a meaningful breakpoint and launch:
break amici::runAmiciSimulation
run
# ... (see one of the many gdb tutorials)
```

Alternative, gdb can attach to a running process by passing the `--pid` argument.

For many users, it may be more convenient to use gdb via some graphical user interface as provided by various C++ IDEs.

---

**Note:** For better debugging experience, but at the cost of runtime performance, consider building the amici and model extension with environment variable `ENABLE_AMICI_DEBUGGING=TRUE`. This will include debugging symbols and disable compiler optimizations.

---

## 12.6 Handling of Discontinuities

This document provides guidance and rationale on the implementation of events in AMICI. Events include any discontinuities in the right hand side of the differential equation. There are three types of discontinuities:

- **Solution Jump Discontinuities** can be created by SBML events or delta functions in the right hand side.
- **Right-Hand-Side Jump Discontinuities** result in removable discontinuities in the solution and can be created by Piecewise, Heaviside functions and other logical operators in the right hand side.
- **Right-Hand-Side Removable Discontinuities** do not lead to discontinuities in the solution, but may lead to discontinuous higher order temporal derivatives and can be created by functions such as max or min in the right hand side.

### 12.6.1 Mathematical Considerations

A detailed mathematical description of the required sensitivity formulas is provided in

- Fröhlich, F., Theis, F. J., Rädler, J. O., & Hasenauer, J. (2017). Parameter estimation for dynamical systems with discrete events and logical operations. *Bioinformatics*, 33(7), 1049-1056. doi:[10.1093/bioinformatics/btw764](https://doi.org/10.1093/bioinformatics/btw764).

## 12.6.2 Algorithmic Considerations

### Solution Jump Discontinuities

SUNDIALS by itself does not support solution jump discontinuities. We implement support by accessing private SUNDIALS API in `amici::Solver::resetState()`, `amici::Solver::reInitPostProcess()` and `amici::Solver::reInitPostProcessB()`. These functions reset interval variables to initial values to simulate a fresh integration start, but keep/update the solution history, which is important for adjoint solutions.

### Right-Hand-Side Jump Discontinuities

In principle these discontinuities do not need any special treatment, but empirically, the solver may overstep or completely ignore the discontinuity, leading to poor solution quality. This is particularly problematic when step size is large and changes in step size, which can be caused by parameter changes, inclusion of forward sensitivities or during backward solves, may alter solutions in unexpected ways. Accordingly, finite difference approximations, forward sensitivities as well as adjoint sensitivities will yield poor derivative approximations.

To address these issues, we use the built-in rootfinding functionality in SUNDIALS, which pauses the solver at the locations of discontinuities and avoids overstepping or ignoring of discontinuities.

Another difficulty comes with the evaluation of Heaviside functions. After or during processing of discontinuities, Heaviside functions need to be evaluated at the left and right hand limit of discontinuities. This is challenging as the solver may slightly over- or understep the discontinuity timepoint by a small epsilon and limits have to be correctly computed in both forward and backward passes.

To address this issue, AMICI uses a vector of Heaviside helper variables  $h$  that keeps track of the values of the Heaviside functions that have the respective root function as argument. These will be automatically updated during events and take either 0 or 1 values as appropriate pre/post event limits.

In order to fully support SBML events and Piecewise functions, AMICI uses the SUNDIALS functionality to only track zero crossings from negative to positive. Accordingly, two root functions are necessary to keep track of Heaviside functions and two Heaviside function helper variables will be created, where one corresponds to the value of  $\text{Heaviside}(\dots)$  and one to the value of  $1 - \text{Heaviside}(\dots)$ . To ensure that Heaviside functions are correctly evaluated at the beginning of the simulation, Heaviside functions are implemented as unit steps that evaluate to 1 at 0. The arguments of Heaviside functions are normalized such that respective properties of Piecewise functions are conserved for the first Heaviside function variable. Accordingly, the value of the second helper variable is incorrect when simulation starts when the respective Heaviside function evaluates to zero at initialization and should generally not be used.

### Right-Hand-Side Removable Discontinuities

Removable discontinuities do not require any special treatment. Numerically, this may be advantageous, but is currently not implemented.



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### a

- `amici`, 264
- `amici.amici`, 266
- `amici.bngl_import`, 333
- `amici.conservated_quantities_demartino`, 387
- `amici.conservated_quantities_rref`, 388
- `amici.de_export`, 350
- `amici.de_model`, 362
- `amici.gradient_check`, 383
- `amici.import_utils`, 345
- `amici.logging`, 381
- `amici.numpy`, 389
- `amici.pandas`, 379
- `amici.parameter_mapping`, 384
- `amici.petab_import`, 334
- `amici.petab_import_pysb`, 338
- `amici.petab_objective`, 339
- `amici.petab_simulate`, 343
- `amici.plotting`, 377
- `amici.pysb_import`, 331
- `amici.sbml_import`, 326
- `amici.sbml_utils`, 391
- `amici.splines`, 396





## Symbols

- `_USE_MATH_DEFINES` (*C macro*), 679
- `__init__()` (*amici.AmiciVersionError method*), 264
- `__init__()` (*amici.ModelModule method*), 264
- `__init__()` (*amici.add\_path method*), 265
- `__init__()` (*amici.amici.CpuTimer method*), 268
- `__init__()` (*amici.amici.ExpData method*), 268
- `__init__()` (*amici.amici.LogItem method*), 279
- `__init__()` (*amici.amici.Model method*), 280
- `__init__()` (*amici.amici.ModelDimensions method*), 295
- `__init__()` (*amici.amici.ReturnData method*), 300
- `__init__()` (*amici.amici.SimulationParameters method*), 313
- `__init__()` (*amici.amici.Solver method*), 316
- `__init__()` (*amici.de\_export.DEExporter method*), 351
- `__init__()` (*amici.de\_export.DEModel method*), 353
- `__init__()` (*amici.de\_export.TemplateAmici method*), 360
- `__init__()` (*amici.de\_model.AlgebraicEquation method*), 363
- `__init__()` (*amici.de\_model.AlgebraicState method*), 364
- `__init__()` (*amici.de\_model.ConservationLaw method*), 365
- `__init__()` (*amici.de\_model.Constant method*), 366
- `__init__()` (*amici.de\_model.DifferentialState method*), 366
- `__init__()` (*amici.de\_model.Event method*), 368
- `__init__()` (*amici.de\_model.EventObservable method*), 369
- `__init__()` (*amici.de\_model.Expression method*), 370
- `__init__()` (*amici.de\_model.LogLikelihoodRZ method*), 370
- `__init__()` (*amici.de\_model.LogLikelihoodY method*), 371
- `__init__()` (*amici.de\_model.LogLikelihoodZ method*), 372
- `__init__()` (*amici.de\_model.ModelQuantity method*), 372
- `__init__()` (*amici.de\_model.Observable method*), 373
- `__init__()` (*amici.de\_model.Parameter method*), 374
- `__init__()` (*amici.de\_model.SigmaY method*), 374
- `__init__()` (*amici.de\_model.SigmaZ method*), 375
- `__init__()` (*amici.de\_model.State method*), 376
- `__init__()` (*amici.import\_utils.CircularDependencyError method*), 346
- `__init__()` (*amici.import\_utils.SBMLException method*), 346
- `__init__()` (*amici.numpy.ExpDataView method*), 389
- `__init__()` (*amici.numpy.ReturnDataView method*), 389
- `__init__()` (*amici.numpy.SwigPtrView method*), 390
- `__init__()` (*amici.parameter\_mapping.ParameterMapping method*), 384
- `__init__()` (*amici.parameter\_mapping.ParameterMappingForCondition method*), 385
- `__init__()` (*amici.petab\_simulate.PetabSimulator method*), 344
- `__init__()` (*amici.sbml\_import.SbmlImporter method*), 327
- `__init__()` (*amici.sbml\_utils.MathMLSbmlPrinter method*), 392
- `__init__()` (*amici.sbml\_utils.SbmlAnnotationError method*), 392
- `__init__()` (*amici.sbml\_utils.SbmlDuplicateComponentIdError method*), 392
- `__init__()` (*amici.sbml\_utils.SbmlInvalidIdSyntax method*), 392
- `__init__()` (*amici.sbml\_utils.SbmlMathError method*), 392
- `__init__()` (*amici.sbml\_utils.SbmlMissingComponentIdError method*), 393
- `__init__()` (*amici.splines.AbstractSpline method*), 396
- `__init__()` (*amici.splines.CubicHermiteSpline method*), 400
- `__init__()` (*amici.splines.UniformGrid method*), 404

## A

- `AbstractSpline` (*class in amici.splines*), 396
- `adams` (*amici.amici.LinearMultistepMethod attribute*), 279
- `add_assignment_rule()` (*in module amici.sbml\_utils*), 393

`add_compartment()` (in module `amici.sbml_utils`), 393  
`add_component()` (`amici.de_export.DEModel` method), 354  
`add_conservation_law()` (`amici.de_export.DEModel` method), 354  
`add_d_dt()` (`amici.sbml_import.SbmlImporter` method), 327  
`add_inflow()` (in module `amici.sbml_utils`), 393  
`add_local_symbol()` (`amici.sbml_import.SbmlImporter` method), 327  
`add_noise()` (`amici.petab_simulate.PetabSimulator` method), 344  
`add_parameter()` (in module `amici.sbml_utils`), 393  
`add_path` (class in `amici`), 265  
`add_rate_rule()` (in module `amici.sbml_utils`), 394  
`add_species()` (in module `amici.sbml_utils`), 394  
`add_to_sbml_model()` (`amici.splines.AbstractSpline` method), 397  
`add_to_sbml_model()` (`amici.splines.CubicHermiteSpline` method), 400  
`adjoint` (`amici.amici.SensitivityMethod` attribute), 313  
`aggregate_slh()` (in module `amici.petab_objective`), 339  
`algebraic_states()` (`amici.de_export.DEModel` method), 354  
`AlgebraicEquation` (class in `amici.de_model`), 363  
`AlgebraicState` (class in `amici.de_model`), 364  
`amici`  
    module, 264  
`amici.amici`  
    module, 266  
`amici.bngl_import`  
    module, 333  
`amici.conservd_quantities_demartino`  
    module, 387  
`amici.conservd_quantities_rref`  
    module, 388  
`amici.de_export`  
    module, 350  
`amici.de_model`  
    module, 362  
`amici.gradient_check`  
    module, 383  
`amici.import_utils`  
    module, 345  
`amici.logging`  
    module, 381  
`amici.numpy`  
    module, 389  
`amici.pandas`  
    module, 379  
`amici.parameter_mapping`  
    module, 384  
`amici.petab_import`  
    module, 334  
`amici.petab_import_pysb`  
    module, 338  
`amici.petab_objective`  
    module, 339  
`amici.petab_simulate`  
    module, 343  
`amici.plotting`  
    module, 377  
`amici.pysb_import`  
    module, 331  
`amici.sbml_import`  
    module, 326  
`amici.sbml_utils`  
    module, 391  
`amici.splines`  
    module, 396  
`amici::AbstractModel` (C++ class), 426  
`amici::AbstractModel::~~AbstractModel` (C++ function), 426  
`amici::AbstractModel::fcreate_splines` (C++ function), 445  
`amici::AbstractModel::fdeltaqB` (C++ function), 436  
`amici::AbstractModel::fdeltasx` (C++ function), 435  
`amici::AbstractModel::fdeltax` (C++ function), 435  
`amici::AbstractModel::fdeltaxB` (C++ function), 436  
`amici::AbstractModel::fdJrzdsigma` (C++ function), 440  
`amici::AbstractModel::fdJrzdz` (C++ function), 440  
`amici::AbstractModel::fdJydsigma` (C++ function), 439  
`amici::AbstractModel::fdJydy` (C++ function), 438  
`amici::AbstractModel::fdJydy_colptrs` (C++ function), 439  
`amici::AbstractModel::fdJydy_rowvals` (C++ function), 439  
`amici::AbstractModel::fdJzdsigma` (C++ function), 440  
`amici::AbstractModel::fdJzdz` (C++ function), 439  
`amici::AbstractModel::fdrzdp` (C++ function), 434  
`amici::AbstractModel::fdrzdx` (C++ function), 434  
`amici::AbstractModel::fdsigmaydp` (C++ function), 437  
`amici::AbstractModel::fdsigmaydy` (C++ function), 437  
`amici::AbstractModel::fdsigmazdp` (C++ function), 437

amici::AbstractModel::fdspline\_slopesdp (C++ function), 445  
 amici::AbstractModel::fdspline\_valuesdp (C++ function), 445  
 amici::AbstractModel::fdtotal\_cldp (C++ function), 444  
 amici::AbstractModel::fdtotal\_cldx\_rdata (C++ function), 444  
 amici::AbstractModel::fdtotal\_cldx\_rdata\_colptrs (C++ function), 444  
 amici::AbstractModel::fdtotal\_cldx\_rdata\_rowvals (C++ function), 444  
 amici::AbstractModel::fdwdp (C++ function), 441  
 amici::AbstractModel::fdwdp\_colptrs (C++ function), 441  
 amici::AbstractModel::fdwdp\_rowvals (C++ function), 441  
 amici::AbstractModel::fdwdw (C++ function), 442  
 amici::AbstractModel::fdwdw\_colptrs (C++ function), 443  
 amici::AbstractModel::fdwdw\_rowvals (C++ function), 443  
 amici::AbstractModel::fdwdx (C++ function), 442  
 amici::AbstractModel::fdwdx\_colptrs (C++ function), 442  
 amici::AbstractModel::fdwdx\_rowvals (C++ function), 442  
 amici::AbstractModel::fdx0 (C++ function), 430  
 amici::AbstractModel::fdx\_rdatadp (C++ function), 443  
 amici::AbstractModel::fdx\_rdatadtcl (C++ function), 443  
 amici::AbstractModel::fdx\_rdatadtcl\_colptrs (C++ function), 444  
 amici::AbstractModel::fdx\_rdatadtcl\_rowvals (C++ function), 444  
 amici::AbstractModel::fdx\_rdatadx\_solver (C++ function), 443  
 amici::AbstractModel::fdx\_rdatadx\_solver\_colptrs (C++ function), 443  
 amici::AbstractModel::fdx\_rdatadx\_solver\_rowvals (C++ function), 443  
 amici::AbstractModel::fdxdotdp (C++ function), 429  
 amici::AbstractModel::fdydp (C++ function), 431  
 amici::AbstractModel::fdydx (C++ function), 432  
 amici::AbstractModel::fdzdp (C++ function), 433  
 amici::AbstractModel::fdzdx (C++ function), 434  
 amici::AbstractModel::fJ (C++ function), 427  
 amici::AbstractModel::fJB (C++ function), 427  
 amici::AbstractModel::fJDiag (C++ function), 428  
 amici::AbstractModel::fJrz (C++ function), 438  
 amici::AbstractModel::fJSparse (C++ function), 428  
 amici::AbstractModel::fJSparseB (C++ function), 428  
 amici::AbstractModel::fJSparseB\_ss (C++ function), 427  
 amici::AbstractModel::fJv (C++ function), 429  
 amici::AbstractModel::fJy (C++ function), 438  
 amici::AbstractModel::fJz (C++ function), 438  
 amici::AbstractModel::froot (C++ function), 426  
 amici::AbstractModel::frz (C++ function), 433  
 amici::AbstractModel::fsigmay (C++ function), 436  
 amici::AbstractModel::fsigmaz (C++ function), 437  
 amici::AbstractModel::fsrz (C++ function), 433  
 amici::AbstractModel::fstau (C++ function), 430  
 amici::AbstractModel::fsx0 (C++ function), 430  
 amici::AbstractModel::fsx0\_fixedParameters (C++ function), 430  
 amici::AbstractModel::fsxdot (C++ function), 426  
 amici::AbstractModel::fsz (C++ function), 432  
 amici::AbstractModel::fw (C++ function), 440  
 amici::AbstractModel::fx0 (C++ function), 429  
 amici::AbstractModel::fx0\_fixedParameters (C++ function), 429  
 amici::AbstractModel::fxBdot\_ss (C++ function), 426  
 amici::AbstractModel::fxdot (C++ function), 426  
 amici::AbstractModel::fy (C++ function), 431  
 amici::AbstractModel::fz (C++ function), 432  
 amici::AbstractModel::getAmiciCommit (C++ function), 429  
 amici::AbstractModel::getAmiciVersion (C++ function), 429  
 amici::AbstractModel::getSolver (C++ function), 426  
 amici::AbstractModel::isFixedParameterStateReinitialization (C++ function), 429  
 amici::AbstractModel::writeSteadystateJB (C++ function), 427  
 amici::AbstractSpline (C++ class), 446  
 amici::AbstractSpline::~AbstractSpline (C++ function), 446  
 amici::AbstractSpline::AbstractSpline (C++ function), 446  
 amici::AbstractSpline::coefficients (C++ member), 449  
 amici::AbstractSpline::coefficients\_extrapolate (C++ member), 449  
 amici::AbstractSpline::coefficients\_extrapolate\_sensi (C++ member), 449  
 amici::AbstractSpline::coefficients\_sensi (C++ member), 449  
 amici::AbstractSpline::compute\_coefficients (C++ function), 446

amici::AbstractSpline::compute\_coefficients\_sensi 675  
 (C++ function), 446  
 amici::AbstractSpline::compute\_final\_sensitivity 675  
 (C++ function), 448  
 amici::AbstractSpline::compute\_final\_value 676  
 (C++ function), 448  
 amici::AbstractSpline::get\_equidistant\_spacing 676  
 (C++ function), 448  
 amici::AbstractSpline::get\_final\_sensitivity 677  
 (C++ function), 448  
 amici::AbstractSpline::get\_final\_sensitivity\_scaled 677  
 (C++ function), 448  
 amici::AbstractSpline::get\_final\_value (C++ 677  
 function), 448  
 amici::AbstractSpline::get\_final\_value\_scaled 678  
 (C++ function), 448  
 amici::AbstractSpline::get\_logarithmic\_parameterization 678  
 (C++ function), 448  
 amici::AbstractSpline::get\_node\_value (C++ 678  
 function), 447  
 amici::AbstractSpline::get\_node\_value\_scaled 450  
 (C++ function), 447  
 amici::AbstractSpline::get\_sensitivity (C++ 450  
 function), 447  
 amici::AbstractSpline::get\_sensitivity\_scaled 450  
 (C++ function), 447  
 amici::AbstractSpline::get\_value (C++ func- 451  
 tion), 447  
 amici::AbstractSpline::get\_value\_scaled 450  
 (C++ function), 447  
 amici::AbstractSpline::n\_nodes (C++ function), 448  
 amici::AbstractSpline::node\_values\_ (C++ 451  
 member), 449  
 amici::AbstractSpline::nodes\_ (C++ member), 449  
 amici::AbstractSpline::set\_final\_sensitivity\_scaled 452  
 (C++ function), 449  
 amici::AbstractSpline::set\_final\_value\_scaled 452  
 (C++ function), 449  
 amici::addSlice (C++ function), 645, 646  
 amici::AMICI\_CONV\_FAILURE (C++ member), 673  
 amici::AMICI\_DAMPING\_FACTOR\_ERROR (C++ mem- 674  
 ber), 674  
 amici::AMICI\_DATA\_RETURN (C++ member), 674  
 amici::amici\_daxpy (C++ function), 646  
 amici::amici\_dgemm (C++ function), 646  
 amici::amici\_dgemv (C++ function), 647  
 amici::AMICI\_ERR\_FAILURE (C++ member), 674  
 amici::AMICI\_ERROR (C++ member), 674  
 amici::AMICI\_FIRST\_RHSFUNC\_ERR (C++ member), 674  
 amici::AMICI\_ILL\_INPUT (C++ member), 675  
 amici::AMICI\_MAX\_TIME\_EXCEEDED (C++ member), 675  
 amici::AMICI\_NO\_STEADY\_STATE (C++ member), 675  
 amici::AMICI\_NORMAL (C++ member), 675  
 amici::AMICI\_NOT\_IMPLEMENTED (C++ member), 676  
 amici::AMICI\_NOT\_RUN (C++ member), 676  
 amici::AMICI\_ONE\_STEP (C++ member), 676  
 amici::AMICI\_ONEOUTPUT (C++ member), 676  
 amici::AMICI\_PREEQUILIBRATE (C++ member), 676  
 amici::AMICI\_RECOVERABLE\_ERROR (C++ member), 677  
 amici::AMICI\_RHSFUNC\_FAIL (C++ member), 677  
 amici::AMICI\_ROOT\_RETURN (C++ member), 677  
 amici::AMICI\_SINGULAR\_JACOBIAN (C++ member), 677  
 amici::AMICI\_SUCCESS (C++ member), 678  
 amici::AMICI\_TOO\_MUCH\_ACC (C++ member), 678  
 amici::AMICI\_TOO\_MUCH\_WORK (C++ member), 678  
 amici::AMICI\_UNRECOVERABLE\_ERROR (C++ mem- 678  
 ber), 678  
 amici::AmiException (C++ class), 450  
 amici::AmiException::AmiException (C++ func- 450  
 tion), 450  
 amici::AmiException::getBacktrace (C++ func- 450  
 tion), 450  
 amici::AmiException::storeBacktrace (C++ 450  
 function), 450  
 amici::AmiException::storeMessage (C++ func- 451  
 tion), 451  
 amici::AmiException::what (C++ function), 450  
 amici::AmiVector (C++ class), 451  
 amici::AmiVector::~~AmiVector (C++ function), 452  
 amici::AmiVector::abs (C++ function), 453  
 amici::AmiVector::AmiVector (C++ function), 451  
 amici::AmiVector::at (C++ function), 453  
 amici::AmiVector::begin (C++ function), 452  
 amici::AmiVector::copy (C++ function), 453  
 amici::AmiVector::data (C++ function), 452  
 amici::AmiVector::end (C++ function), 452  
 amici::AmiVector::getLength (C++ function), 452  
 amici::AmiVector::getNVector (C++ function), 452  
 amici::AmiVector::getVector (C++ function), 452  
 amici::AmiVector::minus (C++ function), 453  
 amici::AmiVector::operator\*= (C++ function), 452  
 amici::AmiVector::operator/= (C++ function), 452  
 amici::AmiVector::operator= (C++ function), 452  
 amici::AmiVector::operator[] (C++ function), 453  
 amici::AmiVector::set (C++ function), 453  
 amici::AmiVector::zero (C++ function), 453  
 amici::AmiVectorArray (C++ class), 453  
 amici::AmiVectorArray::~~AmiVectorArray (C++ 454  
 function), 454  
 amici::AmiVectorArray::AmiVectorArray (C++ 454  
 function), 454  
 amici::AmiVectorArray::at (C++ function), 454

amici::AmiVectorArray::copy (C++ function), 455  
 amici::AmiVectorArray::data (C++ function), 454  
 amici::AmiVectorArray::flatten\_to\_vector (C++ function), 455  
 amici::AmiVectorArray::getLength (C++ function), 455  
 amici::AmiVectorArray::getNVector (C++ function), 455  
 amici::AmiVectorArray::getNVectorArray (C++ function), 455  
 amici::AmiVectorArray::operator= (C++ function), 454  
 amici::AmiVectorArray::operator[] (C++ function), 455  
 amici::AmiVectorArray::zero (C++ function), 455  
 amici::backtraceString (C++ function), 648  
 amici::BackwardProblem (C++ class), 456  
 amici::BackwardProblem::BackwardProblem (C++ function), 456  
 amici::BackwardProblem::getAdjointQuadrature (C++ function), 456  
 amici::BackwardProblem::getAdjointState (C++ function), 456  
 amici::BackwardProblem::getdJydx (C++ function), 456  
 amici::BackwardProblem::gett (C++ function), 456  
 amici::BackwardProblem::getwhich (C++ function), 456  
 amici::BackwardProblem::getwhichptr (C++ function), 456  
 amici::BackwardProblem::workBackwardProblem (C++ function), 456  
 amici::BLASLayout (C++ enum), 632  
 amici::BLASLayout::colMajor (C++ enumerator), 632  
 amici::BLASLayout::rowMajor (C++ enumerator), 632  
 amici::BLASTranspose (C++ enum), 633  
 amici::BLASTranspose::conjTrans (C++ enumerator), 633  
 amici::BLASTranspose::noTrans (C++ enumerator), 633  
 amici::BLASTranspose::trans (C++ enumerator), 633  
 amici::checkBufferSize (C++ function), 648  
 amici::checkSigmaPositivity (C++ function), 648, 649  
 amici::ConditionContext (C++ class), 457  
 amici::ConditionContext::~ConditionContext (C++ function), 457  
 amici::ConditionContext::applyCondition (C++ function), 457  
 amici::ConditionContext::ConditionContext (C++ function), 457  
 amici::ConditionContext::operator= (C++ function), 457  
 amici::ConditionContext::restore (C++ function), 457  
 amici::const\_N\_Vector (C++ type), 683  
 amici::ContextManager (C++ class), 458  
 amici::ContextManager::ContextManager (C++ function), 458  
 amici::CpuTimer (C++ class), 458  
 amici::CpuTimer::CpuTimer (C++ function), 458  
 amici::CpuTimer::elapsed\_milliseconds (C++ function), 459  
 amici::CpuTimer::elapsed\_seconds (C++ function), 458  
 amici::CpuTimer::reset (C++ function), 458  
 amici::CpuTimer::uses\_thread\_clock (C++ member), 459  
 amici::CvodeException (C++ class), 459  
 amici::CvodeException::CvodeException (C++ function), 459  
 amici::CvodeSolver (C++ class), 460  
 amici::CvodeSolver::~CvodeSolver (C++ function), 460  
 amici::CvodeSolver::adjInit (C++ function), 465  
 amici::CvodeSolver::allocateSolver (C++ function), 464  
 amici::CvodeSolver::allocateSolverB (C++ function), 465  
 amici::CvodeSolver::binit (C++ function), 468  
 amici::CvodeSolver::boost::serialization::serialize (C++ function), 469  
 amici::CvodeSolver::calcIC (C++ function), 463  
 amici::CvodeSolver::calcICB (C++ function), 463  
 amici::CvodeSolver::clone (C++ function), 460  
 amici::CvodeSolver::diag (C++ function), 466  
 amici::CvodeSolver::diagB (C++ function), 466  
 amici::CvodeSolver::getAdjBmem (C++ function), 467  
 amici::CvodeSolver::getB (C++ function), 463  
 amici::CvodeSolver::getDky (C++ function), 461  
 amici::CvodeSolver::getDkyB (C++ function), 462  
 amici::CvodeSolver::getLastOrder (C++ function), 467  
 amici::CvodeSolver::getModel (C++ function), 462  
 amici::CvodeSolver::getNumErrTestFails (C++ function), 467  
 amici::CvodeSolver::getNumNonlinSolvConvFails (C++ function), 467  
 amici::CvodeSolver::getNumRhsEvals (C++ function), 467  
 amici::CvodeSolver::getNumSteps (C++ function), 466  
 amici::CvodeSolver::getQuad (C++ function), 463  
 amici::CvodeSolver::getQuadB (C++ function), 463



<code>amici::CNodeSolver::getQuadDky (C++ function),</code>	<code>amici::CNodeSolver::setLinearSolver (C++</code>
<code>463</code>	<code>function), 462</code>
<code>amici::CNodeSolver::getQuadDkyB (C++ function),</code>	<code>amici::CNodeSolver::setLinearSolverB (C++</code>
<code>461</code>	<code>function), 462</code>
<code>amici::CNodeSolver::getRootInfo (C++ function),</code>	<code>amici::CNodeSolver::setMaxNumSteps (C++ func-</code>
<code>462</code>	<code>tion), 464</code>
<code>amici::CNodeSolver::getSens (C++ function), 463</code>	<code>amici::CNodeSolver::setMaxNumStepsB (C++</code>
<code>amici::CNodeSolver::getSensDky (C++ function),</code>	<code>function), 466</code>
<code>461</code>	<code>amici::CNodeSolver::setNonLinearSolver (C++</code>
<code>amici::CNodeSolver::init (C++ function), 467</code>	<code>function), 462</code>
<code>amici::CNodeSolver::initSteadystate (C++</code>	<code>amici::CNodeSolver::setNonLinearSolverB</code>
<code>function), 468</code>	<code>(C++ function), 462</code>
<code>amici::CNodeSolver::operator== (C++ function),</code>	<code>amici::CNodeSolver::setNonLinearSolverSens</code>
<code>469</code>	<code>(C++ function), 462</code>
<code>amici::CNodeSolver::qbinit (C++ function), 468</code>	<code>amici::CNodeSolver::setQuadErrCon (C++ func-</code>
<code>amici::CNodeSolver::quadInit (C++ function), 465</code>	<code>tion), 464</code>
<code>amici::CNodeSolver::quadReInitB (C++ function),</code>	<code>amici::CNodeSolver::setQuadErrConB (C++ func-</code>
<code>461</code>	<code>tion), 464</code>
<code>amici::CNodeSolver::quadSStolerances (C++</code>	<code>amici::CNodeSolver::setSensErrCon (C++ func-</code>
<code>function), 466</code>	<code>tion), 464</code>
<code>amici::CNodeSolver::quadSStolerancesB (C++</code>	<code>amici::CNodeSolver::setSensParams (C++ func-</code>
<code>function), 466</code>	<code>tion), 465</code>
<code>amici::CNodeSolver::reInit (C++ function), 460</code>	<code>amici::CNodeSolver::setSensSStolerances</code>
<code>amici::CNodeSolver::reInitB (C++ function), 460</code>	<code>(C++ function), 464</code>
<code>amici::CNodeSolver::reInitPostProcess (C++</code>	<code>amici::CNodeSolver::setSparseJacFn (C++ func-</code>
<code>function), 463</code>	<code>tion), 468</code>
<code>amici::CNodeSolver::reInitPostProcessB (C++</code>	<code>amici::CNodeSolver::setSparseJacFn_ss (C++</code>
<code>function), 463</code>	<code>function), 469</code>
<code>amici::CNodeSolver::reInitPostProcessF (C++</code>	<code>amici::CNodeSolver::setSparseJacFnB (C++</code>
<code>function), 463</code>	<code>function), 469</code>
<code>amici::CNodeSolver::resetState (C++ function),</code>	<code>amici::CNodeSolver::setSStolerances (C++</code>
<code>465</code>	<code>function), 464</code>
<code>amici::CNodeSolver::rootInit (C++ function), 468</code>	<code>amici::CNodeSolver::setSStolerancesB (C++</code>
<code>amici::CNodeSolver::sensInit1 (C++ function),</code>	<code>function), 466</code>
<code>468</code>	<code>amici::CNodeSolver::setStabLimDet (C++ func-</code>
<code>amici::CNodeSolver::sensReInit (C++ function),</code>	<code>tion), 465</code>
<code>460</code>	<code>amici::CNodeSolver::setStabLimDetB (C++ func-</code>
<code>amici::CNodeSolver::sensToggleOff (C++ func-</code>	<code>tion), 465</code>
<code>tion), 460</code>	<code>amici::CNodeSolver::setStopTime (C++ function),</code>
<code>amici::CNodeSolver::setBandJacFn (C++ func-</code>	<code>462</code>
<code>tion), 468</code>	<code>amici::CNodeSolver::setSuppressAlg (C++ func-</code>
<code>amici::CNodeSolver::setBandJacFnB (C++ func-</code>	<code>tion), 465</code>
<code>tion), 469</code>	<code>amici::CNodeSolver::setUserData (C++ function),</code>
<code>amici::CNodeSolver::setDenseJacFn (C++ func-</code>	<code>464</code>
<code>tion), 468</code>	<code>amici::CNodeSolver::setUserDataB (C++ func-</code>
<code>amici::CNodeSolver::setDenseJacFnB (C++ func-</code>	<code>tion), 464</code>
<code>tion), 468</code>	<code>amici::CNodeSolver::solve (C++ function), 461</code>
<code>amici::CNodeSolver::setErrHandlerFn (C++</code>	<code>amici::CNodeSolver::solveB (C++ function), 461</code>
<code>function), 464</code>	<code>amici::CNodeSolver::solveF (C++ function), 461</code>
<code>amici::CNodeSolver::setId (C++ function), 465</code>	<code>amici::CNodeSolver::Solver (C++ function), 462</code>
<code>amici::CNodeSolver::setJacTimesVecFn (C++</code>	<code>amici::CNodeSolver::turnOffRootFinding (C++</code>
<code>function), 468</code>	<code>function), 462</code>
<code>amici::CNodeSolver::setJacTimesVecFnB (C++</code>	<code>amici::deserializeFromChar (C++ function), 649</code>
<code>function), 469</code>	<code>amici::deserializeFromString (C++ function), 649</code>

amici::dotProd (C++ function), 650  
 amici::ExpData (C++ class), 470  
 amici::ExpData::~~ExpData (C++ function), 471  
 amici::ExpData::applyDataDimension (C++ function), 475  
 amici::ExpData::applyDimensions (C++ function), 475  
 amici::ExpData::applyEventDimension (C++ function), 475  
 amici::ExpData::checkDataDimension (C++ function), 475  
 amici::ExpData::checkEventsDimension (C++ function), 475  
 amici::ExpData::ExpData (C++ function), 470, 471  
 amici::ExpData::getObservedData (C++ function), 472  
 amici::ExpData::getObservedDataPtr (C++ function), 472  
 amici::ExpData::getObservedDataStdDev (C++ function), 473  
 amici::ExpData::getObservedDataStdDevPtr (C++ function), 473  
 amici::ExpData::getObservedEvents (C++ function), 474  
 amici::ExpData::getObservedEventsPtr (C++ function), 474  
 amici::ExpData::getObservedEventsStdDev (C++ function), 474  
 amici::ExpData::getObservedEventsStdDevPtr (C++ function), 474  
 amici::ExpData::getTimepoint (C++ function), 472  
 amici::ExpData::getTimepoints (C++ function), 472  
 amici::ExpData::id (C++ member), 475  
 amici::ExpData::isSetObservedData (C++ function), 472  
 amici::ExpData::isSetObservedDataStdDev (C++ function), 473  
 amici::ExpData::isSetObservedEvents (C++ function), 473  
 amici::ExpData::isSetObservedEventsStdDev (C++ function), 474  
 amici::ExpData::nmaxevent (C++ function), 471  
 amici::ExpData::nmaxevent\_ (C++ member), 475  
 amici::ExpData::nt (C++ function), 472  
 amici::ExpData::nytrue (C++ function), 471  
 amici::ExpData::nytrue\_ (C++ member), 475  
 amici::ExpData::nztrue (C++ function), 471  
 amici::ExpData::nztrue\_ (C++ member), 475  
 amici::ExpData::observed\_data\_ (C++ member), 475  
 amici::ExpData::observed\_data\_std\_dev\_ (C++ member), 476  
 amici::ExpData::observed\_events\_ (C++ member), 476  
 amici::ExpData::observed\_events\_std\_dev\_ (C++ member), 476  
 amici::ExpData::operator== (C++ function), 476  
 amici::ExpData::setObservedData (C++ function), 472  
 amici::ExpData::setObservedDataStdDev (C++ function), 472, 473  
 amici::ExpData::setObservedEvents (C++ function), 473  
 amici::ExpData::setObservedEventsStdDev (C++ function), 474  
 amici::ExpData::setTimepoints (C++ function), 472  
 amici::FinalStateStorer (C++ class), 476  
 amici::FinalStateStorer::~~FinalStateStorer (C++ function), 477  
 amici::FinalStateStorer::FinalStateStorer (C++ function), 477  
 amici::FinalStateStorer::operator= (C++ function), 477  
 amici::FixedParameterContext (C++ enum), 633  
 amici::FixedParameterContext::preequilibration (C++ enumerator), 633  
 amici::FixedParameterContext::presimulation (C++ enumerator), 633  
 amici::FixedParameterContext::simulation (C++ enumerator), 633  
 amici::ForwardProblem (C++ class), 477  
 amici::ForwardProblem::~~ForwardProblem (C++ function), 477  
 amici::ForwardProblem::edata (C++ member), 480  
 amici::ForwardProblem::ForwardProblem (C++ function), 477  
 amici::ForwardProblem::getAdjointUpdates (C++ function), 477  
 amici::ForwardProblem::getCurrentTimeIteration (C++ function), 479  
 amici::ForwardProblem::getDiscontinuities (C++ function), 478  
 amici::ForwardProblem::getDJdydx (C++ function), 478  
 amici::ForwardProblem::getDJzdx (C++ function), 478  
 amici::ForwardProblem::getEventCounter (C++ function), 479  
 amici::ForwardProblem::getFinalSimulationState (C++ function), 479  
 amici::ForwardProblem::getFinalTime (C++ function), 479  
 amici::ForwardProblem::getInitialSimulationState (C++ function), 479  
 amici::ForwardProblem::getNumberOfRoots (C++ function), 478

amici::ForwardProblem::getRHSAtDiscontinuities (C++ function), 478  
 amici::ForwardProblem::getRHSBeforeDiscontinuities (C++ function), 478  
 amici::ForwardProblem::getRootCounter (C++ function), 479  
 amici::ForwardProblem::getRootIndexes (C++ function), 478  
 amici::ForwardProblem::getSimulationStateEvent (C++ function), 479  
 amici::ForwardProblem::getSimulationStateTime (C++ function), 479  
 amici::ForwardProblem::getState (C++ function), 477  
 amici::ForwardProblem::getStateDerivative (C++ function), 478  
 amici::ForwardProblem::getStateDerivativePointer (C++ function), 478  
 amici::ForwardProblem::getStateDerivativeSensitivityPointer (C++ function), 479  
 amici::ForwardProblem::getStatePointer (C++ function), 478  
 amici::ForwardProblem::getStatesAtDiscontinuities (C++ function), 478  
 amici::ForwardProblem::getStateSensitivity (C++ function), 478  
 amici::ForwardProblem::getStateSensitivityPointer (C++ function), 479  
 amici::ForwardProblem::getTime (C++ function), 477  
 amici::ForwardProblem::model (C++ member), 480  
 amici::ForwardProblem::solver (C++ member), 480  
 amici::ForwardProblem::workForwardProblem (C++ function), 477  
 amici::getScaledParameter (C++ function), 650  
 amici::getUnscaledParameter (C++ function), 650  
 amici::hdf5::attributeExists (C++ function), 651  
 amici::hdf5::createAndWriteDouble1DDataset (C++ function), 651  
 amici::hdf5::createAndWriteDouble2DDataset (C++ function), 652  
 amici::hdf5::createAndWriteDouble3DDataset (C++ function), 652  
 amici::hdf5::createAndWriteInt1DDataset (C++ function), 653  
 amici::hdf5::createAndWriteInt2DDataset (C++ function), 653  
 amici::hdf5::createGroup (C++ function), 654  
 amici::hdf5::createOrOpenForWriting (C++ function), 654  
 amici::hdf5::getDoubleDataset1D (C++ function), 654  
 amici::hdf5::getDoubleDataset2D (C++ function), 655  
 amici::hdf5::getDoubleDataset3D (C++ function), 655  
 amici::hdf5::getDoubleScalarAttribute (C++ function), 656  
 amici::hdf5::getIntDataset1D (C++ function), 656  
 amici::hdf5::getIntScalarAttribute (C++ function), 656  
 amici::hdf5::getStringAttribute (C++ function), 657  
 amici::hdf5::locationExists (C++ function), 657  
 amici::hdf5::readModelDataFromHDF5 (C++ function), 658  
 amici::hdf5::readSimulationExpData (C++ function), 658  
 amici::hdf5::readSolverSettingsFromHDF5 (C++ function), 659  
 amici::hdf5::writeReturnData (C++ function), 660  
 amici::hdf5::writeReturnDataDiagnosis (C++ function), 660  
 amici::hdf5::writeSimulationExpData (C++ function), 661  
 amici::hdf5::writeSolverSettingsToHDF5 (C++ function), 661  
 amici::HermiteSpline (C++ class), 480  
 amici::HermiteSpline::compute\_coefficients (C++ function), 481  
 amici::HermiteSpline::compute\_coefficients\_sensitivity (C++ function), 481  
 amici::HermiteSpline::compute\_final\_sensitivity (C++ function), 481  
 amici::HermiteSpline::compute\_final\_value (C++ function), 481  
 amici::HermiteSpline::get\_node\_derivative (C++ function), 482  
 amici::HermiteSpline::get\_node\_derivative\_by\_fd (C++ function), 482  
 amici::HermiteSpline::get\_node\_derivative\_scaled (C++ function), 482  
 amici::HermiteSpline::get\_sensitivity\_scaled (C++ function), 482  
 amici::HermiteSpline::get\_value\_scaled (C++ function), 482  
 amici::HermiteSpline::HermiteSpline (C++ function), 480  
 amici::IDAException (C++ class), 483  
 amici::IDAException::IDAException (C++ function), 483  
 amici::IDASolver (C++ class), 483  
 amici::IDASolver::~IDASolver (C++ function), 483  
 amici::IDASolver::adjInit (C++ function), 489  
 amici::IDASolver::allocateSolver (C++ function), 487  
 amici::IDASolver::allocateSolverB (C++ function), 487



tion), 489

amici::IDASolver::binit (C++ function), 491

amici::IDASolver::calcIC (C++ function), 486

amici::IDASolver::calcICB (C++ function), 486

amici::IDASolver::clone (C++ function), 483

amici::IDASolver::diag (C++ function), 490

amici::IDASolver::diagB (C++ function), 490

amici::IDASolver::getAdjBmem (C++ function), 490

amici::IDASolver::getB (C++ function), 485

amici::IDASolver::getDky (C++ function), 485

amici::IDASolver::getDkyB (C++ function), 485

amici::IDASolver::getLastOrder (C++ function), 490

amici::IDASolver::getModel (C++ function), 486

amici::IDASolver::getNumErrTestFails (C++ function), 490

amici::IDASolver::getNumNonlinSolvConvFails (C++ function), 490

amici::IDASolver::getNumRhsEvals (C++ function), 490

amici::IDASolver::getNumSteps (C++ function), 490

amici::IDASolver::getQuad (C++ function), 486

amici::IDASolver::getQuadB (C++ function), 486

amici::IDASolver::getQuadDky (C++ function), 486

amici::IDASolver::getQuadDkyB (C++ function), 486

amici::IDASolver::getRootInfo (C++ function), 485

amici::IDASolver::getSens (C++ function), 485

amici::IDASolver::getSensDky (C++ function), 485

amici::IDASolver::init (C++ function), 491

amici::IDASolver::initSteadystate (C++ function), 491

amici::IDASolver::qbinit (C++ function), 491

amici::IDASolver::quadInit (C++ function), 489

amici::IDASolver::quadReInitB (C++ function), 484

amici::IDASolver::quadSStolerances (C++ function), 484

amici::IDASolver::quadSStolerancesB (C++ function), 484

amici::IDASolver::reInit (C++ function), 483

amici::IDASolver::reInitB (C++ function), 484

amici::IDASolver::reInitPostProcess (C++ function), 487

amici::IDASolver::reInitPostProcessB (C++ function), 483

amici::IDASolver::reInitPostProcessF (C++ function), 483

amici::IDASolver::resetState (C++ function), 489

amici::IDASolver::rootInit (C++ function), 491

amici::IDASolver::sensInit1 (C++ function), 491

amici::IDASolver::sensReInit (C++ function), 484

amici::IDASolver::sensToggleOff (C++ function), 484

amici::IDASolver::setBandJacFn (C++ function), 492

amici::IDASolver::setBandJacFnB (C++ function), 492

amici::IDASolver::setDenseJacFn (C++ function), 492

amici::IDASolver::setDenseJacFnB (C++ function), 492

amici::IDASolver::setErrHandlerFn (C++ function), 488

amici::IDASolver::setId (C++ function), 488

amici::IDASolver::setJacTimesVecFn (C++ function), 492

amici::IDASolver::setJacTimesVecFnB (C++ function), 492

amici::IDASolver::setLinearSolver (C++ function), 487

amici::IDASolver::setLinearSolverB (C++ function), 487

amici::IDASolver::setMaxNumSteps (C++ function), 488

amici::IDASolver::setMaxNumStepsB (C++ function), 489

amici::IDASolver::setNonLinearSolver (C++ function), 487

amici::IDASolver::setNonLinearSolverB (C++ function), 487

amici::IDASolver::setNonLinearSolverSens (C++ function), 487

amici::IDASolver::setQuadErrCon (C++ function), 488

amici::IDASolver::setQuadErrConB (C++ function), 488

amici::IDASolver::setSensErrCon (C++ function), 488

amici::IDASolver::setSensParams (C++ function), 489

amici::IDASolver::setSensSStolerances (C++ function), 487

amici::IDASolver::setSparseJacFn (C++ function), 492

amici::IDASolver::setSparseJacFn\_ss (C++ function), 492

amici::IDASolver::setSparseJacFnB (C++ function), 492

amici::IDASolver::setSStolerances (C++ function), 487

amici::IDASolver::setSStolerancesB (C++ function), 489

amici::IDASolver::setStabLimDet (C++ function), 488

amici::IDASolver::setStabLimDetB (C++ function), 488

tion), 488

`amici::IDASolver::setStopTime` (C++ function), 486

`amici::IDASolver::setSuppressAlg` (C++ function), 489

`amici::IDASolver::setUserData` (C++ function), 488

`amici::IDASolver::setUserDataB` (C++ function), 488

`amici::IDASolver::solve` (C++ function), 484

`amici::IDASolver::solveB` (C++ function), 485

`amici::IDASolver::solveF` (C++ function), 485

`amici::IDASolver::Solver` (C++ function), 487

`amici::IDASolver::turnOffRootFinding` (C++ function), 486

`amici::IntegrationFailure` (C++ class), 493

`amici::IntegrationFailure::error_code` (C++ member), 493

`amici::IntegrationFailure::IntegrationFailure` (C++ function), 493

`amici::IntegrationFailure::time` (C++ member), 493

`amici::IntegrationFailureB` (C++ class), 493

`amici::IntegrationFailureB::error_code` (C++ member), 494

`amici::IntegrationFailureB::IntegrationFailureB` (C++ function), 494

`amici::IntegrationFailureB::time` (C++ member), 494

`amici::InternalSensitivityMethod` (C++ enum), 633

`amici::InternalSensitivityMethod::simultaneous` (C++ enumerator), 633

`amici::InternalSensitivityMethod::staggered` (C++ enumerator), 633

`amici::InternalSensitivityMethod::staggered1` (C++ enumerator), 634

`amici::InterpolationType` (C++ enum), 634

`amici::InterpolationType::hermite` (C++ enumerator), 634

`amici::InterpolationType::polynomial` (C++ enumerator), 634

`amici::is_equal` (C++ function), 662

`amici::LinearMultistepMethod` (C++ enum), 634

`amici::LinearMultistepMethod::adams` (C++ enumerator), 634

`amici::LinearMultistepMethod::BDF` (C++ enumerator), 634

`amici::LinearSolver` (C++ enum), 635

`amici::LinearSolver::band` (C++ enumerator), 635

`amici::LinearSolver::dense` (C++ enumerator), 635

`amici::LinearSolver::diag` (C++ enumerator), 635

`amici::LinearSolver::KLU` (C++ enumerator), 635

`amici::LinearSolver::LAPACKBand` (C++ enumerator), 635

`amici::LinearSolver::LAPACKDense` (C++ enumerator), 635

`amici::LinearSolver::SPBCG` (C++ enumerator), 635

`amici::LinearSolver::SPGMR` (C++ enumerator), 635

`amici::LinearSolver::SPTFQMR` (C++ enumerator), 635

`amici::LinearSolver::SuperLUMT` (C++ enumerator), 635

`amici::linearSum` (C++ function), 662

`amici::Logger` (C++ class), 494

`amici::Logger::items` (C++ member), 495

`amici::Logger::log` (C++ function), 494

`amici::Logger::Logger` (C++ function), 494

`amici::LogItem` (C++ struct), 415

`amici::LogItem::identifier` (C++ member), 416

`amici::LogItem::LogItem` (C++ function), 416

`amici::LogItem::message` (C++ member), 416

`amici::LogItem::severity` (C++ member), 416

`amici::LogSeverity` (C++ enum), 635

`amici::LogSeverity::debug` (C++ enumerator), 635

`amici::LogSeverity::error` (C++ enumerator), 635

`amici::LogSeverity::warning` (C++ enumerator), 635

`amici::Model` (C++ class), 495

`amici::Model::~~Model` (C++ function), 496

`amici::Model::addAdjointQuadratureEventUpdate` (C++ function), 512

`amici::Model::addAdjointStateEventUpdate` (C++ function), 511

`amici::Model::addEventObjective` (C++ function), 509

`amici::Model::addEventObjectiveRegularization` (C++ function), 509

`amici::Model::addEventObjectiveSensitivity` (C++ function), 509

`amici::Model::addObservableObjective` (C++ function), 506

`amici::Model::addObservableObjectiveSensitivity` (C++ function), 506

`amici::Model::addPartialEventObjectiveSensitivity` (C++ function), 510

`amici::Model::addPartialObservableObjectiveSensitivity` (C++ function), 507

`amici::Model::addStateEventUpdate` (C++ function), 511

`amici::Model::addStateSensitivityEventUpdate` (C++ function), 511

`amici::Model::always_check_finite_` (C++ member), 539

`amici::Model::any_state_non_negative_` (C++

member), 538

amici::Model::boost::serialization::serialize (C++ function), 539

amici::Model::checkFinite (C++ function), 512

amici::Model::checkLLHBufferSize (C++ function), 530

amici::Model::clone (C++ function), 496

amici::Model::computeX\_pos (C++ function), 537, 538

amici::Model::derived\_state\_ (C++ member), 538

amici::Model::fdeltaqB (C++ function), 514

amici::Model::fdeltasx (C++ function), 514

amici::Model::fdeltax (C++ function), 515

amici::Model::fdeltaxB (C++ function), 515

amici::Model::fdJrzdsigma (C++ function), 516, 535

amici::Model::fdJrzdz (C++ function), 516, 535

amici::Model::fdJydp (C++ function), 532

amici::Model::fdJydsigma (C++ function), 516, 532

amici::Model::fdJydx (C++ function), 532

amici::Model::fdJydy (C++ function), 517, 531

amici::Model::fdJydy\_colptrs (C++ function), 517

amici::Model::fdJydy\_rowvals (C++ function), 517

amici::Model::fdJzdp (C++ function), 534

amici::Model::fdJzdsigma (C++ function), 517, 534

amici::Model::fdJzdx (C++ function), 534

amici::Model::fdJzdz (C++ function), 517, 534

amici::Model::fdrzdp (C++ function), 518, 533

amici::Model::fdrzdx (C++ function), 518, 533

amici::Model::fdsigmaydp (C++ function), 518, 531

amici::Model::fdsigmaydy (C++ function), 519, 531

amici::Model::fdsigmazdp (C++ function), 519, 533

amici::Model::fdtotal\_cldp (C++ function), 519

amici::Model::fdtotal\_cldx\_rdata (C++ function), 519

amici::Model::fdtotal\_cldx\_rdata\_colptrs (C++ function), 520

amici::Model::fdtotal\_cldx\_rdata\_rowvals (C++ function), 520

amici::Model::fdwdp (C++ function), 520, 535

amici::Model::fdwdp\_colptrs (C++ function), 521

amici::Model::fdwdp\_rowvals (C++ function), 521

amici::Model::fdwdw (C++ function), 521, 536

amici::Model::fdwdw\_colptrs (C++ function), 521

amici::Model::fdwdw\_rowvals (C++ function), 521

amici::Model::fdwdx (C++ function), 521, 536

amici::Model::fdwdx\_colptrs (C++ function), 522

amici::Model::fdwdx\_rowvals (C++ function), 522

amici::Model::fdx\_rdatadp (C++ function), 522

amici::Model::fdx\_rdatadtcl (C++ function), 522

amici::Model::fdx\_rdatadtcl\_colptrs (C++ function), 522

amici::Model::fdx\_rdatadtcl\_rowvals (C++ function), 522

amici::Model::fdx\_rdatadx\_solver (C++ function), 522

amici::Model::fdx\_rdatadx\_solver\_colptrs (C++ function), 523

amici::Model::fdx\_rdatadx\_solver\_rowvals (C++ function), 523

amici::Model::fdydp (C++ function), 523, 530

amici::Model::fdydx (C++ function), 524, 531

amici::Model::fdzdp (C++ function), 524, 532

amici::Model::fdzdx (C++ function), 524, 532

amici::Model::fJrz (C++ function), 525, 534

amici::Model::fJy (C++ function), 525, 531

amici::Model::fJz (C++ function), 525, 533

amici::Model::frz (C++ function), 525, 532

amici::Model::fsdx0 (C++ function), 513

amici::Model::fsigmay (C++ function), 526, 531

amici::Model::fsigmaz (C++ function), 526, 533

amici::Model::fspl (C++ function), 535

amici::Model::fsrz (C++ function), 526

amici::Model::fsspl (C++ function), 535

amici::Model::fstau (C++ function), 526

amici::Model::fstotal\_cl (C++ function), 537

amici::Model::fsx0 (C++ function), 513, 527

amici::Model::fsx0\_fixedParameters (C++ function), 513, 527

amici::Model::fsx\_rdata (C++ function), 514, 536

amici::Model::fsx\_solver (C++ function), 537

amici::Model::fsz (C++ function), 527

amici::Model::ftotal\_cl (C++ function), 537

amici::Model::fw (C++ function), 528, 535

amici::Model::fx0 (C++ function), 513, 528

amici::Model::fx0\_fixedParameters (C++ function), 513, 528

amici::Model::fx\_rdata (C++ function), 513, 536

amici::Model::fx\_solver (C++ function), 536

amici::Model::fy (C++ function), 529, 530

amici::Model::fz (C++ function), 529, 532

amici::Model::get\_dxdotdp (C++ function), 514

amici::Model::get\_dxdotdp\_full (C++ function), 514

amici::Model::getAddSigmaResiduals (C++ function), 504

amici::Model::getAdjointStateEventUpdate (C++ function), 510

amici::Model::getAdjointStateObservableUpdate (C++ function), 507

amici::Model::getAlwaysCheckFinite (C++ function), 513

amici::Model::getEvent (C++ function), 507

amici::Model::getEventRegularization (C++ function), 508

amici::Model::getEventRegularizationSensitivity (C++ function), 508

`amici::Model::getEventSensitivity` (C++ function), 507  
`amici::Model::getEventSigma` (C++ function), 508  
`amici::Model::getEventSigmaSensitivity` (C++ function), 509  
`amici::Model::getEventTimeSensitivity` (C++ function), 510  
`amici::Model::getExpression` (C++ function), 505  
`amici::Model::getExpressionIds` (C++ function), 502  
`amici::Model::getExpressionNames` (C++ function), 501  
`amici::Model::getFixedParameterById` (C++ function), 499  
`amici::Model::getFixedParameterByName` (C++ function), 499  
`amici::Model::getFixedParameterIds` (C++ function), 502  
`amici::Model::getFixedParameterNames` (C++ function), 501  
`amici::Model::getFixedParameters` (C++ function), 499  
`amici::Model::getInitialStates` (C++ function), 504  
`amici::Model::getInitialStateSensitivities` (C++ function), 504  
`amici::Model::getMinimumSigmaResiduals` (C++ function), 503  
`amici::Model::getModelState` (C++ function), 503  
`amici::Model::getName` (C++ function), 500  
`amici::Model::getObservable` (C++ function), 505  
`amici::Model::getObservableIds` (C++ function), 502  
`amici::Model::getObservableNames` (C++ function), 501  
`amici::Model::getObservableScaling` (C++ function), 505  
`amici::Model::getObservableSensitivity` (C++ function), 506  
`amici::Model::getObservableSigma` (C++ function), 506  
`amici::Model::getObservableSigmaSensitivity` (C++ function), 506  
`amici::Model::getParameterById` (C++ function), 498  
`amici::Model::getParameterByName` (C++ function), 498  
`amici::Model::getParameterIds` (C++ function), 501  
`amici::Model::getParameterList` (C++ function), 504  
`amici::Model::getParameterNames` (C++ function), 500  
`amici::Model::getParameters` (C++ function), 498  
`amici::Model::getParameterScale` (C++ function), 498  
`amici::Model::getReinitializationStateIdxs` (C++ function), 514  
`amici::Model::getReinitializeFixedParameterInitialStates` (C++ function), 505  
`amici::Model::getStateIds` (C++ function), 502  
`amici::Model::getStateIdsSolver` (C++ function), 502  
`amici::Model::getStateIsNonNegative` (C++ function), 503  
`amici::Model::getStateNames` (C++ function), 501  
`amici::Model::getStateNamesSolver` (C++ function), 501  
`amici::Model::getSteadyStateComputationMode` (C++ function), 505  
`amici::Model::getSteadyStateSensitivityMode` (C++ function), 505  
`amici::Model::getTimepoint` (C++ function), 503  
`amici::Model::getTimepoints` (C++ function), 502  
`amici::Model::getUnobservedEventSensitivity` (C++ function), 508  
`amici::Model::getUnscaledParameters` (C++ function), 498  
`amici::Model::hasCustomInitialStates` (C++ function), 504  
`amici::Model::hasCustomInitialStateSensitivities` (C++ function), 504  
`amici::Model::hasExpressionIds` (C++ function), 502  
`amici::Model::hasExpressionNames` (C++ function), 501  
`amici::Model::hasFixedParameterIds` (C++ function), 502  
`amici::Model::hasFixedParameterNames` (C++ function), 501  
`amici::Model::hasObservableIds` (C++ function), 502  
`amici::Model::hasObservableNames` (C++ function), 501  
`amici::Model::hasParameterIds` (C++ function), 501  
`amici::Model::hasParameterNames` (C++ function), 500  
`amici::Model::hasQuadraticLLH` (C++ function), 502  
`amici::Model::hasStateIds` (C++ function), 502  
`amici::Model::hasStateNames` (C++ function), 501  
`amici::Model::idlist` (C++ member), 529  
`amici::Model::initEvents` (C++ function), 497  
`amici::Model::initialize` (C++ function), 496  
`amici::Model::initializeB` (C++ function), 496  
`amici::Model::initializeSplines` (C++ function), 497



---

```

amici::Model::initializeSplineSensitivities (C++ function), 497
amici::Model::initializeStates (C++ function), 496
amici::Model::initializeStateSensitivities (C++ function), 496
amici::Model::initializeVectors (C++ function), 530
amici::Model::k (C++ function), 497
amici::Model::logger (C++ member), 529
amici::Model::min_sigma_ (C++ member), 539
amici::Model::Model (C++ function), 495
amici::Model::ncl (C++ function), 497
amici::Model::nk (C++ function), 497
amici::Model::nMaxEvent (C++ function), 497
amici::Model::nmaxevent_ (C++ member), 538
amici::Model::np (C++ function), 497
amici::Model::nplist (C++ function), 497
amici::Model::nt (C++ function), 498
amici::Model::nx_reinit (C++ function), 497
amici::Model::o2mode (C++ member), 529
amici::Model::operator= (C++ function), 496
amici::Model::operator== (C++ function), 539
amici::Model::plist (C++ function), 504
amici::Model::pythonGenerated (C++ member), 529
amici::Model::requireSensitivitiesForAllParameters (C++ function), 505
amici::Model::root_initial_values_ (C++ member), 538
amici::Model::setAddSigmaResiduals (C++ function), 503
amici::Model::setAllStatesNonNegative (C++ function), 503
amici::Model::setAlwaysCheckFinite (C++ function), 513
amici::Model::setFixedParameterById (C++ function), 500
amici::Model::setFixedParameterByName (C++ function), 500
amici::Model::setFixedParameters (C++ function), 500
amici::Model::setFixedParametersByIdRegex (C++ function), 500
amici::Model::setFixedParametersByNameRegex (C++ function), 500
amici::Model::setInitialStates (C++ function), 504
amici::Model::setInitialStateSensitivities (C++ function), 504
amici::Model::setMinimumSigmaResiduals (C++ function), 503
amici::Model::setModelState (C++ function), 503
amici::Model::setNMaxEvent (C++ function), 497
amici::Model::setParameterById (C++ function), 498, 499
amici::Model::setParameterByName (C++ function), 499
amici::Model::setParameterList (C++ function), 504
amici::Model::setParameters (C++ function), 498
amici::Model::setParametersByIdRegex (C++ function), 499
amici::Model::setParametersByNameRegex (C++ function), 499
amici::Model::setParameterScale (C++ function), 498
amici::Model::setReinitializationStateIdxs (C++ function), 514
amici::Model::setReinitializeFixedParameterInitialStates (C++ function), 505
amici::Model::setStateIsNonNegative (C++ function), 503
amici::Model::setSteadyStateComputationMode (C++ function), 505
amici::Model::setSteadyStateSensitivityMode (C++ function), 505
amici::Model::setT0 (C++ function), 503
amici::Model::setTimepoints (C++ function), 503
amici::Model::setUnscaledInitialStateSensitivities (C++ function), 504
amici::Model::sigma_res_ (C++ member), 539
amici::Model::splines_ (C++ member), 538
amici::Model::state_ (C++ member), 538
amici::Model::state_is_non_negative_ (C++ member), 538
amici::Model::steadystate_computation_mode_ (C++ member), 538
amici::Model::steadystate_sensitivity_mode_ (C++ member), 539
amici::Model::sx0data_ (C++ member), 538
amici::Model::t0 (C++ function), 503
amici::Model::updateHeaviside (C++ function), 512
amici::Model::updateHeavisideB (C++ function), 512
amici::Model::writeLLHSensitivitySlice (C++ function), 530
amici::Model::writeSensitivitySliceEvent (C++ function), 530
amici::Model::writeSliceEvent (C++ function), 530
amici::Model::x0data_ (C++ member), 538
amici::Model::z2event_ (C++ member), 538
amici::Model_DAE (C++ class), 540
amici::Model_DAE::fdxdotdp (C++ function), 546, 548
amici::Model_DAE::fdxdotdp_explicit (C++

```

*function*), 548  
amici::Model\_DAE::fdxdotdp\_explicit\_colptrs  
(C++ *function*), 549  
amici::Model\_DAE::fdxdotdp\_explicit\_rowvals  
(C++ *function*), 549  
amici::Model\_DAE::fdxdotdw (C++ *function*), 549,  
550  
amici::Model\_DAE::fdxdotdw\_colptrs (C++ *func-*  
*tion*), 550  
amici::Model\_DAE::fdxdotdw\_rowvals (C++ *func-*  
*tion*), 550  
amici::Model\_DAE::fdxdotdx\_explicit (C++  
*function*), 549  
amici::Model\_DAE::fdxdotdx\_explicit\_colptrs  
(C++ *function*), 549  
amici::Model\_DAE::fdxdotdx\_explicit\_rowvals  
(C++ *function*), 549  
amici::Model\_DAE::fJ (C++ *function*), 540, 541  
amici::Model\_DAE::fJB (C++ *function*), 541  
amici::Model\_DAE::fJDiag (C++ *function*), 542  
amici::Model\_DAE::fJSparse (C++ *function*), 541,  
542, 547  
amici::Model\_DAE::fJSparseB (C++ *function*), 542  
amici::Model\_DAE::fJSparseB\_ss (C++ *function*),  
545  
amici::Model\_DAE::fJv (C++ *function*), 543  
amici::Model\_DAE::fJvB (C++ *function*), 543  
amici::Model\_DAE::fM (C++ *function*), 547, 550  
amici::Model\_DAE::fqBdot (C++ *function*), 545  
amici::Model\_DAE::fqBdot\_ss (C++ *function*), 545  
amici::Model\_DAE::froot (C++ *function*), 544, 547  
amici::Model\_DAE::fsxdot (C++ *function*), 546  
amici::Model\_DAE::fxBdot (C++ *function*), 544  
amici::Model\_DAE::fxBdot\_ss (C++ *function*), 545  
amici::Model\_DAE::fxdot (C++ *function*), 544, 548  
amici::Model\_DAE::getSolver (C++ *function*), 547  
amici::Model\_DAE::Model\_DAE (C++ *function*), 540  
amici::Model\_DAE::writeSteadystateJB (C++  
*function*), 546  
amici::Model\_ODE (C++ *class*), 551  
amici::Model\_ODE::fdxdotdp (C++ *function*), 559,  
561  
amici::Model\_ODE::fdxdotdp\_explicit (C++  
*function*), 559  
amici::Model\_ODE::fdxdotdp\_explicit\_colptrs  
(C++ *function*), 559  
amici::Model\_ODE::fdxdotdp\_explicit\_rowvals  
(C++ *function*), 559  
amici::Model\_ODE::fdxdotdw (C++ *function*), 560  
amici::Model\_ODE::fdxdotdw\_colptrs (C++ *func-*  
*tion*), 560  
amici::Model\_ODE::fdxdotdw\_rowvals (C++ *func-*  
*tion*), 560  
amici::Model\_ODE::fdxdotdx\_explicit (C++  
*function*), 559  
amici::Model\_ODE::fdxdotdx\_explicit\_colptrs  
(C++ *function*), 560  
amici::Model\_ODE::fdxdotdx\_explicit\_rowvals  
(C++ *function*), 560  
amici::Model\_ODE::fJ (C++ *function*), 551, 552  
amici::Model\_ODE::fJB (C++ *function*), 552  
amici::Model\_ODE::fJDiag (C++ *function*), 553, 554  
amici::Model\_ODE::fJSparse (C++ *function*), 552,  
553, 557, 558  
amici::Model\_ODE::fJSparse\_colptrs (C++ *func-*  
*tion*), 558  
amici::Model\_ODE::fJSparse\_rowvals (C++ *func-*  
*tion*), 558  
amici::Model\_ODE::fJSparseB (C++ *function*), 553  
amici::Model\_ODE::fJSparseB\_ss (C++ *function*),  
556  
amici::Model\_ODE::fJv (C++ *function*), 554  
amici::Model\_ODE::fJvB (C++ *function*), 554  
amici::Model\_ODE::fqBdot (C++ *function*), 556  
amici::Model\_ODE::fqBdot\_ss (C++ *function*), 556  
amici::Model\_ODE::froot (C++ *function*), 555, 558  
amici::Model\_ODE::fsxdot (C++ *function*), 557  
amici::Model\_ODE::fxBdot (C++ *function*), 555  
amici::Model\_ODE::fxBdot\_ss (C++ *function*), 556  
amici::Model\_ODE::fxdot (C++ *function*), 555, 558  
amici::Model\_ODE::getSolver (C++ *function*), 557  
amici::Model\_ODE::Model\_ODE (C++ *function*), 551  
amici::Model\_ODE::writeSteadystateJB (C++  
*function*), 556  
amici::model\_quantity\_to\_str (C++ *member*), 678  
amici::ModelContext (C++ *class*), 561  
amici::ModelContext::~ModelContext (C++ *func-*  
*tion*), 561  
amici::ModelContext::ModelContext (C++ *func-*  
*tion*), 561  
amici::ModelContext::operator= (C++ *function*),  
561  
amici::ModelContext::restore (C++ *function*), 561  
amici::ModelDimensions (C++ *struct*), 416  
amici::ModelDimensions::lbw (C++ *member*), 419  
amici::ModelDimensions::ModelDimensions  
(C++ *function*), 417  
amici::ModelDimensions::ndJydy (C++ *member*),  
419  
amici::ModelDimensions::ndtotal\_cldx\_rdata  
(C++ *member*), 419  
amici::ModelDimensions::ndwdp (C++ *member*),  
419  
amici::ModelDimensions::ndwdw (C++ *member*),  
419  
amici::ModelDimensions::ndwdx (C++ *member*),  
418

amici::ModelDimensions::ndxdotdw (C++ member), 419  
 amici::ModelDimensions::ndxrdatadtcl (C++ member), 419  
 amici::ModelDimensions::ndxrdatadxsolver (C++ member), 419  
 amici::ModelDimensions::ne (C++ member), 418  
 amici::ModelDimensions::nJ (C++ member), 419  
 amici::ModelDimensions::nk (C++ member), 418  
 amici::ModelDimensions::nnz (C++ member), 419  
 amici::ModelDimensions::np (C++ member), 418  
 amici::ModelDimensions::nspl (C++ member), 418  
 amici::ModelDimensions::nw (C++ member), 418  
 amici::ModelDimensions::nx\_rdata (C++ member), 418  
 amici::ModelDimensions::nx\_solver (C++ member), 418  
 amici::ModelDimensions::nx\_solver\_reinit (C++ member), 418  
 amici::ModelDimensions::nxtrue\_rdata (C++ member), 418  
 amici::ModelDimensions::nxtrue\_solver (C++ member), 418  
 amici::ModelDimensions::ny (C++ member), 418  
 amici::ModelDimensions::nytrue (C++ member), 418  
 amici::ModelDimensions::nz (C++ member), 418  
 amici::ModelDimensions::nztrue (C++ member), 418  
 amici::ModelDimensions::ubw (C++ member), 419  
 amici::ModelQuantity (C++ enum), 636  
 amici::ModelQuantity::deltaqB (C++ enumerator), 638  
 amici::ModelQuantity::deltasx (C++ enumerator), 637  
 amici::ModelQuantity::deltax (C++ enumerator), 637  
 amici::ModelQuantity::deltaxB (C++ enumerator), 637  
 amici::ModelQuantity::dJrzdsigma (C++ enumerator), 638  
 amici::ModelQuantity::dJrzdxdx (C++ enumerator), 638  
 amici::ModelQuantity::dJrzdxdz (C++ enumerator), 638  
 amici::ModelQuantity::dJydsigma (C++ enumerator), 638  
 amici::ModelQuantity::dJydx (C++ enumerator), 638  
 amici::ModelQuantity::dJydy (C++ enumerator), 637  
 amici::ModelQuantity::dJydy\_matlab (C++ enumerator), 637  
 amici::ModelQuantity::dJzdsigma (C++ enumerator), 638  
 amici::ModelQuantity::dJzdx (C++ enumerator), 638  
 amici::ModelQuantity::dJzdz (C++ enumerator), 638  
 amici::ModelQuantity::drzdp (C++ enumerator), 638  
 amici::ModelQuantity::drzdx (C++ enumerator), 638  
 amici::ModelQuantity::dsigmaydp (C++ enumerator), 638  
 amici::ModelQuantity::dsigmaydy (C++ enumerator), 638  
 amici::ModelQuantity::dsigmazdp (C++ enumerator), 638  
 amici::ModelQuantity::dwdp (C++ enumerator), 637  
 amici::ModelQuantity::dwdw (C++ enumerator), 637  
 amici::ModelQuantity::dwdx (C++ enumerator), 637  
 amici::ModelQuantity::dydp (C++ enumerator), 637  
 amici::ModelQuantity::dydx (C++ enumerator), 637  
 amici::ModelQuantity::dzdp (C++ enumerator), 638  
 amici::ModelQuantity::dzdx (C++ enumerator), 638  
 amici::ModelQuantity::J (C++ enumerator), 636  
 amici::ModelQuantity::JB (C++ enumerator), 636  
 amici::ModelQuantity::JDiag (C++ enumerator), 636  
 amici::ModelQuantity::JSparseB\_ss (C++ enumerator), 637  
 amici::ModelQuantity::Jv (C++ enumerator), 636  
 amici::ModelQuantity::JvB (C++ enumerator), 636  
 amici::ModelQuantity::k (C++ enumerator), 637  
 amici::ModelQuantity::p (C++ enumerator), 637  
 amici::ModelQuantity::qBdot (C++ enumerator), 637  
 amici::ModelQuantity::qBdot\_ss (C++ enumerator), 637  
 amici::ModelQuantity::root (C++ enumerator), 637  
 amici::ModelQuantity::srz (C++ enumerator), 636  
 amici::ModelQuantity::ssigmay (C++ enumerator), 636  
 amici::ModelQuantity::ssigmaz (C++ enumerator), 636  
 amici::ModelQuantity::sx (C++ enumerator), 636  
 amici::ModelQuantity::sxdot (C++ enumerator), 636  
 amici::ModelQuantity::sy (C++ enumerator), 636

amici::ModelQuantity::sz (C++ *enumerator*), 636  
 amici::ModelQuantity::ts (C++ *enumerator*), 637  
 amici::ModelQuantity::w (C++ *enumerator*), 637  
 amici::ModelQuantity::x (C++ *enumerator*), 637  
 amici::ModelQuantity::x0 (C++ *enumerator*), 636  
 amici::ModelQuantity::x0\_rdata (C++ *enumerator*), 636  
 amici::ModelQuantity::x\_rdata (C++ *enumerator*), 636  
 amici::ModelQuantity::xBdot (C++ *enumerator*), 636  
 amici::ModelQuantity::xBdot\_ss (C++ *enumerator*), 637  
 amici::ModelQuantity::xdot (C++ *enumerator*), 636  
 amici::ModelQuantity::y (C++ *enumerator*), 637  
 amici::ModelState (C++ *struct*), 420  
 amici::ModelState::fixedParameters (C++ *member*), 420  
 amici::ModelState::h (C++ *member*), 420  
 amici::ModelState::plist (C++ *member*), 420  
 amici::ModelState::spl\_ (C++ *member*), 420  
 amici::ModelState::stotal\_cl (C++ *member*), 420  
 amici::ModelState::total\_cl (C++ *member*), 420  
 amici::ModelState::unscaledParameters (C++ *member*), 420  
 amici::ModelStateDerived (C++ *struct*), 420  
 amici::ModelStateDerived::deltaqB\_ (C++ *member*), 424  
 amici::ModelStateDerived::deltasx\_ (C++ *member*), 424  
 amici::ModelStateDerived::deltax\_ (C++ *member*), 424  
 amici::ModelStateDerived::deltaxB\_ (C++ *member*), 424  
 amici::ModelStateDerived::dfdx\_ (C++ *member*), 421  
 amici::ModelStateDerived::dJrzdsigma\_ (C++ *member*), 423  
 amici::ModelStateDerived::dJrzdz\_ (C++ *member*), 422  
 amici::ModelStateDerived::dJydp\_ (C++ *member*), 422  
 amici::ModelStateDerived::dJydsigma\_ (C++ *member*), 422  
 amici::ModelStateDerived::dJydx\_ (C++ *member*), 422  
 amici::ModelStateDerived::dJydy\_ (C++ *member*), 422  
 amici::ModelStateDerived::dJydy\_matlab\_ (C++ *member*), 422  
 amici::ModelStateDerived::dJzdp\_ (C++ *member*), 423  
 amici::ModelStateDerived::dJzdsigma\_ (C++ *member*), 422  
 amici::ModelStateDerived::dJzdx\_ (C++ *member*), 423  
 amici::ModelStateDerived::dJzdz\_ (C++ *member*), 422  
 amici::ModelStateDerived::drzdp\_ (C++ *member*), 423  
 amici::ModelStateDerived::drzdx\_ (C++ *member*), 423  
 amici::ModelStateDerived::dsigmaydp\_ (C++ *member*), 424  
 amici::ModelStateDerived::dsigmaydy\_ (C++ *member*), 424  
 amici::ModelStateDerived::dsigmazdp\_ (C++ *member*), 424  
 amici::ModelStateDerived::dtotal\_cldx\_rdata (C++ *member*), 422  
 amici::ModelStateDerived::dwdp\_ (C++ *member*), 421  
 amici::ModelStateDerived::dwdx\_ (C++ *member*), 421  
 amici::ModelStateDerived::dx\_rdatadtcl (C++ *member*), 422  
 amici::ModelStateDerived::dx\_rdatadx\_solver (C++ *member*), 422  
 amici::ModelStateDerived::dxdotdp (C++ *member*), 422  
 amici::ModelStateDerived::dxdotdp\_explicit (C++ *member*), 421  
 amici::ModelStateDerived::dxdotdp\_full (C++ *member*), 421  
 amici::ModelStateDerived::dxdotdp\_implicit (C++ *member*), 421  
 amici::ModelStateDerived::dxdotdw\_ (C++ *member*), 421  
 amici::ModelStateDerived::dxdotdx\_explicit (C++ *member*), 421  
 amici::ModelStateDerived::dxdotdx\_implicit (C++ *member*), 422  
 amici::ModelStateDerived::dydp\_ (C++ *member*), 423  
 amici::ModelStateDerived::dydx\_ (C++ *member*), 423  
 amici::ModelStateDerived::dzdp\_ (C++ *member*), 423  
 amici::ModelStateDerived::dzdx\_ (C++ *member*), 423  
 amici::ModelStateDerived::J\_ (C++ *member*), 421  
 amici::ModelStateDerived::JB\_ (C++ *member*), 421  
 amici::ModelStateDerived::M\_ (C++ *member*), 421  
 amici::ModelStateDerived::ModelStateDerived (C++ *function*), 421  
 amici::ModelStateDerived::MSparse\_ (C++ *member*), 422



ber), 421  
 amici::ModelStateDerived::rz\_ (C++ member), 424  
 amici::ModelStateDerived::sigmay\_ (C++ member), 424  
 amici::ModelStateDerived::sigmaz\_ (C++ member), 424  
 amici::ModelStateDerived::sspl\_ (C++ member), 424  
 amici::ModelStateDerived::sx\_ (C++ member), 423  
 amici::ModelStateDerived::sx\_rdata\_ (C++ member), 423  
 amici::ModelStateDerived::sy\_ (C++ member), 423  
 amici::ModelStateDerived::w\_ (C++ member), 423  
 amici::ModelStateDerived::x\_pos\_tmp\_ (C++ member), 424  
 amici::ModelStateDerived::x\_rdata\_ (C++ member), 423  
 amici::ModelStateDerived::y\_ (C++ member), 423  
 amici::ModelStateDerived::z\_ (C++ member), 424  
 amici::N\_VGetArrayPointerConst (C++ function), 663  
 amici::NewtonDampingFactorMode (C++ enum), 639  
 amici::NewtonDampingFactorMode::off (C++ enumerator), 639  
 amici::NewtonDampingFactorMode::on (C++ enumerator), 639  
 amici::NewtonFailure (C++ class), 562  
 amici::NewtonFailure::error\_code (C++ member), 562  
 amici::NewtonFailure::NewtonFailure (C++ function), 562  
 amici::NewtonSolver (C++ class), 563  
 amici::NewtonSolver::~~NewtonSolver (C++ function), 564  
 amici::NewtonSolver::computeNewtonSensis (C++ function), 563  
 amici::NewtonSolver::dxB\_ (C++ member), 564  
 amici::NewtonSolver::getSolver (C++ function), 564  
 amici::NewtonSolver::getStep (C++ function), 563  
 amici::NewtonSolver::is\_singular (C++ function), 564  
 amici::NewtonSolver::NewtonSolver (C++ function), 563  
 amici::NewtonSolver::prepareLinearSystem (C++ function), 563  
 amici::NewtonSolver::prepareLinearSystemB (C++ function), 563  
 amici::NewtonSolver::reinitialize (C++ function), 563  
 amici::NewtonSolver::solveLinearSystem (C++ function), 563  
 amici::NewtonSolver::xB\_ (C++ member), 564  
 amici::NewtonSolver::xB\_ (C++ member), 564  
 amici::NewtonSolver::xdot\_ (C++ member), 564  
 amici::NewtonSolverDense (C++ class), 565  
 amici::NewtonSolverDense::~~NewtonSolverDense (C++ function), 565  
 amici::NewtonSolverDense::is\_singular (C++ function), 565  
 amici::NewtonSolverDense::NewtonSolverDense (C++ function), 565  
 amici::NewtonSolverDense::operator= (C++ function), 565  
 amici::NewtonSolverDense::prepareLinearSystem (C++ function), 565  
 amici::NewtonSolverDense::prepareLinearSystemB (C++ function), 565  
 amici::NewtonSolverDense::reinitialize (C++ function), 565  
 amici::NewtonSolverDense::solveLinearSystem (C++ function), 565  
 amici::NewtonSolverSparse (C++ class), 566  
 amici::NewtonSolverSparse::~~NewtonSolverSparse (C++ function), 566  
 amici::NewtonSolverSparse::is\_singular (C++ function), 567  
 amici::NewtonSolverSparse::NewtonSolverSparse (C++ function), 566  
 amici::NewtonSolverSparse::operator= (C++ function), 566  
 amici::NewtonSolverSparse::prepareLinearSystem (C++ function), 566  
 amici::NewtonSolverSparse::prepareLinearSystemB (C++ function), 566  
 amici::NewtonSolverSparse::reinitialize (C++ function), 567  
 amici::NewtonSolverSparse::solveLinearSystem (C++ function), 566  
 amici::NonlinearSolverIteration (C++ enum), 639  
 amici::NonlinearSolverIteration::fixedpoint (C++ enumerator), 639  
 amici::NonlinearSolverIteration::functional (C++ enumerator), 639  
 amici::NonlinearSolverIteration::newton (C++ enumerator), 639  
 amici::ObservableScaling (C++ enum), 639  
 amici::ObservableScaling::lin (C++ enumerator), 639  
 amici::ObservableScaling::log (C++ enumerator), 639  
 amici::ObservableScaling::log10 (C++ enumerator), 639  
 amici::operator== (C++ function), 663, 664

`amici::ParameterScaling` (C++ *enum*), 640  
`amici::ParameterScaling::ln` (C++ *enumerator*), 640  
`amici::ParameterScaling::log10` (C++ *enumerator*), 640  
`amici::ParameterScaling::none` (C++ *enumerator*), 640  
`amici::pi` (C++ *member*), 679  
`amici::printfToString` (C++ *function*), 664  
`amici::RDataReporting` (C++ *enum*), 640  
`amici::RDataReporting::full` (C++ *enumerator*), 640  
`amici::RDataReporting::likelihood` (C++ *enumerator*), 640  
`amici::RDataReporting::residuals` (C++ *enumerator*), 640  
`amici::realtype` (C++ *type*), 683  
`amici::regexErrorToString` (C++ *function*), 665  
`amici::ReturnData` (C++ *class*), 567  
`amici::ReturnData::~~ReturnData` (C++ *function*), 568  
`amici::ReturnData::applyChainRuleFactorToSimulationResults` (C++ *function*), 576  
`amici::ReturnData::boost::serialization::serialize` (C++ *function*), 578  
`amici::ReturnData::chi2` (C++ *member*), 572  
`amici::ReturnData::computingFSA` (C++ *function*), 576  
`amici::ReturnData::cpu_time` (C++ *member*), 570  
`amici::ReturnData::cpu_time_total` (C++ *member*), 571  
`amici::ReturnData::cpu_timeB` (C++ *member*), 570  
`amici::ReturnData::dx_solver_` (C++ *member*), 577  
`amici::ReturnData::fchi2` (C++ *function*), 575  
`amici::ReturnData::fFIM` (C++ *function*), 576  
`amici::ReturnData::FIM` (C++ *member*), 570  
`amici::ReturnData::fres` (C++ *function*), 575  
`amici::ReturnData::fsres` (C++ *function*), 575  
`amici::ReturnData::getDataOutput` (C++ *function*), 576  
`amici::ReturnData::getDataSensisFSA` (C++ *function*), 576  
`amici::ReturnData::getEventOutput` (C++ *function*), 576  
`amici::ReturnData::getEventSensisFSA` (C++ *function*), 577  
`amici::ReturnData::handleSx0Backward` (C++ *function*), 577  
`amici::ReturnData::handleSx0Forward` (C++ *function*), 577  
`amici::ReturnData::id` (C++ *member*), 568  
`amici::ReturnData::initializeFullReporting` (C++ *function*), 574  
`amici::ReturnData::initializeLikelihoodReporting` (C++ *function*), 574  
`amici::ReturnData::initializeObjectiveFunction` (C++ *function*), 574  
`amici::ReturnData::initializeResidualReporting` (C++ *function*), 574  
`amici::ReturnData::invalidate` (C++ *function*), 576  
`amici::ReturnData::invalidateLLH` (C++ *function*), 576  
`amici::ReturnData::invalidateSLLH` (C++ *function*), 576  
`amici::ReturnData::J` (C++ *member*), 569  
`amici::ReturnData::llh` (C++ *member*), 572  
`amici::ReturnData::messages` (C++ *member*), 574  
`amici::ReturnData::newton_maxsteps` (C++ *member*), 573  
`amici::ReturnData::nmaxevent` (C++ *member*), 573  
`amici::ReturnData::nplist` (C++ *member*), 573  
`amici::ReturnData::nroots_` (C++ *member*), 578  
`amici::ReturnData::nt` (C++ *member*), 573  
`amici::ReturnData::numerrtestfails` (C++ *member*), 570  
`amici::ReturnData::numerrtestfailsB` (C++ *member*), 570  
`amici::ReturnData::numnonlinsolvconvfails` (C++ *member*), 570  
`amici::ReturnData::numnonlinsolvconvfailsB` (C++ *member*), 570  
`amici::ReturnData::numrhsevals` (C++ *member*), 570  
`amici::ReturnData::numrhsevalsB` (C++ *member*), 570  
`amici::ReturnData::numsteps` (C++ *member*), 570  
`amici::ReturnData::numstepsB` (C++ *member*), 570  
`amici::ReturnData::nx` (C++ *member*), 573  
`amici::ReturnData::nxtrue` (C++ *member*), 573  
`amici::ReturnData::o2mode` (C++ *member*), 573  
`amici::ReturnData::order` (C++ *member*), 570  
`amici::ReturnData::posteq_cpu_time` (C++ *member*), 571  
`amici::ReturnData::posteq_cpu_timeB` (C++ *member*), 571  
`amici::ReturnData::posteq_numsteps` (C++ *member*), 571  
`amici::ReturnData::posteq_numstepsB` (C++ *member*), 572  
`amici::ReturnData::posteq_status` (C++ *member*), 571  
`amici::ReturnData::posteq_t` (C++ *member*), 572  
`amici::ReturnData::posteq_wrms` (C++ *member*), 572  
`amici::ReturnData::preeq_cpu_time` (C++ *member*), 571

amici::ReturnData::preeq\_cpu\_timeB (C++ member), 571  
 amici::ReturnData::preeq\_numsteps (C++ member), 571  
 amici::ReturnData::preeq\_numstepsB (C++ member), 571  
 amici::ReturnData::preeq\_status (C++ member), 571  
 amici::ReturnData::preeq\_t (C++ member), 572  
 amici::ReturnData::preeq\_wrms (C++ member), 572  
 amici::ReturnData::processBackwardProblem (C++ function), 574  
 amici::ReturnData::processForwardProblem (C++ function), 574  
 amici::ReturnData::processPostEquilibration (C++ function), 574  
 amici::ReturnData::processPreEquilibration (C++ function), 574  
 amici::ReturnData::processSimulationObjects (C++ function), 568  
 amici::ReturnData::processSolver (C++ function), 575  
 amici::ReturnData::pscale (C++ member), 573  
 amici::ReturnData::rdata\_reporting (C++ member), 573  
 amici::ReturnData::readSimulationState (C++ function), 575  
 amici::ReturnData::res (C++ member), 570  
 amici::ReturnData::ReturnData (C++ function), 567, 568  
 amici::ReturnData::rz (C++ member), 569  
 amici::ReturnData::s2llh (C++ member), 572  
 amici::ReturnData::s2rz (C++ member), 569  
 amici::ReturnData::sensi (C++ member), 573  
 amici::ReturnData::sensi\_meth (C++ member), 573  
 amici::ReturnData::sigma\_offset (C++ member), 577  
 amici::ReturnData::sigma\_res (C++ member), 573  
 amici::ReturnData::sigmay (C++ member), 569  
 amici::ReturnData::sigmaz (C++ member), 569  
 amici::ReturnData::sllh (C++ member), 572  
 amici::ReturnData::sres (C++ member), 570  
 amici::ReturnData::srz (C++ member), 569  
 amici::ReturnData::ssigmay (C++ member), 569  
 amici::ReturnData::ssigmaz (C++ member), 569  
 amici::ReturnData::status (C++ member), 572  
 amici::ReturnData::storeJacobianAndDerivative (C++ function), 575  
 amici::ReturnData::sx (C++ member), 569  
 amici::ReturnData::sx0 (C++ member), 572  
 amici::ReturnData::sx\_rdata\_ (C++ member), 578  
 amici::ReturnData::sx\_solver\_ (C++ member), 577  
 amici::ReturnData::sx\_ss (C++ member), 572  
 amici::ReturnData::sy (C++ member), 569  
 amici::ReturnData::sz (C++ member), 569  
 amici::ReturnData::t\_ (C++ member), 577  
 amici::ReturnData::ts (C++ member), 568  
 amici::ReturnData::w (C++ member), 569  
 amici::ReturnData::x (C++ member), 569  
 amici::ReturnData::x0 (C++ member), 572  
 amici::ReturnData::x\_rdata\_ (C++ member), 578  
 amici::ReturnData::x\_solver\_ (C++ member), 577  
 amici::ReturnData::x\_ss (C++ member), 572  
 amici::ReturnData::xdot (C++ member), 568  
 amici::ReturnData::y (C++ member), 569  
 amici::ReturnData::z (C++ member), 569  
 amici::runAmiciSimulation (C++ function), 665  
 amici::runAmiciSimulations (C++ function), 665  
 amici::scaleParameters (C++ function), 666  
 amici::SecondOrderMode (C++ enum), 641  
 amici::SecondOrderMode::directional (C++ enumerator), 641  
 amici::SecondOrderMode::full (C++ enumerator), 641  
 amici::SecondOrderMode::none (C++ enumerator), 641  
 amici::SensitivityMethod (C++ enum), 641  
 amici::SensitivityMethod::adjoint (C++ enumerator), 641  
 amici::SensitivityMethod::forward (C++ enumerator), 641  
 amici::SensitivityMethod::none (C++ enumerator), 641  
 amici::SensitivityOrder (C++ enum), 642  
 amici::SensitivityOrder::first (C++ enumerator), 642  
 amici::SensitivityOrder::none (C++ enumerator), 642  
 amici::SensitivityOrder::second (C++ enumerator), 642  
 amici::serializeToChar (C++ function), 666  
 amici::serializeToStdVec (C++ function), 667  
 amici::serializeToString (C++ function), 667  
 amici::SetupFailure (C++ class), 578  
 amici::SetupFailure::SetupFailure (C++ function), 579  
 amici::simulation\_status\_to\_str (C++ function), 667  
 amici::SimulationParameters (C++ class), 579  
 amici::SimulationParameters::fixedParameters (C++ member), 580  
 amici::SimulationParameters::fixedParametersPreequilibration (C++ member), 580  
 amici::SimulationParameters::fixedParametersPresimulation (C++ member), 580

amici::SimulationParameters::parameters (C++ member), 580  
 amici::SimulationParameters::plist (C++ member), 581  
 amici::SimulationParameters::pscale (C++ member), 581  
 amici::SimulationParameters::reinitialization\_state (C++ member), 581  
 amici::SimulationParameters::reinitialization\_state\_sim (C++ member), 581  
 amici::SimulationParameters::reinitializeAllFixedParametersDependentInitialStates (C++ function), 580  
 amici::SimulationParameters::reinitializeAllFixedParametersDependentInitialStatesASA (C++ function), 580  
 amici::SimulationParameters::reinitializeAllFixedParametersDependentInitialStatesFSA (C++ function), 580  
 amici::SimulationParameters::reinitializeFixedParametersInitialStates (C++ member), 581  
 amici::SimulationParameters::SimulationParameters (C++ function), 579, 580  
 amici::SimulationParameters::sx0 (C++ member), 581  
 amici::SimulationParameters::t\_presim (C++ member), 581  
 amici::SimulationParameters::ts\_ (C++ member), 581  
 amici::SimulationParameters::tstart\_ (C++ member), 581  
 amici::SimulationParameters::x0 (C++ member), 581  
 amici::SimulationState (C++ struct), 425  
 amici::SimulationState::dx (C++ member), 425  
 amici::SimulationState::state (C++ member), 425  
 amici::SimulationState::sx (C++ member), 425  
 amici::SimulationState::t (C++ member), 425  
 amici::SimulationState::x (C++ member), 425  
 amici::slice (C++ function), 668  
 amici::Solver (C++ class), 582  
 amici::Solver::~~Solver (C++ function), 582  
 amici::Solver::adjInit (C++ function), 599  
 amici::Solver::allocateSolver (C++ function), 596  
 amici::Solver::allocateSolverB (C++ function), 599  
 amici::Solver::applyQuadTolerances (C++ function), 602  
 amici::Solver::applyQuadTolerancesASA (C++ function), 602  
 amici::Solver::applySensitivityTolerances (C++ function), 602  
 amici::Solver::applyTolerances (C++ function), 602  
 amici::Solver::applyTolerancesASA (C++ function), 602  
 amici::Solver::applyTolerancesFSA (C++ function), 602  
 amici::Solver::binit (C++ function), 595  
 amici::Solver::boost::serialization::serialize (C++ function), 605  
 amici::Solver::calcIC (C++ function), 584  
 amici::Solver::calcICB (C++ function), 584  
 amici::Solver::checkSensitivityMethod (C++ function), 603  
 amici::Solver::computeASA (C++ function), 592  
 amici::Solver::diag (C++ function), 602  
 amici::Solver::dky\_ (C++ member), 604  
 amici::Solver::dxB\_ (C++ member), 604  
 amici::Solver::dxB\_ (C++ member), 605  
 amici::Solver::force\_reinit\_postprocess\_B\_ (C++ member), 605  
 amici::Solver::force\_reinit\_postprocess\_F\_ (C++ member), 605  
 amici::Solver::getAbsoluteTolerance (C++ function), 585  
 amici::Solver::getAbsoluteToleranceB (C++ function), 586  
 amici::Solver::getAbsoluteToleranceFSA (C++ function), 586  
 amici::Solver::getAbsoluteToleranceQuadratures (C++ function), 586  
 amici::Solver::getAbsoluteToleranceSteadyState (C++ function), 587  
 amici::Solver::getAbsoluteToleranceSteadyStateSensi (C++ function), 588  
 amici::Solver::getAdjBmem (C++ function), 602  
 amici::Solver::getAdjInitDone (C++ function), 601  
 amici::Solver::getAdjointDerivativeState (C++ function), 591  
 amici::Solver::getAdjointQuadrature (C++ function), 591  
 amici::Solver::getAdjointState (C++ function), 591  
 amici::Solver::getB (C++ function), 595  
 amici::Solver::getCpuTime (C++ function), 592  
 amici::Solver::getCpuTimeB (C++ function), 592  
 amici::Solver::getDerivativeState (C++ function), 590  
 amici::Solver::getDky (C++ function), 598  
 amici::Solver::getDkyB (C++ function), 598  
 amici::Solver::getInitDone (C++ function), 601  
 amici::Solver::getInitDoneB (C++ function), 601  
 amici::Solver::getInternalSensitivityMethod (C++ function), 589

amici::Solver::getInterpolationType (C++ function), 589  
 amici::Solver::getLastOrder (C++ function), 593, 600  
 amici::Solver::getLinearMultistepMethod (C++ function), 588  
 amici::Solver::getLinearSolver (C++ function), 589  
 amici::Solver::getMaxSteps (C++ function), 588  
 amici::Solver::getMaxStepsBackwardProblem (C++ function), 588  
 amici::Solver::getMaxTime (C++ function), 588  
 amici::Solver::getModel (C++ function), 601  
 amici::Solver::getNewtonDampingFactorLowerBound (C++ function), 585  
 amici::Solver::getNewtonDampingFactorMode (C++ function), 585  
 amici::Solver::getNewtonMaxSteps (C++ function), 584  
 amici::Solver::getNewtonStepSteadyStateCheck (C++ function), 593  
 amici::Solver::getNonlinearSolverIteration (C++ function), 589  
 amici::Solver::getNumErrTestFails (C++ function), 593, 600  
 amici::Solver::getNumErrTestFailsB (C++ function), 593  
 amici::Solver::getNumNonlinSolvConvFails (C++ function), 593, 600  
 amici::Solver::getNumNonlinSolvConvFailsB (C++ function), 593  
 amici::Solver::getNumRhsEvals (C++ function), 593, 600  
 amici::Solver::getNumRhsEvalsB (C++ function), 593  
 amici::Solver::getNumSteps (C++ function), 593, 600  
 amici::Solver::getNumStepsB (C++ function), 593  
 amici::Solver::getQuad (C++ function), 595  
 amici::Solver::getQuadB (C++ function), 595  
 amici::Solver::getQuadDky (C++ function), 599  
 amici::Solver::getQuadDkyB (C++ function), 599  
 amici::Solver::getQuadInitDone (C++ function), 602  
 amici::Solver::getQuadInitDoneB (C++ function), 601  
 amici::Solver::getQuadrature (C++ function), 591  
 amici::Solver::getRelativeTolerance (C++ function), 585  
 amici::Solver::getRelativeToleranceB (C++ function), 586  
 amici::Solver::getRelativeToleranceFSA (C++ function), 586  
 amici::Solver::getRelativeToleranceQuadratures (C++ function), 586  
 amici::Solver::getRelativeToleranceSteadyState (C++ function), 587  
 amici::Solver::getRelativeToleranceSteadyStateSensi (C++ function), 587  
 amici::Solver::getReturnDataReportingMode (C++ function), 590  
 amici::Solver::getRootInfo (C++ function), 584  
 amici::Solver::getSens (C++ function), 595  
 amici::Solver::getSensDky (C++ function), 599  
 amici::Solver::getSensInitDone (C++ function), 601  
 amici::Solver::getSensiSteadyStateCheck (C++ function), 593  
 amici::Solver::getSensitivityMethod (C++ function), 584  
 amici::Solver::getSensitivityMethodPreequilibration (C++ function), 584  
 amici::Solver::getSensitivityOrder (C++ function), 585  
 amici::Solver::getStabilityLimitFlag (C++ function), 589  
 amici::Solver::getState (C++ function), 590  
 amici::Solver::getStateOrdering (C++ function), 589  
 amici::Solver::getStateSensitivity (C++ function), 590  
 amici::Solver::getSteadyStateSensiToleranceFactor (C++ function), 587  
 amici::Solver::getSteadyStateToleranceFactor (C++ function), 587  
 amici::Solver::gett (C++ function), 592  
 amici::Solver::init (C++ function), 595  
 amici::Solver::initializeLinearSolver (C++ function), 600  
 amici::Solver::initializeLinearSolverB (C++ function), 601  
 amici::Solver::initializeNonLinearSolver (C++ function), 600  
 amici::Solver::initializeNonLinearSolverB (C++ function), 601  
 amici::Solver::initializeNonLinearSolverSens (C++ function), 596  
 amici::Solver::initSteadystate (C++ function), 595  
 amici::Solver::interp\_type\_ (C++ member), 604  
 amici::Solver::ism\_ (C++ member), 603  
 amici::Solver::iter\_ (C++ member), 603  
 amici::Solver::linear\_solver\_ (C++ member), 604  
 amici::Solver::linear\_solver\_B\_ (C++ member), 604  
 amici::Solver::lmm\_ (C++ member), 603  
 amici::Solver::logger (C++ member), 594



amici::Solver::maxsteps\_ (C++ member), 604  
 amici::Solver::maxtime\_ (C++ member), 604  
 amici::Solver::non\_linear\_solver\_ (C++ member), 604  
 amici::Solver::non\_linear\_solver\_B\_ (C++ member), 604  
 amici::Solver::non\_linear\_solver\_sens\_ (C++ member), 604  
 amici::Solver::nplist (C++ function), 592  
 amici::Solver::nquad (C++ function), 592  
 amici::Solver::nx (C++ function), 592  
 amici::Solver::operator== (C++ function), 605  
 amici::Solver::qbinit (C++ function), 596  
 amici::Solver::quadInit (C++ function), 599  
 amici::Solver::quadReInitB (C++ function), 592  
 amici::Solver::quadSStolerances (C++ function), 600  
 amici::Solver::quadSStolerancesB (C++ function), 599  
 amici::Solver::reInit (C++ function), 591  
 amici::Solver::reInitB (C++ function), 592  
 amici::Solver::reInitPostProcessB (C++ function), 594  
 amici::Solver::reInitPostProcessF (C++ function), 594  
 amici::Solver::resetDiagnosis (C++ function), 592  
 amici::Solver::resetMutableMemory (C++ function), 602  
 amici::Solver::rootInit (C++ function), 596  
 amici::Solver::run (C++ function), 582  
 amici::Solver::runB (C++ function), 583  
 amici::Solver::sdx\_ (C++ member), 605  
 amici::Solver::sens\_initialized\_ (C++ member), 605  
 amici::Solver::sensInit1 (C++ function), 595  
 amici::Solver::sensReInit (C++ function), 591  
 amici::Solver::sensToggleOff (C++ function), 591  
 amici::Solver::setAbsoluteTolerance (C++ function), 585  
 amici::Solver::setAbsoluteToleranceB (C++ function), 586  
 amici::Solver::setAbsoluteToleranceFSA (C++ function), 586  
 amici::Solver::setAbsoluteToleranceQuadratures (C++ function), 586  
 amici::Solver::setAbsoluteToleranceSteadyState (C++ function), 587  
 amici::Solver::setAbsoluteToleranceSteadyStateSensi (C++ function), 588  
 amici::Solver::setAdjInitDone (C++ function), 603  
 amici::Solver::setBandJacFn (C++ function), 596  
 amici::Solver::setBandJacFnB (C++ function), 596  
 amici::Solver::setDenseJacFn (C++ function), 596  
 amici::Solver::setDenseJacFnB (C++ function), 596  
 amici::Solver::setErrHandlerFn (C++ function), 597  
 amici::Solver::setId (C++ function), 598  
 amici::Solver::setInitDone (C++ function), 603  
 amici::Solver::setInitDoneB (C++ function), 603  
 amici::Solver::setInternalSensitivityMethod (C++ function), 590  
 amici::Solver::setInterpolationType (C++ function), 589  
 amici::Solver::setJacTimesVecFn (C++ function), 596  
 amici::Solver::setJacTimesVecFnB (C++ function), 596  
 amici::Solver::setLinearMultistepMethod (C++ function), 589  
 amici::Solver::setLinearSolver (C++ function), 589, 601  
 amici::Solver::setLinearSolverB (C++ function), 601  
 amici::Solver::setMaxNumSteps (C++ function), 597  
 amici::Solver::setMaxNumStepsB (C++ function), 597  
 amici::Solver::setMaxSteps (C++ function), 588  
 amici::Solver::setMaxStepsBackwardProblem (C++ function), 588  
 amici::Solver::setMaxTime (C++ function), 588  
 amici::Solver::setNewtonDampingFactorLowerBound (C++ function), 585  
 amici::Solver::setNewtonDampingFactorMode (C++ function), 585  
 amici::Solver::setNewtonMaxSteps (C++ function), 585  
 amici::Solver::setNewtonStepSteadyStateCheck (C++ function), 594  
 amici::Solver::setNonLinearSolver (C++ function), 601  
 amici::Solver::setNonLinearSolverB (C++ function), 601  
 amici::Solver::setNonlinearSolverIteration (C++ function), 589  
 amici::Solver::setNonLinearSolverSens (C++ function), 601  
 amici::Solver::setQuadErrCon (C++ function), 597  
 amici::Solver::setQuadErrConB (C++ function), 597  
 amici::Solver::setQuadInitDone (C++ function), 603  
 amici::Solver::setQuadInitDoneB (C++ function), 603  
 amici::Solver::setRelativeTolerance (C++

[function\), 585](#)  
[amici::Solver::setRelativeToleranceB \(C++ function\), 586](#)  
[amici::Solver::setRelativeToleranceFSA \(C++ function\), 586](#)  
[amici::Solver::setRelativeToleranceQuadratures \(C++ function\), 586](#)  
[amici::Solver::setRelativeToleranceSteadyState \(C++ function\), 587](#)  
[amici::Solver::setRelativeToleranceSteadyStateSensi \(C++ function\), 587](#)  
[amici::Solver::setReturnDataReportingMode \(C++ function\), 590](#)  
[amici::Solver::setSensErrCon \(C++ function\), 597](#)  
[amici::Solver::setSensInitDone \(C++ function\), 603](#)  
[amici::Solver::setSensInitOff \(C++ function\), 603](#)  
[amici::Solver::setSensiSteadyStateCheck \(C++ function\), 594](#)  
[amici::Solver::setSensitivityMethod \(C++ function\), 584](#)  
[amici::Solver::setSensitivityMethodPreequilibration \(C++ function\), 584](#)  
[amici::Solver::setSensitivityOrder \(C++ function\), 585](#)  
[amici::Solver::setSensParams \(C++ function\), 598](#)  
[amici::Solver::setSensSStolerances \(C++ function\), 597](#)  
[amici::Solver::setSparseJacFn \(C++ function\), 596](#)  
[amici::Solver::setSparseJacFn\\_ss \(C++ function\), 596](#)  
[amici::Solver::setSparseJacFnB \(C++ function\), 596](#)  
[amici::Solver::setSStolerances \(C++ function\), 596](#)  
[amici::Solver::setSStolerancesB \(C++ function\), 599](#)  
[amici::Solver::setStabilityLimitFlag \(C++ function\), 589](#)  
[amici::Solver::setStabLimDet \(C++ function\), 598](#)  
[amici::Solver::setStabLimDetB \(C++ function\), 598](#)  
[amici::Solver::setStateOrdering \(C++ function\), 589](#)  
[amici::Solver::setSteadyStateSensiToleranceFactor \(C++ function\), 587](#)  
[amici::Solver::setSteadyStateToleranceFactor \(C++ function\), 587](#)  
[amici::Solver::setStopTime \(C++ function\), 594](#)  
[amici::Solver::setSuppressAlg \(C++ function\), 598](#)  
[amici::Solver::setup \(C++ function\), 583](#)  
[amici::Solver::setupB \(C++ function\), 583](#)  
[amici::Solver::setupSteadystate \(C++ function\), 583](#)  
[amici::Solver::setUserData \(C++ function\), 597](#)  
[amici::Solver::setUserDataB \(C++ function\), 597](#)  
[amici::Solver::simulation\\_timer\\_ \(C++ member\), 604](#)  
[amici::Solver::solve \(C++ function\), 594](#)  
[amici::Solver::solveB \(C++ function\), 584](#)  
[amici::Solver::solveF \(C++ function\), 594](#)  
[amici::Solver::Solver \(C++ function\), 582](#)  
[amici::Solver::solver\\_memory\\_ \(C++ member\), 603](#)  
[amici::Solver::solver\\_memory\\_B\\_ \(C++ member\), 603](#)  
[amici::Solver::solver\\_was\\_called\\_B\\_ \(C++ member\), 604](#)  
[amici::Solver::solver\\_was\\_called\\_F\\_ \(C++ member\), 604](#)  
[amici::Solver::startTimer \(C++ function\), 588](#)  
[amici::Solver::step \(C++ function\), 583](#)  
[amici::Solver::storeDiagnosis \(C++ function\), 593](#)  
[amici::Solver::storeDiagnosisB \(C++ function\), 593](#)  
[amici::Solver::switchForwardSensisOff \(C++ function\), 584](#)  
[amici::Solver::sx\\_ \(C++ member\), 604](#)  
[amici::Solver::t\\_ \(C++ member\), 605](#)  
[amici::Solver::timeExceeded \(C++ function\), 588](#)  
[amici::Solver::turnOffRootFinding \(C++ function\), 584](#)  
[amici::Solver::updateAndReinitStatesAndSensitivities \(C++ function\), 584](#)  
[amici::Solver::user\\_data \(C++ member\), 603](#)  
[amici::Solver::user\\_data\\_type \(C++ type\), 582](#)  
[amici::Solver::writeSolution \(C++ function\), 590](#)  
[amici::Solver::writeSolutionB \(C++ function\), 590](#)  
[amici::Solver::x\\_ \(C++ member\), 604](#)  
[amici::Solver::xB\\_ \(C++ member\), 605](#)  
[amici::Solver::xQ\\_ \(C++ member\), 605](#)  
[amici::Solver::xQB\\_ \(C++ member\), 605](#)  
[amici::SplineBoundaryCondition \(C++ enum\), 642](#)  
[amici::SplineBoundaryCondition::given \(C++ enumerator\), 642](#)  
[amici::SplineBoundaryCondition::natural \(C++ enumerator\), 642](#)  
[amici::SplineBoundaryCondition::naturalZeroDerivative \(C++ enumerator\), 642](#)  
[amici::SplineBoundaryCondition::periodic \(C++ enumerator\), 642](#)  
[amici::SplineBoundaryCondition::zeroDerivative \(C++ enumerator\), 642](#)

amici::SplineExtrapolation (C++ *enum*), 643  
 amici::SplineExtrapolation::constant (C++ *enumerator*), 643  
 amici::SplineExtrapolation::linear (C++ *enumerator*), 643  
 amici::SplineExtrapolation::noExtrapolation (C++ *enumerator*), 643  
 amici::SplineExtrapolation::periodic (C++ *enumerator*), 643  
 amici::SplineExtrapolation::polynomial (C++ *enumerator*), 643  
 amici::SteadyStateComputationMode (C++ *enum*), 643  
 amici::SteadyStateComputationMode::integrateIfNewtonFails (C++ *enumerator*), 643  
 amici::SteadyStateComputationMode::integrationOnly (C++ *enumerator*), 643  
 amici::SteadyStateComputationMode::newtonOnly (C++ *enumerator*), 643  
 amici::SteadyStateContext (C++ *enum*), 644  
 amici::SteadyStateContext::newtonSensi (C++ *enumerator*), 644  
 amici::SteadyStateContext::sensiStorage (C++ *enumerator*), 644  
 amici::SteadyStateContext::solverCreation (C++ *enumerator*), 644  
 amici::SteadystateProblem (C++ *class*), 606  
 amici::SteadystateProblem::checkSteadyStateSuccess (C++ *function*), 608  
 amici::SteadystateProblem::getAdjointQuadrature (C++ *function*), 608  
 amici::SteadystateProblem::getAdjointState (C++ *function*), 607  
 amici::SteadystateProblem::getAdjointUpdates (C++ *function*), 607  
 amici::SteadystateProblem::getCPUTime (C++ *function*), 607  
 amici::SteadystateProblem::getCPUTimeB (C++ *function*), 607  
 amici::SteadystateProblem::getDJydx (C++ *function*), 607  
 amici::SteadystateProblem::getEquilibrationQuadratures (C++ *function*), 606  
 amici::SteadystateProblem::getFinalSimulationState (C++ *function*), 606  
 amici::SteadystateProblem::getNumSteps (C++ *function*), 607  
 amici::SteadystateProblem::getNumStepsB (C++ *function*), 607  
 amici::SteadystateProblem::getResidualNorm (C++ *function*), 607  
 amici::SteadystateProblem::getState (C++ *function*), 607  
 amici::SteadystateProblem::getStateSensitivity (C++ *function*), 607  
 amici::SteadystateProblem::getSteadyStateStatus (C++ *function*), 607  
 amici::SteadystateProblem::getSteadyStateTime (C++ *function*), 607  
 amici::SteadystateProblem::hasQuadrature (C++ *function*), 608  
 amici::SteadystateProblem::SteadystateProblem (C++ *function*), 606  
 amici::SteadystateProblem::workSteadyStateBackwardProblem (C++ *function*), 606  
 amici::SteadystateProblem::workSteadyStateProblem (C++ *function*), 606  
 amici::SteadystateProblem::workSteadyStateSensitivityMode (C++ *enum*), 644  
 amici::SteadyStateSensitivityMode::integrateIfNewtonFails (C++ *enumerator*), 644  
 amici::SteadyStateSensitivityMode::integrationOnly (C++ *enumerator*), 644  
 amici::SteadyStateSensitivityMode::newtonOnly (C++ *enumerator*), 644  
 amici::SteadyStateStatus (C++ *enum*), 645  
 amici::SteadyStateStatus::failed (C++ *enumerator*), 645  
 amici::SteadyStateStatus::failed\_convergence (C++ *enumerator*), 645  
 amici::SteadyStateStatus::failed\_damping (C++ *enumerator*), 645  
 amici::SteadyStateStatus::failed\_factorization (C++ *enumerator*), 645  
 amici::SteadyStateStatus::failed\_too\_long\_simulation (C++ *enumerator*), 645  
 amici::SteadyStateStatus::not\_run (C++ *enumerator*), 645  
 amici::SteadyStateStatus::success (C++ *enumerator*), 645  
 amici::SUNLinSolBand (C++ *class*), 608  
 amici::SUNLinSolBand::getMatrix (C++ *function*), 609  
 amici::SUNLinSolBand::SUNLinSolBand (C++ *function*), 608  
 amici::SUNLinSolDense (C++ *class*), 609  
 amici::SUNLinSolDense::getMatrix (C++ *function*), 609  
 amici::SUNLinSolDense::SUNLinSolDense (C++ *function*), 609  
 amici::SUNLinSolKLU (C++ *class*), 610  
 amici::SUNLinSolKLU::getMatrix (C++ *function*), 610  
 amici::SUNLinSolKLU::reInit (C++ *function*), 610  
 amici::SUNLinSolKLU::setOrdering (C++ *function*), 611  
 amici::SUNLinSolKLU::StateOrdering (C++ *enum*), 610



---

```

amici::SUNLinSolKLU::StateOrdering::AMD
    (C++ enumerator), 610
amici::SUNLinSolKLU::StateOrdering::COLAMD
    (C++ enumerator), 610
amici::SUNLinSolKLU::StateOrdering::natural
    (C++ enumerator), 610
amici::SUNLinSolKLU::SUNLinSolKLU (C++ function), 610
amici::SUNLinSolPCG (C++ class), 611
amici::SUNLinSolPCG::getNumIters (C++ function), 612
amici::SUNLinSolPCG::getResid (C++ function), 612
amici::SUNLinSolPCG::getResNorm (C++ function), 612
amici::SUNLinSolPCG::setATimes (C++ function), 611
amici::SUNLinSolPCG::setPreconditioner (C++ function), 611
amici::SUNLinSolPCG::setScalingVectors (C++ function), 612
amici::SUNLinSolPCG::SUNLinSolPCG (C++ function), 611
amici::SUNLinSolSPBCGS (C++ class), 612
amici::SUNLinSolSPBCGS::getNumIters (C++ function), 613
amici::SUNLinSolSPBCGS::getResid (C++ function), 614
amici::SUNLinSolSPBCGS::getResNorm (C++ function), 614
amici::SUNLinSolSPBCGS::setATimes (C++ function), 613
amici::SUNLinSolSPBCGS::setPreconditioner
    (C++ function), 613
amici::SUNLinSolSPBCGS::setScalingVectors
    (C++ function), 613
amici::SUNLinSolSPBCGS::SUNLinSolSPBCGS
    (C++ function), 613
amici::SUNLinSolSPFGMR (C++ class), 614
amici::SUNLinSolSPFGMR::getNumIters (C++ function), 615
amici::SUNLinSolSPFGMR::getResid (C++ function), 615
amici::SUNLinSolSPFGMR::getResNorm (C++ function), 615
amici::SUNLinSolSPFGMR::setATimes (C++ function), 614
amici::SUNLinSolSPFGMR::setPreconditioner
    (C++ function), 614
amici::SUNLinSolSPFGMR::setScalingVectors
    (C++ function), 615
amici::SUNLinSolSPFGMR::SUNLinSolSPFGMR
    (C++ function), 614
amici::SUNLinSolSPGMR (C++ class), 616
amici::SUNLinSolSPGMR::getNumIters (C++ function), 616
amici::SUNLinSolSPGMR::getResid (C++ function), 617
amici::SUNLinSolSPGMR::getResNorm (C++ function), 616
amici::SUNLinSolSPGMR::setATimes (C++ function), 616
amici::SUNLinSolSPGMR::setPreconditioner
    (C++ function), 616
amici::SUNLinSolSPGMR::setScalingVectors
    (C++ function), 616
amici::SUNLinSolSPGMR::SUNLinSolSPGMR (C++ function), 616
amici::SUNLinSolSPTFQMR (C++ class), 617
amici::SUNLinSolSPTFQMR::getNumIters (C++ function), 618
amici::SUNLinSolSPTFQMR::getResid (C++ function), 618
amici::SUNLinSolSPTFQMR::getResNorm (C++ function), 618
amici::SUNLinSolSPTFQMR::setATimes (C++ function), 617
amici::SUNLinSolSPTFQMR::setPreconditioner
    (C++ function), 618
amici::SUNLinSolSPTFQMR::setScalingVectors
    (C++ function), 618
amici::SUNLinSolSPTFQMR::SUNLinSolSPTFQMR
    (C++ function), 617
amici::SUNLinSolWrapper (C++ class), 619
amici::SUNLinSolWrapper::~SUNLinSolWrapper
    (C++ function), 619
amici::SUNLinSolWrapper::get (C++ function), 620
amici::SUNLinSolWrapper::getLastFlag (C++ function), 620
amici::SUNLinSolWrapper::getMatrix (C++ function), 620
amici::SUNLinSolWrapper::getType (C++ function), 620
amici::SUNLinSolWrapper::initialize (C++ function), 621
amici::SUNLinSolWrapper::operator= (C++ function), 619
amici::SUNLinSolWrapper::setup (C++ function), 620
amici::SUNLinSolWrapper::Solve (C++ function), 620
amici::SUNLinSolWrapper::solver_ (C++ member), 621
amici::SUNLinSolWrapper::space (C++ function), 620
amici::SUNLinSolWrapper::SUNLinSolWrapper
    (C++ function), 619
amici::SUNMatrixWrapper (C++ class), 621

```

amici::SUNMatrixWrapper::~~SUNMatrixWrapper  
     (C++ function), 622  
 amici::SUNMatrixWrapper::capacity (C++ function), 623  
 amici::SUNMatrixWrapper::columns (C++ function), 623  
 amici::SUNMatrixWrapper::data (C++ function), 623  
 amici::SUNMatrixWrapper::get (C++ function), 622  
 amici::SUNMatrixWrapper::get\_data (C++ function), 623  
 amici::SUNMatrixWrapper::get\_indexptr (C++ function), 624  
 amici::SUNMatrixWrapper::get\_indexval (C++ function), 624  
 amici::SUNMatrixWrapper::matrix\_id (C++ function), 627  
 amici::SUNMatrixWrapper::multiply (C++ function), 625  
 amici::SUNMatrixWrapper::num\_indexptrs (C++ function), 623  
 amici::SUNMatrixWrapper::num\_nonzeros (C++ function), 623  
 amici::SUNMatrixWrapper::operator= (C++ function), 622  
 amici::SUNMatrixWrapper::realloc (C++ function), 622  
 amici::SUNMatrixWrapper::reallocate (C++ function), 622  
 amici::SUNMatrixWrapper::refresh (C++ function), 627  
 amici::SUNMatrixWrapper::rows (C++ function), 623  
 amici::SUNMatrixWrapper::scale (C++ function), 625  
 amici::SUNMatrixWrapper::scatter (C++ function), 626  
 amici::SUNMatrixWrapper::set\_data (C++ function), 623, 624  
 amici::SUNMatrixWrapper::set\_indexptr (C++ function), 624  
 amici::SUNMatrixWrapper::set\_indexptrs (C++ function), 624  
 amici::SUNMatrixWrapper::set\_indexval (C++ function), 624  
 amici::SUNMatrixWrapper::set\_indexvals (C++ function), 624  
 amici::SUNMatrixWrapper::sparse\_add (C++ function), 626  
 amici::SUNMatrixWrapper::sparse\_multiply (C++ function), 626  
 amici::SUNMatrixWrapper::sparse\_sum (C++ function), 626  
 amici::SUNMatrixWrapper::sparsetype (C++ function), 624  
 amici::SUNMatrixWrapper::SUNMatrixWrapper (C++ function), 621, 622  
 amici::SUNMatrixWrapper::to\_dense (C++ function), 627  
 amici::SUNMatrixWrapper::to\_diag (C++ function), 627  
 amici::SUNMatrixWrapper::transpose (C++ function), 627  
 amici::SUNMatrixWrapper::zero (C++ function), 627  
 amici::SUNNonLinSolFixedPoint (C++ class), 628  
 amici::SUNNonLinSolFixedPoint::getSysFn (C++ function), 628  
 amici::SUNNonLinSolFixedPoint::SUNNonLinSolFixedPoint (C++ function), 628  
 amici::SUNNonLinSolNewton (C++ class), 629  
 amici::SUNNonLinSolNewton::getSysFn (C++ function), 629  
 amici::SUNNonLinSolNewton::SUNNonLinSolNewton (C++ function), 629  
 amici::SUNNonLinSolWrapper (C++ class), 630  
 amici::SUNNonLinSolWrapper::~~SUNNonLinSolWrapper (C++ function), 630  
 amici::SUNNonLinSolWrapper::get (C++ function), 630  
 amici::SUNNonLinSolWrapper::getCurIter (C++ function), 631  
 amici::SUNNonLinSolWrapper::getNumConvFails (C++ function), 632  
 amici::SUNNonLinSolWrapper::getNumIters (C++ function), 631  
 amici::SUNNonLinSolWrapper::getType (C++ function), 630  
 amici::SUNNonLinSolWrapper::initialize (C++ function), 632  
 amici::SUNNonLinSolWrapper::operator= (C++ function), 630  
 amici::SUNNonLinSolWrapper::setConvTestFn (C++ function), 631  
 amici::SUNNonLinSolWrapper::setLSetupFn (C++ function), 631  
 amici::SUNNonLinSolWrapper::setLSolveFn (C++ function), 631  
 amici::SUNNonLinSolWrapper::setMaxIters (C++ function), 631  
 amici::SUNNonLinSolWrapper::setSysFn (C++ function), 631  
 amici::SUNNonLinSolWrapper::setup (C++ function), 630  
 amici::SUNNonLinSolWrapper::Solve (C++ function), 630  
 amici::SUNNonLinSolWrapper::solver (C++ member), 632

amici::SUNNonLinSolWrapper::SUNNonLinSolWrapper (C++ function), 630  
 amici::unravel\_index (C++ function), 668, 669  
 amici::unscaleParameters (C++ function), 669  
 amici::wrapErrHandlerFn (C++ function), 669  
 amici::writeSlice (C++ function), 670, 671  
 amici\_annotation (amici.splines.AbstractSpline property), 397  
 amici\_annotation (amici.splines.CubicHermiteSpline property), 401  
 AMICI\_H5\_RESTORE\_ERROR\_HANDLER (C macro), 679  
 AMICI\_H5\_SAVE\_ERROR\_HANDLER (C macro), 679  
 amici\_path (in module amici), 265  
 amici\_to\_petab\_scale() (in module amici.parameter\_mapping), 385  
 AMICI\_VERSION (C macro), 680  
 amiciModulePath (in module amici), 265  
 amiciSrcPath (in module amici), 265  
 amiciSwigPath (in module amici), 265  
 AmiciVersionError, 264  
 amievent (built-in class), 697  
 amifun (built-in class), 698  
 append() (amici.parameter\_mapping.ParameterMapping method), 384  
 apply\_patch() (amici.sbml\_utils.MathML\_SbmlPrinter method), 392  
 apply\_template() (in module amici.de\_export), 360  
 args (amici.AmiciVersionError attribute), 264  
 args (amici.import\_utils.CircularDependencyError attribute), 346  
 args (amici.import\_utils.SBMLException attribute), 346  
 args (amici.sbml\_utils.SbmlAnnotationError attribute), 392  
 args (amici.sbml\_utils.SbmlDuplicateComponentIdError attribute), 392  
 args (amici.sbml\_utils.SbmlInvalidIdSyntax attribute), 392  
 args (amici.sbml\_utils.SbmlMathError attribute), 392  
 args (amici.sbml\_utils.SbmlMissingComponentIdError attribute), 393  
 assignmentRules2observables() (in module amici.sbml\_import), 330

## B

band (amici.amici.LinearSolver attribute), 279  
 bc (amici.splines.AbstractSpline property), 398  
 bc (amici.splines.CubicHermiteSpline property), 401  
 BDF (amici.amici.LinearMultistepMethod attribute), 278  
 BNGL, 55  
 bngl2amici() (in module amici.bngl\_import), 334  
 BoolVector (class in amici.amici), 268  
 boost::serialization::archiveVector (C++ function), 671

boost::serialization::serialize (C++ function), 671, 672  
 braceidpattern (amici.de\_export.TemplateAmici attribute), 360  
 by\_id() (amici.numpy.ReturnDataView method), 389

## C

cast\_to\_sym() (in module amici.import\_utils), 346  
 check\_derivatives() (in module amici.gradient\_check), 383  
 check\_event\_support() (amici.sbml\_import.SbmlImporter method), 328  
 check\_finite\_difference() (in module amici.gradient\_check), 383  
 check\_if\_valid() (amici.splines.AbstractSpline method), 398  
 check\_if\_valid() (amici.splines.CubicHermiteSpline method), 401  
 check\_model() (in module amici.petab\_import), 334  
 check\_support() (amici.sbml\_import.SbmlImporter method), 328  
 chi2 (amici.amici.ReturnData property), 301  
 chi2 (amici.amici.ReturnDataPtr property), 307  
 CircularDependencyError, 346  
 clone() (amici.amici.Model method), 281  
 clone() (amici.amici.Solver method), 316  
 colptrs() (amici.de\_export.DEModel method), 354  
 compile\_model() (amici.de\_export.DEEExporter method), 352  
 compiledWithOpenMP() (in module amici.amici), 324  
 compute\_moiety\_conservation\_laws() (in module amici.conservated\_quantities\_demartino), 387  
 computingASA() (amici.amici.Solver method), 316  
 computingFSA() (amici.amici.Solver method), 316  
 conservation\_law\_has\_multispecies() (amici.de\_export.DEModel method), 354  
 conservation\_laws() (amici.de\_export.DEModel method), 354  
 ConservationLaw (class in amici.de\_model), 365  
 Constant (class in amici.de\_model), 366  
 constants() (amici.de\_export.DEModel method), 355  
 count() (amici.parameter\_mapping.ParameterMapping method), 384  
 count() (amici.splines.UniformGrid method), 404  
 cpu\_time (amici.amici.ReturnData property), 301  
 cpu\_time (amici.amici.ReturnDataPtr property), 307  
 cpu\_time\_total (amici.amici.ReturnData property), 301  
 cpu\_time\_total (amici.amici.ReturnDataPtr property), 307  
 cpu\_timeB (amici.amici.ReturnData property), 301  
 cpu\_timeB (amici.amici.ReturnDataPtr property), 307  
 CpuTimer (class in amici.amici), 268

`create_edata_for_condition()` (in module `amici.petab_objective`), 339  
`create_edatas()` (in module `amici.petab_objective`), 340  
`create_parameter_mapping()` (in module `amici.petab_objective`), 340  
`create_parameter_mapping_for_condition()` (in module `amici.petab_objective`), 341  
`create_parameterized_edatas()` (in module `amici.petab_objective`), 341  
`create_sbml_model()` (in module `amici.sbml_utils`), 394  
`CubicHermiteSpline` (class in `amici.splines`), 400  
`CVODES`, 55

## D

`d_scaled()` (`amici.splines.CubicHermiteSpline` method), 401  
`DAE`, 55  
`DEExporter` (class in `amici.de_export`), 351  
`delimiter` (`amici.de_export.TemplateAmici` attribute), 360  
`DEModel` (class in `amici.de_export`), 352  
`dense` (`amici.amici.LinearSolver` attribute), 279  
`derivative()` (`amici.splines.AbstractSpline` method), 398  
`derivative()` (`amici.splines.CubicHermiteSpline` method), 401  
`derivatives_at_nodes` (`amici.splines.CubicHermiteSpline` property), 401  
`derivatives_by_fd` (`amici.splines.CubicHermiteSpline` property), 401  
`diag` (`amici.amici.LinearSolver` attribute), 279  
`differential_states()` (`amici.de_export.DEModel` method), 355  
`DifferentialState` (class in `amici.de_model`), 366  
`directional` (`amici.amici.SecondOrderMode` attribute), 313  
`doprint()` (`amici.sbml_utils.MathMLSBmlPrinter` method), 392  
`DoubleVector` (class in `amici.amici`), 268

## E

`elapsed_milliseconds()` (`amici.amici.CpuTimer` method), 268  
`elapsed_seconds()` (`amici.amici.CpuTimer` method), 268  
`emptyPrinter()` (`amici.sbml_utils.MathMLSBmlPrinter` method), 392  
`enum()` (in module `amici.amici`), 324  
`eq()` (`amici.de_export.DEModel` method), 355

`evaluate()` (`amici.splines.AbstractSpline` method), 398  
`evaluate()` (`amici.splines.CubicHermiteSpline` method), 401  
`evaluate()` (in module `amici.numpy`), 390  
`evaluate_at` (`amici.splines.AbstractSpline` property), 398  
`evaluate_at` (`amici.splines.CubicHermiteSpline` property), 401  
`Event` (class in `amici.de_model`), 368  
`event_observables()` (`amici.de_export.DEModel` method), 355  
`EventObservable` (class in `amici.de_model`), 369  
`events()` (`amici.de_export.DEModel` method), 355  
`ExpData` (class in `amici.amici`), 268  
`ExpDataPtr` (class in `amici.amici`), 277  
`ExpDataPtrVector` (class in `amici.amici`), 278  
`ExpDataView` (class in `amici.numpy`), 389  
`Expression` (class in `amici.de_model`), 370  
`expressions()` (`amici.de_export.DEModel` method), 355  
`extract_monomers()` (in module `amici.pysb_import`), 331  
`extrapolate` (`amici.splines.AbstractSpline` property), 398  
`extrapolate` (`amici.splines.CubicHermiteSpline` property), 401  
`extrapolation_formulas` (`amici.splines.AbstractSpline` property), 398  
`extrapolation_formulas` (`amici.splines.CubicHermiteSpline` property), 401

## F

`failed` (`amici.amici.SteadyStateStatus` attribute), 324  
`failed_convergence` (`amici.amici.SteadyStateStatus` attribute), 324  
`failed_damping` (`amici.amici.SteadyStateStatus` attribute), 324  
`failed_factorization` (`amici.amici.SteadyStateStatus` attribute), 324  
`failed_too_long_simulation` (`amici.amici.SteadyStateStatus` attribute), 324  
`fdsigmaydy()` (`amici.amici.Model` method), 281  
`fdspline_slopesdp()` (`amici.amici.Model` method), 281  
`fdspline_valuesdp()` (`amici.amici.Model` method), 281  
`fdtotal_cldp()` (`amici.amici.Model` method), 281  
`fdtotal_cldx_rdata()` (`amici.amici.Model` method), 282  
`fdx_rdatadp()` (`amici.amici.Model` method), 282  
`fdx_rdatadtcl()` (`amici.amici.Model` method), 282  
`fdx_rdatadx_solver()` (`amici.amici.Model` method), 282



fill\_in\_parameters() (in module *amici.parameter\_mapping*), 385  
 fill\_in\_parameters\_for\_condition() (in module *amici.parameter\_mapping*), 386  
 FIM (*amici.amici.ReturnData* property), 300  
 FIM (*amici.amici.ReturnDataPtr* property), 307  
 first (*amici.amici.SensitivityOrder* attribute), 313  
 fixed parameters, 55  
 FixedParameterContext (class in *amici.amici*), 278  
 fixedParameters (*amici.amici.ExpData* property), 271  
 fixedParameters (*amici.amici.ExpDataPtr* property), 277  
 fixedParameters (*amici.amici.SimulationParameters* property), 314  
 fixedParametersPreequilibration (*amici.amici.ExpData* property), 271  
 fixedParametersPreequilibration (*amici.amici.ExpDataPtr* property), 277  
 fixedParametersPreequilibration (*amici.amici.SimulationParameters* property), 314  
 fixedParametersPresimulation (*amici.amici.ExpData* property), 271  
 fixedParametersPresimulation (*amici.amici.ExpDataPtr* property), 277  
 fixedParametersPresimulation (*amici.amici.SimulationParameters* property), 314  
 fixedpoint (*amici.amici.NonlinearSolverIteration* attribute), 299  
 flags (*amici.de\_export.TemplateAmici* attribute), 360  
 formula (*amici.splines.AbstractSpline* property), 398  
 formula (*amici.splines.CubicHermiteSpline* property), 402  
 forward (*amici.amici.SensitivityMethod* attribute), 313  
 free\_symbols (*amici.parameter\_mapping.ParameterMapping* property), 384  
 free\_symbols (*amici.parameter\_mapping.ParameterMappingForCondition* property), 385  
 free\_symbols() (*amici.de\_export.DEModel* method), 355  
 from\_annotation() (*amici.splines.AbstractSpline* static method), 398  
 from\_annotation() (*amici.splines.CubicHermiteSpline* static method), 402  
 full (*amici.amici.RDataReporting* attribute), 300  
 full (*amici.amici.SecondOrderMode* attribute), 313  
 functional (*amici.amici.NonlinearSolverIteration* attribute), 299  
 generate\_basic\_variables() (*amici.de\_export.DEModel* method), 355  
 generate\_flux\_symbol() (in module *amici.import\_utils*), 346  
 generate\_measurement\_symbol() (in module *amici.import\_utils*), 347  
 generate\_model\_code() (*amici.de\_export.DEEExporter* method), 352  
 generate\_regularization\_symbol() (in module *amici.import\_utils*), 347  
 get() (*amici.numpy.ExpDataView* method), 389  
 get() (*amici.numpy.ReturnDataView* method), 390  
 get() (*amici.numpy.SwigPtrView* method), 390  
 get\_annotation() (*amici.splines.AbstractSpline* static method), 398  
 get\_annotation() (*amici.splines.CubicHermiteSpline* static method), 402  
 get\_appearance\_counts() (*amici.de\_export.DEModel* method), 355  
 get\_conservation\_laws() (*amici.de\_export.DEModel* method), 355  
 get\_dt() (*amici.de\_model.DifferentialState* method), 367  
 get\_dx\_rdata\_dx\_solver() (*amici.de\_model.AlgebraicState* method), 364  
 get\_dx\_rdata\_dx\_solver() (*amici.de\_model.DifferentialState* method), 367  
 get\_dx\_rdata\_dx\_solver() (*amici.de\_model.State* method), 376  
 get\_event() (*amici.de\_model.EventObservable* method), 369  
 get\_expressions\_as\_dataframe() (in module *amici.pandas*), 381  
 get\_fixed\_parameters() (in module *amici.petab\_import*), 334  
 get\_free\_symbols() (*amici.de\_model.AlgebraicEquation* method), 363  
 get\_free\_symbols() (*amici.de\_model.AlgebraicState* method), 364  
 get\_free\_symbols() (*amici.de\_model.DifferentialState* method), 367  
 get\_function\_extern\_declaration() (in module *amici.de\_export*), 360  
 get\_id() (*amici.de\_model.AlgebraicEquation* method), 363  
 get\_id() (*amici.de\_model.AlgebraicState* method), 364  
 get\_id() (*amici.de\_model.ConservationLaw* method), 365  
 get\_id() (*amici.de\_model.Constant* method), 366  
 get\_id() (*amici.de\_model.DifferentialState* method), 367  
 get\_id() (*amici.de\_model.Event* method), 368  
 get\_id() (*amici.de\_model.EventObservable* method), 369  
 get\_id() (*amici.de\_model.Expression* method), 370

## G

[get\\_id\(\)](#) (*amici.de\_model.LogLikelihoodRZ* method), 371  
[get\\_id\(\)](#) (*amici.de\_model.LogLikelihoodY* method), 371  
[get\\_id\(\)](#) (*amici.de\_model.LogLikelihoodZ* method), 372  
[get\\_id\(\)](#) (*amici.de\_model.ModelQuantity* method), 372  
[get\\_id\(\)](#) (*amici.de\_model.Observable* method), 373  
[get\\_id\(\)](#) (*amici.de\_model.Parameter* method), 374  
[get\\_id\(\)](#) (*amici.de\_model.SigmaY* method), 375  
[get\\_id\(\)](#) (*amici.de\_model.SigmaZ* method), 375  
[get\\_id\(\)](#) (*amici.de\_model.State* method), 376  
[get\\_initial\\_value\(\)](#) (*amici.de\_model.Event* method), 368  
[get\\_logger\(\)](#) (in module *amici.logging*), 382  
[get\\_measurement\\_symbol\(\)](#) (*amici.de\_model.EventObservable* method), 369  
[get\\_measurement\\_symbol\(\)](#) (*amici.de\_model.Observable* method), 373  
[get\\_model\(\)](#) (*amici.ModelModule* method), 265  
[get\\_model\\_override\\_implementation\(\)](#) (in module *amici.de\_export*), 360  
[get\\_name\(\)](#) (*amici.de\_model.AlgebraicEquation* method), 364  
[get\\_name\(\)](#) (*amici.de\_model.AlgebraicState* method), 364  
[get\\_name\(\)](#) (*amici.de\_model.ConservationLaw* method), 365  
[get\\_name\(\)](#) (*amici.de\_model.Constant* method), 366  
[get\\_name\(\)](#) (*amici.de\_model.DifferentialState* method), 367  
[get\\_name\(\)](#) (*amici.de\_model.Event* method), 368  
[get\\_name\(\)](#) (*amici.de\_model.EventObservable* method), 369  
[get\\_name\(\)](#) (*amici.de\_model.Expression* method), 370  
[get\\_name\(\)](#) (*amici.de\_model.LogLikelihoodRZ* method), 371  
[get\\_name\(\)](#) (*amici.de\_model.LogLikelihoodY* method), 371  
[get\\_name\(\)](#) (*amici.de\_model.LogLikelihoodZ* method), 372  
[get\\_name\(\)](#) (*amici.de\_model.ModelQuantity* method), 373  
[get\\_name\(\)](#) (*amici.de\_model.Observable* method), 373  
[get\\_name\(\)](#) (*amici.de\_model.Parameter* method), 374  
[get\\_name\(\)](#) (*amici.de\_model.SigmaY* method), 375  
[get\\_name\(\)](#) (*amici.de\_model.SigmaZ* method), 375  
[get\\_name\(\)](#) (*amici.de\_model.State* method), 376  
[get\\_ncoeff\(\)](#) (*amici.de\_model.ConservationLaw* method), 365  
[get\\_observable\\_transformations\(\)](#) (*amici.de\_export.DEModel* method), 356  
[get\\_observation\\_model\(\)](#) (in module *amici.petab\_import*), 335  
[get\\_regularization\\_symbol\(\)](#) (*amici.de\_model.EventObservable* method), 369  
[get\\_regularization\\_symbol\(\)](#) (*amici.de\_model.Observable* method), 374  
[get\\_sbml\\_units\(\)](#) (in module *amici.sbml\_utils*), 394  
[get\\_solver\\_indices\(\)](#) (*amici.de\_export.DEModel* method), 356  
[get\\_species\\_initial\(\)](#) (in module *amici.sbml\_import*), 330  
[get\\_sunindex\\_extern\\_declaration\(\)](#) (in module *amici.de\_export*), 361  
[get\\_sunindex\\_override\\_implementation\(\)](#) (in module *amici.de\_export*), 361  
[get\\_val\(\)](#) (*amici.de\_model.AlgebraicEquation* method), 364  
[get\\_val\(\)](#) (*amici.de\_model.AlgebraicState* method), 364  
[get\\_val\(\)](#) (*amici.de\_model.ConservationLaw* method), 365  
[get\\_val\(\)](#) (*amici.de\_model.Constant* method), 366  
[get\\_val\(\)](#) (*amici.de\_model.DifferentialState* method), 367  
[get\\_val\(\)](#) (*amici.de\_model.Event* method), 369  
[get\\_val\(\)](#) (*amici.de\_model.EventObservable* method), 369  
[get\\_val\(\)](#) (*amici.de\_model.Expression* method), 370  
[get\\_val\(\)](#) (*amici.de\_model.LogLikelihoodRZ* method), 371  
[get\\_val\(\)](#) (*amici.de\_model.LogLikelihoodY* method), 371  
[get\\_val\(\)](#) (*amici.de\_model.LogLikelihoodZ* method), 372  
[get\\_val\(\)](#) (*amici.de\_model.ModelQuantity* method), 373  
[get\\_val\(\)](#) (*amici.de\_model.Observable* method), 374  
[get\\_val\(\)](#) (*amici.de\_model.Parameter* method), 374  
[get\\_val\(\)](#) (*amici.de\_model.SigmaY* method), 375  
[get\\_val\(\)](#) (*amici.de\_model.SigmaZ* method), 376  
[get\\_val\(\)](#) (*amici.de\_model.State* method), 376  
[get\\_x\\_rdata\(\)](#) (*amici.de\_model.AlgebraicState* method), 364  
[get\\_x\\_rdata\(\)](#) (*amici.de\_model.ConservationLaw* method), 366  
[get\\_x\\_rdata\(\)](#) (*amici.de\_model.DifferentialState* method), 367  
[get\\_x\\_rdata\(\)](#) (*amici.de\_model.State* method), 376  
[getAbsoluteTolerance\(\)](#) (*amici.amici.Solver* method), 316  
[getAbsoluteToleranceB\(\)](#) (*amici.amici.Solver* method), 316  
[getAbsoluteToleranceFSA\(\)](#) (*amici.amici.Solver* method), 316

[getAbsoluteToleranceQuadratures\(\)](#) (*amici.amici.Solver method*), 316  
[getAbsoluteToleranceSteadyState\(\)](#) (*amici.amici.Solver method*), 317  
[getAbsoluteToleranceSteadyStateSensi\(\)](#) (*amici.amici.Solver method*), 317  
[getAddSigmaResiduals\(\)](#) (*amici.amici.Model method*), 283  
[getAlwaysCheckFinite\(\)](#) (*amici.amici.Model method*), 283  
[getAmiciCommit\(\)](#) (*amici.amici.Model method*), 283  
[getAmiciVersion\(\)](#) (*amici.amici.Model method*), 283  
[getDataObservablesAsDataFrame\(\)](#) (in module *amici.pandas*), 379  
[getEdataFromDataFrame\(\)](#) (in module *amici.pandas*), 379  
[getExpressionIds\(\)](#) (*amici.amici.Model method*), 283  
[getExpressionNames\(\)](#) (*amici.amici.Model method*), 283  
[getFixedParameterById\(\)](#) (*amici.amici.Model method*), 283  
[getFixedParameterByName\(\)](#) (*amici.amici.Model method*), 283  
[getFixedParameterIds\(\)](#) (*amici.amici.Model method*), 283  
[getFixedParameterNames\(\)](#) (*amici.amici.Model method*), 284  
[getFixedParameters\(\)](#) (*amici.amici.Model method*), 284  
[getInitialStates\(\)](#) (*amici.amici.Model method*), 284  
[getInitialStateSensitivities\(\)](#) (*amici.amici.Model method*), 284  
[getInternalSensitivityMethod\(\)](#) (*amici.amici.Solver method*), 317  
[getInterpolationType\(\)](#) (*amici.amici.Solver method*), 317  
[getLinearMultistepMethod\(\)](#) (*amici.amici.Solver method*), 317  
[getLinearSolver\(\)](#) (*amici.amici.Solver method*), 317  
[getMaxSteps\(\)](#) (*amici.amici.Solver method*), 317  
[getMaxStepsBackwardProblem\(\)](#) (*amici.amici.Solver method*), 317  
[getMaxTime\(\)](#) (*amici.amici.Solver method*), 317  
[getMinimumSigmaResiduals\(\)](#) (*amici.amici.Model method*), 284  
[getModel\(\)](#) (*amici.ModelModule method*), 265  
[getName\(\)](#) (*amici.amici.Model method*), 284  
[getNewtonDampingFactorLowerBound\(\)](#) (*amici.amici.Solver method*), 318  
[getNewtonDampingFactorMode\(\)](#) (*amici.amici.Solver method*), 318  
[getNewtonMaxSteps\(\)](#) (*amici.amici.Solver method*), 318  
[getNewtonStepSteadyStateCheck\(\)](#) (*amici.amici.Solver method*), 318  
[getNonlinearSolverIteration\(\)](#) (*amici.amici.Solver method*), 318  
[getObservableIds\(\)](#) (*amici.amici.Model method*), 284  
[getObservableNames\(\)](#) (*amici.amici.Model method*), 284  
[getObservableScaling\(\)](#) (*amici.amici.Model method*), 284  
[getObservedData\(\)](#) (*amici.amici.ExpData method*), 271  
[getObservedDataPtr\(\)](#) (*amici.amici.ExpData method*), 271  
[getObservedDataStdDev\(\)](#) (*amici.amici.ExpData method*), 271  
[getObservedDataStdDevPtr\(\)](#) (*amici.amici.ExpData method*), 271  
[getObservedEvents\(\)](#) (*amici.amici.ExpData method*), 271  
[getObservedEventsPtr\(\)](#) (*amici.amici.ExpData method*), 271  
[getObservedEventsStdDev\(\)](#) (*amici.amici.ExpData method*), 272  
[getObservedEventsStdDevPtr\(\)](#) (*amici.amici.ExpData method*), 272  
[getParameterById\(\)](#) (*amici.amici.Model method*), 285  
[getParameterByName\(\)](#) (*amici.amici.Model method*), 285  
[getParameterIds\(\)](#) (*amici.amici.Model method*), 285  
[getParameterList\(\)](#) (*amici.amici.Model method*), 285  
[getParameterNames\(\)](#) (*amici.amici.Model method*), 285  
[getParameters\(\)](#) (*amici.amici.Model method*), 285  
[getParameterScale\(\)](#) (*amici.amici.Model method*), 285  
[getReinitializationStateIdxs\(\)](#) (*amici.amici.Model method*), 285  
[getReinitializeFixedParameterInitialStates\(\)](#) (*amici.amici.Model method*), 286  
[getRelativeTolerance\(\)](#) (*amici.amici.Solver method*), 318  
[getRelativeToleranceB\(\)](#) (*amici.amici.Solver method*), 318  
[getRelativeToleranceFSA\(\)](#) (*amici.amici.Solver method*), 318  
[getRelativeToleranceQuadratures\(\)](#) (*amici.amici.Solver method*), 318  
[getRelativeToleranceSteadyState\(\)](#) (*amici.amici.Solver method*), 319  
[getRelativeToleranceSteadyStateSensi\(\)](#) (*amici.amici.Solver method*), 319  
[getResidualsAsDataFrame\(\)](#) (in module *amici.pandas*), 380  
[getReturnDataReportingMode\(\)](#) (*amici.amici.Solver method*), 319

getScaledParameter() (in module *amici.amici*), 324  
 getSensiSteadyStateCheck() (*amici.amici.Solver* method), 319  
 getSensitivityMethod() (*amici.amici.Solver* method), 319  
 getSensitivityMethodPreequilibration() (*amici.amici.Solver* method), 319  
 getSensitivityOrder() (*amici.amici.Solver* method), 319  
 getSimulationObservablesAsDataFrame() (in module *amici.pandas*), 380  
 getSimulationStatesAsDataFrame() (in module *amici.pandas*), 381  
 getSolver() (*amici.amici.Model* method), 286  
 getStabilityLimitFlag() (*amici.amici.Solver* method), 319  
 getStateIds() (*amici.amici.Model* method), 286  
 getStateIdsSolver() (*amici.amici.Model* method), 286  
 getStateIsNonNegative() (*amici.amici.Model* method), 286  
 getStateNames() (*amici.amici.Model* method), 286  
 getStateNamesSolver() (*amici.amici.Model* method), 286  
 getStateOrdering() (*amici.amici.Solver* method), 319  
 getSteadyStateComputationMode() (*amici.amici.Model* method), 286  
 getSteadyStateSensitivityMode() (*amici.amici.Model* method), 286  
 getSteadyStateSensiToleranceFactor() (*amici.amici.Solver* method), 320  
 getSteadyStateToleranceFactor() (*amici.amici.Solver* method), 320  
 getTimepoint() (*amici.amici.ExpData* method), 272  
 getTimepoint() (*amici.amici.Model* method), 287  
 getTimepoints() (*amici.amici.ExpData* method), 272  
 getTimepoints() (*amici.amici.Model* method), 287  
 getUnscaledParameter() (in module *amici.amici*), 324  
 getUnscaledParameters() (*amici.amici.Model* method), 287  
 grouper() (in module *amici.import\_utils*), 347  
 gsl::make\_span (C++ function), 673

## H

has\_conservation\_law() (*amici.de\_model.AlgebraicState* method), 365  
 has\_conservation\_law() (*amici.de\_model.DifferentialState* method), 367  
 has\_conservation\_law() (*amici.de\_model.State* method), 376  
 has\_fixed\_parameter\_ic() (in module *amici.pysb\_import*), 331  
 hasCustomInitialStates() (*amici.amici.Model* method), 287  
 hasCustomInitialStateSensitivities() (*amici.amici.Model* method), 287  
 hasExpressionIds() (*amici.amici.Model* method), 287  
 hasExpressionNames() (*amici.amici.Model* method), 287  
 hasFixedParameterIds() (*amici.amici.Model* method), 287  
 hasFixedParameterNames() (*amici.amici.Model* method), 288  
 hasObservableIds() (*amici.amici.Model* method), 288  
 hasObservableNames() (*amici.amici.Model* method), 288  
 hasParameterIds() (*amici.amici.Model* method), 288  
 hasParameterNames() (*amici.amici.Model* method), 288  
 hasQuadraticLLH() (*amici.amici.Model* method), 288  
 hasStateIds() (*amici.amici.Model* method), 288  
 hasStateNames() (*amici.amici.Model* method), 288  
 hdf5\_enabled (in module *amici*), 265  
 hermite (*amici.amici.InterpolationType* attribute), 278  
 |  
 id (*amici.amici.ExpData* property), 272  
 id (*amici.amici.ExpDataPtr* property), 277  
 id (*amici.amici.ReturnData* property), 301  
 id (*amici.amici.ReturnDataPtr* property), 307  
 IDAS, 55  
 identifier (*amici.amici.LogItem* property), 279  
 idlist (*amici.amici.Model* property), 289  
 idlist (*amici.amici.ModelPtr* property), 298  
 idpattern (*amici.de\_export.TemplateAmici* attribute), 360  
 import\_from\_sbml\_importer() (*amici.de\_export.DEModel* method), 356  
 import\_model() (in module *amici.petab\_import*), 335  
 import\_model\_module() (in module *amici*), 265  
 import\_model\_pysb() (in module *amici.petab\_import\_pysb*), 338  
 import\_model\_sbml() (in module *amici.petab\_import*), 336  
 import\_petab\_problem() (in module *amici.petab\_import*), 337  
 index() (*amici.parameter\_mapping.ParameterMapping* method), 384  
 index() (*amici.splines.UniformGrid* method), 404  
 initializeSplines() (*amici.amici.Model* method), 289  
 initializeSplineSensitivities() (*amici.amici.Model* method), 289  
 integrate() (*amici.splines.AbstractSpline* method), 398



- `integrate()` (*amici.splines.CubicHermiteSpline* method), 402
- `integrateIfNewtonFails` (*amici.amici.SteadyStateComputationMode* attribute), 323
- `integrateIfNewtonFails` (*amici.amici.SteadyStateSensitivityMode* attribute), 323
- `integrationOnly` (*amici.amici.SteadyStateComputationMode* attribute), 323
- `integrationOnly` (*amici.amici.SteadyStateSensitivityMode* attribute), 323
- `InternalSensitivityMethod` (class in *amici.amici*), 278
- `InterpolationType` (class in *amici.amici*), 278
- `IntVector` (class in *amici.amici*), 278
- `is_assignment_rule_target()` (*amici.sbml\_import.SbmlImporter* method), 328
- `is_ode()` (*amici.de\_export.DEModel* method), 356
- `is_rate_rule_target()` (*amici.sbml\_import.SbmlImporter* method), 328
- `is_spline()` (*amici.splines.AbstractSpline* static method), 398
- `is_spline()` (*amici.splines.CubicHermiteSpline* static method), 402
- `is_valid_identifider()` (in module *amici.de\_export*), 361
- `isFixedParameterStateReinitializationAllowed()` (*amici.amici.Model* method), 289
- `isSetObservedData()` (*amici.amici.ExpData* method), 272
- `isSetObservedDataStdDev()` (*amici.amici.ExpData* method), 272
- `isSetObservedEvents()` (*amici.amici.ExpData* method), 272
- `isSetObservedEventsStdDev()` (*amici.amici.ExpData* method), 273
- `items()` (*amici.numpy.ExpDataView* method), 389
- `items()` (*amici.numpy.ReturnDataView* method), 390
- `items()` (*amici.numpy.SwigPtrView* method), 390
- ## J
- `J` (*amici.amici.ReturnData* property), 300
- `J` (*amici.amici.ReturnDataPtr* property), 307
- ## K
- `k()` (*amici.amici.Model* method), 289
- `keys()` (*amici.numpy.ExpDataView* method), 389
- `keys()` (*amici.numpy.ReturnDataView* method), 390
- `keys()` (*amici.numpy.SwigPtrView* method), 390
- ## L
- `KLU` (*amici.amici.LinearSolver* attribute), 279
- `LAPACKBand` (*amici.amici.LinearSolver* attribute), 279
- `LAPACKDense` (*amici.amici.LinearSolver* attribute), 279
- `lbw` (*amici.amici.Model* property), 289
- `lbw` (*amici.amici.ModelDimensions* property), 296
- `lbw` (*amici.amici.ModelPtr* property), 298
- `lbw` (*amici.amici.ReturnData* property), 301
- `lbw` (*amici.amici.ReturnDataPtr* property), 307
- `likelihood` (*amici.amici.RDataReporting* attribute), 300
- `lin` (*amici.amici.ObservableScaling* attribute), 299
- `LIN` (*amici.import\_utils.ObservableTransformation* attribute), 346
- `LinearMultistepMethod` (class in *amici.amici*), 278
- `LinearSolver` (class in *amici.amici*), 279
- `llh` (*amici.amici.ReturnData* property), 301
- `llh` (*amici.amici.ReturnDataPtr* property), 307
- `ln` (*amici.amici.ParameterScaling* attribute), 300
- `log` (*amici.amici.ObservableScaling* attribute), 299
- `LOG` (*amici.import\_utils.ObservableTransformation* attribute), 346
- `log10` (*amici.amici.ObservableScaling* attribute), 299
- `log10` (*amici.amici.ParameterScaling* attribute), 300
- `LOG10` (*amici.import\_utils.ObservableTransformation* attribute), 346
- `log_execution_time()` (in module *amici.logging*), 382
- `log_likelihood_rzs()` (*amici.de\_export.DEModel* method), 356
- `log_likelihood_ys()` (*amici.de\_export.DEModel* method), 356
- `log_likelihood_zs()` (*amici.de\_export.DEModel* method), 356
- `logarithmic_parametrization` (*amici.splines.AbstractSpline* property), 399
- `logarithmic_parametrization` (*amici.splines.CubicHermiteSpline* property), 402
- `logger` (*amici.amici.Model* property), 289
- `logger` (*amici.amici.ModelPtr* property), 298
- `logger` (*amici.amici.Solver* property), 320
- `logger` (*amici.amici.SolverPtr* property), 323
- `LogItem` (class in *amici.amici*), 279
- `LogItemVector` (class in *amici.amici*), 279
- `LogLikelihoodRZ` (class in *amici.de\_model*), 370
- `LogLikelihoodY` (class in *amici.de\_model*), 371
- `LogLikelihoodZ` (class in *amici.de\_model*), 372
- ## M
- `M_1_PI` (C macro), 680
- `M_2_PI` (C macro), 680
- `M_2_SQRTPI` (C macro), 680
- `M_E` (C macro), 681

- `M_LN10` (*C macro*), 681
- `M_LN2` (*C macro*), 681
- `M_LOG10E` (*C macro*), 681
- `M_LOG2E` (*C macro*), 681
- `M_PI` (*C macro*), 682
- `M_PI_2` (*C macro*), 682
- `M_PI_4` (*C macro*), 682
- `M_SQRT1_2` (*C macro*), 682
- `M_SQRT2` (*C macro*), 683
- `mathml2sympy()` (*in module amici.sbml\_utils*), 395
- `mathml_formula` (*amici.splines.AbstractSpline property*), 399
- `mathml_formula` (*amici.splines.CubicHermiteSpline property*), 402
- `mathml_tag()` (*amici.sbml\_utils.MathMLSbmlPrinter method*), 392
- `MathMLSbmlPrinter` (*class in amici.sbml\_utils*), 391
- `message` (*amici.amici.LogItem property*), 279
- `messages` (*amici.amici.ReturnData property*), 302
- `messages` (*amici.amici.ReturnDataPtr property*), 307
- `method` (*amici.splines.AbstractSpline property*), 399
- `method` (*amici.splines.CubicHermiteSpline property*), 402
- `Model` (*class in amici.amici*), 279
- `ModelDimensions` (*class in amici.amici*), 295
- `ModelModule` (*class in amici*), 264
- `ModelPtr` (*class in amici.amici*), 298
- `ModelQuantity` (*class in amici.de\_model*), 372
- `module`
  - `amici`, 264
  - `amici.amici`, 266
  - `amici.bngl_import`, 333
  - `amici.conservated_quantities_demartino`, 387
  - `amici.conservated_quantities_rref`, 388
  - `amici.de_export`, 350
  - `amici.de_model`, 362
  - `amici.gradient_check`, 383
  - `amici.import_utils`, 345
  - `amici.logging`, 381
  - `amici.numpy`, 389
  - `amici.pandas`, 379
  - `amici.parameter_mapping`, 384
  - `amici.petab_import`, 334
  - `amici.petab_import_pysb`, 338
  - `amici.petab_objective`, 339
  - `amici.petab_simulate`, 343
  - `amici.plotting`, 377
  - `amici.pysb_import`, 331
  - `amici.sbml_import`, 326
  - `amici.sbml_utils`, 391
  - `amici.splines`, 396
  - `nc1()` (*amici.amici.Model method*), 289
  - `ndJydy` (*amici.amici.Model property*), 289
  - `ndJydy` (*amici.amici.ModelDimensions property*), 296
  - `ndJydy` (*amici.amici.ModelPtr property*), 298
  - `ndJydy` (*amici.amici.ReturnData property*), 302
  - `ndJydy` (*amici.amici.ReturnDataPtr property*), 308
  - `ndtotal_cldx_rdata` (*amici.amici.Model property*), 289
  - `ndtotal_cldx_rdata` (*amici.amici.ModelDimensions property*), 296
  - `ndtotal_cldx_rdata` (*amici.amici.ModelPtr property*), 298
  - `ndtotal_cldx_rdata` (*amici.amici.ReturnData property*), 302
  - `ndtotal_cldx_rdata` (*amici.amici.ReturnDataPtr property*), 308
  - `ndwdp` (*amici.amici.Model property*), 289
  - `ndwdp` (*amici.amici.ModelDimensions property*), 296
  - `ndwdp` (*amici.amici.ModelPtr property*), 298
  - `ndwdp` (*amici.amici.ReturnData property*), 302
  - `ndwdp` (*amici.amici.ReturnDataPtr property*), 308
  - `ndwdw` (*amici.amici.Model property*), 289
  - `ndwdw` (*amici.amici.ModelDimensions property*), 296
  - `ndwdw` (*amici.amici.ModelPtr property*), 298
  - `ndwdw` (*amici.amici.ReturnData property*), 302
  - `ndwdw` (*amici.amici.ReturnDataPtr property*), 308
  - `ndwdx` (*amici.amici.Model property*), 289
  - `ndwdx` (*amici.amici.ModelDimensions property*), 297
  - `ndwdx` (*amici.amici.ModelPtr property*), 298
  - `ndwdx` (*amici.amici.ReturnData property*), 302
  - `ndwdx` (*amici.amici.ReturnDataPtr property*), 308
  - `ndxdotdw` (*amici.amici.Model property*), 290
  - `ndxdotdw` (*amici.amici.ModelDimensions property*), 297
  - `ndxdotdw` (*amici.amici.ModelPtr property*), 298
  - `ndxdotdw` (*amici.amici.ReturnData property*), 302
  - `ndxdotdw` (*amici.amici.ReturnDataPtr property*), 308
  - `ndxrdatadtcl` (*amici.amici.Model property*), 290
  - `ndxrdatadtcl` (*amici.amici.ModelDimensions property*), 297
  - `ndxrdatadtcl` (*amici.amici.ModelPtr property*), 298
  - `ndxrdatadtcl` (*amici.amici.ReturnData property*), 302
  - `ndxrdatadtcl` (*amici.amici.ReturnDataPtr property*), 308
  - `ndxrdatadxsolver` (*amici.amici.Model property*), 290
  - `ndxrdatadxsolver` (*amici.amici.ModelDimensions property*), 297
  - `ndxrdatadxsolver` (*amici.amici.ModelPtr property*), 298
  - `ndxrdatadxsolver` (*amici.amici.ReturnData property*), 302
  - `ndxrdatadxsolver` (*amici.amici.ReturnDataPtr property*), 308
  - `ne` (*amici.amici.Model property*), 290
  - `ne` (*amici.amici.ModelDimensions property*), 297

## N

`name()` (*amici.de\_export.DEModel method*), 356

- `ne` (*amici.amici.ModelPtr* property), 298
- `ne` (*amici.amici.ReturnData* property), 302
- `ne` (*amici.amici.ReturnDataPtr* property), 308
- `newton` (*amici.amici.NonlinearSolverIteration* attribute), 299
- `newton_maxsteps` (*amici.amici.ReturnData* property), 302
- `newton_maxsteps` (*amici.amici.ReturnDataPtr* property), 308
- `NewtonDampingFactorMode` (class in *amici.amici*), 299
- `newtonOnly` (*amici.amici.SteadyStateComputationMode* attribute), 323
- `newtonOnly` (*amici.amici.SteadyStateSensitivityMode* attribute), 324
- `nJ` (*amici.amici.Model* property), 289
- `nJ` (*amici.amici.ModelDimensions* property), 296
- `nJ` (*amici.amici.ModelPtr* property), 298
- `nJ` (*amici.amici.ReturnData* property), 302
- `nJ` (*amici.amici.ReturnDataPtr* property), 307
- `nk` (*amici.amici.ModelDimensions* property), 297
- `nk` (*amici.amici.ReturnData* property), 302
- `nk` (*amici.amici.ReturnDataPtr* property), 308
- `nk()` (*amici.amici.Model* method), 290
- `nmaxevent` (*amici.amici.ReturnData* property), 302
- `nmaxevent` (*amici.amici.ReturnDataPtr* property), 308
- `nmaxevent()` (*amici.amici.ExpData* method), 273
- `nMaxEvent()` (*amici.amici.Model* method), 289
- `nnz` (*amici.amici.Model* property), 290
- `nnz` (*amici.amici.ModelDimensions* property), 297
- `nnz` (*amici.amici.ModelPtr* property), 298
- `nnz` (*amici.amici.ReturnData* property), 302
- `nnz` (*amici.amici.ReturnDataPtr* property), 308
- `nodes` (*amici.splines.AbstractSpline* property), 399
- `nodes` (*amici.splines.CubicHermiteSpline* property), 402
- `noise_distribution_to_cost_function()` (in module *amici.import\_utils*), 347
- `noise_distribution_to_observable_transformation()` (in module *amici.import\_utils*), 348
- `none` (*amici.amici.ParameterScaling* attribute), 300
- `none` (*amici.amici.SecondOrderMode* attribute), 313
- `none` (*amici.amici.SensitivityMethod* attribute), 313
- `none` (*amici.amici.SensitivityOrder* attribute), 313
- `NonlinearSolverIteration` (class in *amici.amici*), 299
- `NonlinearSolverIteration_fixedpoint` (in module *amici.amici*), 299
- `not_run` (*amici.amici.SteadyStateStatus* attribute), 324
- `np` (*amici.amici.ModelDimensions* property), 297
- `np` (*amici.amici.ReturnData* property), 302
- `np` (*amici.amici.ReturnDataPtr* property), 308
- `np()` (*amici.amici.Model* method), 290
- `nplist` (*amici.amici.ReturnData* property), 302
- `nplist` (*amici.amici.ReturnDataPtr* property), 308
- `nplist()` (*amici.amici.Model* method), 290
- `nplist()` (*amici.amici.Solver* method), 320
- `nquad()` (*amici.amici.Solver* method), 320
- `nspl` (*amici.amici.Model* property), 290
- `nspl` (*amici.amici.ModelDimensions* property), 297
- `nspl` (*amici.amici.ModelPtr* property), 298
- `nspl` (*amici.amici.ReturnData* property), 302
- `nspl` (*amici.amici.ReturnDataPtr* property), 308
- `nt` (*amici.amici.ReturnData* property), 302
- `nt` (*amici.amici.ReturnDataPtr* property), 308
- `nt()` (*amici.amici.ExpData* method), 273
- `nt()` (*amici.amici.Model* method), 290
- `nullspace_by_rref()` (in module *amici.conservated\_quantities\_rref*), 388
- `num_cons_law()` (*amici.de\_export.DEModel* method), 356
- `num_const()` (*amici.de\_export.DEModel* method), 356
- `num_eventobs()` (*amici.de\_export.DEModel* method), 357
- `num_events()` (*amici.de\_export.DEModel* method), 357
- `num_expr()` (*amici.de\_export.DEModel* method), 357
- `num_obs()` (*amici.de\_export.DEModel* method), 357
- `num_par()` (*amici.de\_export.DEModel* method), 357
- `num_state_reinits()` (*amici.de\_export.DEModel* method), 357
- `num_states_rdata()` (*amici.de\_export.DEModel* method), 357
- `num_states_solver()` (*amici.de\_export.DEModel* method), 357
- `number_of_nodes` (*amici.splines.UniformGrid* property), 404
- `numerrtestfails` (*amici.amici.ReturnData* property), 303
- `numerrtestfails` (*amici.amici.ReturnDataPtr* property), 308
- `numerrtestfailsB` (*amici.amici.ReturnData* property), 303
- `numerrtestfailsB` (*amici.amici.ReturnDataPtr* property), 308
- `numnonlinsolvconvfails` (*amici.amici.ReturnData* property), 303
- `numnonlinsolvconvfails` (*amici.amici.ReturnDataPtr* property), 309
- `numnonlinsolvconvfailsB` (*amici.amici.ReturnData* property), 303
- `numnonlinsolvconvfailsB` (*amici.amici.ReturnDataPtr* property), 309
- `numrhsevals` (*amici.amici.ReturnData* property), 303
- `numrhsevals` (*amici.amici.ReturnDataPtr* property), 309
- `numrhsevalsB` (*amici.amici.ReturnData* property), 303
- `numrhsevalsB` (*amici.amici.ReturnDataPtr* property), 309
- `numsteps` (*amici.amici.ReturnData* property), 303
- `numsteps` (*amici.amici.ReturnDataPtr* property), 309

numstepsB (*amici.amici.ReturnData* property), 303  
 numstepsB (*amici.amici.ReturnDataPtr* property), 309  
 nw (*amici.amici.Model* property), 290  
 nw (*amici.amici.ModelDimensions* property), 297  
 nw (*amici.amici.ModelPtr* property), 298  
 nw (*amici.amici.ReturnData* property), 303  
 nw (*amici.amici.ReturnDataPtr* property), 309  
 nx (*amici.amici.ReturnData* property), 303  
 nx (*amici.amici.ReturnDataPtr* property), 309  
 nx() (*amici.amici.Solver* method), 320  
 nx\_rdata (*amici.amici.Model* property), 290  
 nx\_rdata (*amici.amici.ModelDimensions* property), 297  
 nx\_rdata (*amici.amici.ModelPtr* property), 298  
 nx\_rdata (*amici.amici.ReturnData* property), 303  
 nx\_rdata (*amici.amici.ReturnDataPtr* property), 309  
 nx\_reinit() (*amici.amici.Model* method), 290  
 nx\_solver (*amici.amici.Model* property), 290  
 nx\_solver (*amici.amici.ModelDimensions* property), 297  
 nx\_solver (*amici.amici.ModelPtr* property), 298  
 nx\_solver (*amici.amici.ReturnData* property), 303  
 nx\_solver (*amici.amici.ReturnDataPtr* property), 309  
 nx\_solver\_reinit (*amici.amici.Model* property), 291  
 nx\_solver\_reinit (*amici.amici.ModelDimensions* property), 297  
 nx\_solver\_reinit (*amici.amici.ModelPtr* property), 299  
 nx\_solver\_reinit (*amici.amici.ReturnData* property), 303  
 nx\_solver\_reinit (*amici.amici.ReturnDataPtr* property), 309  
 nxtrue (*amici.amici.ReturnData* property), 303  
 nxtrue (*amici.amici.ReturnDataPtr* property), 309  
 nxtrue\_rdata (*amici.amici.Model* property), 291  
 nxtrue\_rdata (*amici.amici.ModelDimensions* property), 297  
 nxtrue\_rdata (*amici.amici.ModelPtr* property), 299  
 nxtrue\_rdata (*amici.amici.ReturnData* property), 303  
 nxtrue\_rdata (*amici.amici.ReturnDataPtr* property), 309  
 nxtrue\_solver (*amici.amici.Model* property), 291  
 nxtrue\_solver (*amici.amici.ModelDimensions* property), 297  
 nxtrue\_solver (*amici.amici.ModelPtr* property), 299  
 nxtrue\_solver (*amici.amici.ReturnData* property), 303  
 nxtrue\_solver (*amici.amici.ReturnDataPtr* property), 309  
 ny (*amici.amici.Model* property), 291  
 ny (*amici.amici.ModelDimensions* property), 297  
 ny (*amici.amici.ModelPtr* property), 299  
 ny (*amici.amici.ReturnData* property), 303  
 ny (*amici.amici.ReturnDataPtr* property), 309  
 nytrue (*amici.amici.Model* property), 291  
 nytrue (*amici.amici.ModelDimensions* property), 297

nytrue (*amici.amici.ModelPtr* property), 299  
 nytrue (*amici.amici.ReturnData* property), 303  
 nytrue (*amici.amici.ReturnDataPtr* property), 309  
 nytrue() (*amici.amici.ExpData* method), 273  
 nz (*amici.amici.Model* property), 291  
 nz (*amici.amici.ModelDimensions* property), 297  
 nz (*amici.amici.ModelPtr* property), 299  
 nz (*amici.amici.ReturnData* property), 303  
 nz (*amici.amici.ReturnDataPtr* property), 309  
 nztrue (*amici.amici.Model* property), 291  
 nztrue (*amici.amici.ModelDimensions* property), 297  
 nztrue (*amici.amici.ModelPtr* property), 299  
 nztrue (*amici.amici.ReturnData* property), 304  
 nztrue (*amici.amici.ReturnDataPtr* property), 309  
 nztrue() (*amici.amici.ExpData* method), 273

## O

o2mode (*amici.amici.Model* property), 291  
 o2mode (*amici.amici.ModelPtr* property), 299  
 o2mode (*amici.amici.ReturnData* property), 304  
 o2mode (*amici.amici.ReturnDataPtr* property), 309  
 Observable (class in *amici.de\_model*), 373  
 observables() (*amici.de\_export.DEModel* method), 357  
 ObservableScaling (class in *amici.amici*), 299  
 ObservableTransformation (class in *amici.import\_utils*), 346  
 ODE, 55  
 ode\_model\_from\_pysb\_importer() (in module *amici.pysb\_import*), 331  
 ode\_model\_symbol() (*amici.splines.AbstractSpline* method), 399  
 ode\_model\_symbol() (*amici.splines.CubicHermiteSpline* method), 402  
 off (*amici.amici.NewtonDampingFactorMode* attribute), 299  
 on (*amici.amici.NewtonDampingFactorMode* attribute), 299  
 order (*amici.amici.ReturnData* property), 304  
 order (*amici.amici.ReturnDataPtr* property), 310  
 order (*amici.sbml\_utils.MathMLSBmlPrinter* property), 392

## P

Parameter (class in *amici.de\_model*), 374  
 ParameterMapping (class in *amici.parameter\_mapping*), 384  
 ParameterMappingForCondition (class in *amici.parameter\_mapping*), 385  
 parameters (*amici.amici.ExpData* property), 273  
 parameters (*amici.amici.ExpDataPtr* property), 277  
 parameters (*amici.amici.SimulationParameters* property), 314



[parameters\(\)](#) (*amici.de\_export.DEModel* method), 358  
[parameters\(\)](#) (*amici.splines.AbstractSpline* method), 399  
[parameters\(\)](#) (*amici.splines.CubicHermiteSpline* method), 402  
[ParameterScaling](#) (class in *amici.amici*), 300  
[parameterScalingFromIntVector\(\)](#) (in module *amici.amici*), 324  
[ParameterScalingVector](#) (class in *amici.amici*), 300  
[parse\\_events\(\)](#) (*amici.de\_export.DEModel* method), 358  
[pattern](#) (*amici.de\_export.TemplateAmici* attribute), 360  
[period](#) (*amici.splines.AbstractSpline* property), 399  
[period](#) (*amici.splines.CubicHermiteSpline* property), 402  
[PETab](#), 55  
[petab\\_noise\\_distributions\\_to\\_amici\(\)](#) (in module *amici.petab\_import*), 337  
[petab\\_scale\\_to\\_amici\\_scale\(\)](#) (in module *amici.petab\_import*), 337  
[petab\\_to\\_amici\\_scale\(\)](#) (in module *amici.parameter\_mapping*), 386  
[PetabSimulator](#) (class in *amici.petab\_simulate*), 344  
[pivots\(\)](#) (in module *amici.conservated\_quantities\_rref*), 388  
[plist](#) (*amici.amici.ExpData* property), 273  
[plist](#) (*amici.amici.ExpDataPtr* property), 277  
[plist](#) (*amici.amici.SimulationParameters* property), 315  
[plist\(\)](#) (*amici.amici.Model* method), 291  
[plot\(\)](#) (*amici.splines.AbstractSpline* method), 399  
[plot\(\)](#) (*amici.splines.CubicHermiteSpline* method), 403  
[plot\\_expressions\(\)](#) (in module *amici.plotting*), 378  
[plot\\_jacobian\(\)](#) (in module *amici.plotting*), 378  
[plot\\_observable\\_trajectories\(\)](#) (in module *amici.plotting*), 378  
[plot\\_state\\_trajectories\(\)](#) (in module *amici.plotting*), 378  
[plotObservableTrajectories\(\)](#) (in module *amici.plotting*), 377  
[plotStateTrajectories\(\)](#) (in module *amici.plotting*), 377  
[poly\(\)](#) (*amici.splines.AbstractSpline* method), 399  
[poly\(\)](#) (*amici.splines.CubicHermiteSpline* method), 403  
[poly\\_variable\(\)](#) (*amici.splines.AbstractSpline* method), 399  
[poly\\_variable\(\)](#) (*amici.splines.CubicHermiteSpline* method), 403  
[polynomial](#) (*amici.amici.InterpolationType* attribute), 278  
[posteq\\_cpu\\_time](#) (*amici.amici.ReturnData* property), 304  
[posteq\\_cpu\\_time](#) (*amici.amici.ReturnDataPtr* property), 310  
[posteq\\_cpu\\_timeB](#) (*amici.amici.ReturnData* property), 304  
[posteq\\_cpu\\_timeB](#) (*amici.amici.ReturnDataPtr* property), 310  
[posteq\\_numsteps](#) (*amici.amici.ReturnData* property), 304  
[posteq\\_numsteps](#) (*amici.amici.ReturnDataPtr* property), 310  
[posteq\\_numstepsB](#) (*amici.amici.ReturnData* property), 304  
[posteq\\_numstepsB](#) (*amici.amici.ReturnDataPtr* property), 310  
[posteq\\_status](#) (*amici.amici.ReturnData* property), 304  
[posteq\\_status](#) (*amici.amici.ReturnDataPtr* property), 310  
[posteq\\_t](#) (*amici.amici.ReturnData* property), 304  
[posteq\\_t](#) (*amici.amici.ReturnDataPtr* property), 310  
[posteq\\_wrms](#) (*amici.amici.ReturnData* property), 304  
[posteq\\_wrms](#) (*amici.amici.ReturnDataPtr* property), 310  
[preeq\\_cpu\\_time](#) (*amici.amici.ReturnData* property), 304  
[preeq\\_cpu\\_time](#) (*amici.amici.ReturnDataPtr* property), 310  
[preeq\\_cpu\\_timeB](#) (*amici.amici.ReturnData* property), 304  
[preeq\\_cpu\\_timeB](#) (*amici.amici.ReturnDataPtr* property), 310  
[preeq\\_numsteps](#) (*amici.amici.ReturnData* property), 305  
[preeq\\_numsteps](#) (*amici.amici.ReturnDataPtr* property), 310  
[preeq\\_numstepsB](#) (*amici.amici.ReturnData* property), 305  
[preeq\\_numstepsB](#) (*amici.amici.ReturnDataPtr* property), 311  
[preeq\\_status](#) (*amici.amici.ReturnData* property), 305  
[preeq\\_status](#) (*amici.amici.ReturnDataPtr* property), 311  
[preeq\\_t](#) (*amici.amici.ReturnData* property), 305  
[preeq\\_t](#) (*amici.amici.ReturnDataPtr* property), 311  
[preeq\\_wrms](#) (*amici.amici.ReturnData* property), 305  
[preeq\\_wrms](#) (*amici.amici.ReturnDataPtr* property), 311  
[preequilibration](#), 55  
[preequilibration](#) (*amici.amici.FixedParameterContext* attribute), 278  
[presimulation](#), 55  
[presimulation](#) (*amici.amici.FixedParameterContext* attribute), 278  
[pretty\\_xml\(\)](#) (in module *amici.sbml\_utils*), 395  
[printmethod](#) (*amici.sbml\_utils.MathMLSBmlPrinter* attribute), 392  
[process\\_conservation\\_laws\(\)](#) (*amici.sbml\_import.SbmlImporter* method), 392

328  
 pscale (*amici.amici.ExpData* property), 273  
 pscale (*amici.amici.ExpDataPtr* property), 277  
 pscale (*amici.amici.ReturnData* property), 305  
 pscale (*amici.amici.ReturnDataPtr* property), 311  
 pscale (*amici.amici.SimulationParameters* property), 315  
 PySB, 55  
 pysb2amici() (in module *amici.pysb\_import*), 332  
 pysb\_model\_from\_path() (in module *amici.pysb\_import*), 333  
 pythonGenerated (*amici.amici.Model* property), 291  
 pythonGenerated (*amici.amici.ModelPtr* property), 299

## R

rdata\_reporting (*amici.amici.ReturnData* property), 305  
 rdata\_reporting (*amici.amici.ReturnDataPtr* property), 311  
 RDataReporting (class in *amici.amici*), 300  
 rdatas\_to\_measurement\_df() (in module *amici.petab\_objective*), 342  
 rdatas\_to\_simulation\_df() (in module *amici.petab\_objective*), 342  
 reinitialization\_state\_idxs\_presim (*amici.amici.ExpData* property), 274  
 reinitialization\_state\_idxs\_presim (*amici.amici.ExpDataPtr* property), 277  
 reinitialization\_state\_idxs\_presim (*amici.amici.SimulationParameters* property), 315  
 reinitialization\_state\_idxs\_sim (*amici.amici.ExpData* property), 274  
 reinitialization\_state\_idxs\_sim (*amici.amici.ExpDataPtr* property), 277  
 reinitialization\_state\_idxs\_sim (*amici.amici.SimulationParameters* property), 315  
 reinitializeAllFixedParameterDependentInitialStates() (*amici.amici.ExpData* method), 274  
 reinitializeAllFixedParameterDependentInitialStates() (*amici.amici.SimulationParameters* method), 315  
 reinitializeAllFixedParameterDependentInitialStatesForPresimulation() (*amici.amici.ExpData* method), 274  
 reinitializeAllFixedParameterDependentInitialStatesForPresimulation() (*amici.amici.SimulationParameters* method), 315  
 reinitializeAllFixedParameterDependentInitialStatesForSimulation() (*amici.amici.ExpData* method), 274  
 reinitializeAllFixedParameterDependentInitialStatesForSimulation() (*amici.amici.SimulationParameters* method), 315

reinitializeFixedParameterInitialStates (*amici.amici.ExpData* property), 274  
 reinitializeFixedParameterInitialStates (*amici.amici.ExpDataPtr* property), 277  
 reinitializeFixedParameterInitialStates (*amici.amici.SimulationParameters* property), 315  
 remove\_argument\_types() (in module *amici.de\_export*), 361  
 remove\_working\_dir() (*amici.petab\_simulate.PetabSimulator* method), 344  
 replace\_logx() (in module *amici.sbml\_import*), 330  
 requireSensitivitiesForAllParameters() (*amici.amici.Model* method), 291  
 res (*amici.amici.ReturnData* property), 305  
 res (*amici.amici.ReturnDataPtr* property), 311  
 rescale\_sensitivity() (in module *amici.petab\_objective*), 342  
 reset() (*amici.amici.CpuTimer* method), 268  
 residuals (*amici.amici.RDataReporting* attribute), 300  
 restore\_patch() (*amici.sbml\_utils.MathMLSBmlPrinter* method), 392  
 ReturnData (class in *amici.amici*), 300  
 ReturnDataPtr (class in *amici.amici*), 307  
 ReturnDataView (class in *amici.numpy*), 389  
 rowvals() (*amici.de\_export.DEModel* method), 358  
 rref() (in module *amici.conservated\_quantities\_rref*), 388  
 runAmiciSimulation() (in module *amici.amici*), 325  
 runAmiciSimulations() (in module *amici.amici*), 325  
 rz (*amici.amici.ReturnData* property), 305  
 rz (*amici.amici.ReturnDataPtr* property), 311

## S

s2llh (*amici.amici.ReturnData* property), 305  
 s2llh (*amici.amici.ReturnDataPtr* property), 311  
 s2rz (*amici.amici.ReturnData* property), 305  
 s2rz (*amici.amici.ReturnDataPtr* property), 311  
 state\_substitute() (*amici.de\_export.TemplateAmici* method), 360  
 sbml2amici() (*amici.sbml\_import.SbmlImporter* method), 328  
 spline\_formula (*amici.splines.AbstractSpline* property), 399  
 spline\_formula (*amici.splines.CubicHermiteSpline* property), 403  
 sbml\_id (*amici.splines.AbstractSpline* property), 399  
 spline\_id (*amici.splines.CubicHermiteSpline* property), 403  
 sbml\_id (*amici.splines.CubicHermiteSpline* property), 403  
 sbml\_id (*amici.splines.CubicHermiteSpline* property), 403  
 sbml\_mathml() (in module *amici.sbml\_utils*), 395  
 SbmlAnnotationError, 392

SbmlDuplicateComponentIdError, 392  
 SBMLException, 346  
 SbmlImporter (class in amici.sbml\_import), 326  
 SbmlInvalidIdSyntax, 392  
 SbmlMathError, 392  
 SbmlMissingComponentIdError, 393  
 scale\_parameter() (in module amici.parameter\_mapping), 386  
 scale\_parameters\_dict() (in module amici.parameter\_mapping), 386  
 scaleParameters() (in module amici.amici), 325  
 second (amici.amici.SensitivityOrder attribute), 313  
 second\_derivative() (amici.splines.AbstractSpline method), 399  
 second\_derivative() (amici.splines.CubicHermiteSpline method), 403  
 SecondOrderMode (class in amici.amici), 313  
 segment\_formula() (amici.splines.AbstractSpline method), 399  
 segment\_formula() (amici.splines.CubicHermiteSpline method), 403  
 sensi (amici.amici.ReturnData property), 305  
 sensi (amici.amici.ReturnDataPtr property), 311  
 sensi\_meth (amici.amici.ReturnData property), 305  
 sensi\_meth (amici.amici.ReturnDataPtr property), 311  
 SensitivityMethod (class in amici.amici), 313  
 SensitivityMethod\_adjoint (in module amici.amici), 313  
 SensitivityMethod\_forward (in module amici.amici), 313  
 SensitivityMethod\_none (in module amici.amici), 313  
 SensitivityOrder (class in amici.amici), 313  
 SensitivityOrder\_first (in module amici.amici), 313  
 SensitivityOrder\_none (in module amici.amici), 313  
 SensitivityOrder\_second (in module amici.amici), 313  
 set\_conservation\_law() (amici.de\_model.DifferentialState method), 367  
 set\_dt() (amici.de\_model.DifferentialState method), 368  
 set\_global\_settings() (amici.sbml\_utils.MathMLSbmlPrinter class method), 392  
 set\_log\_level() (in module amici.logging), 382  
 set\_name() (amici.de\_export.DEExporter method), 352  
 set\_paths() (amici.de\_export.DEExporter method), 352  
 set\_sbml\_math() (in module amici.sbml\_utils), 395  
 set\_val() (amici.de\_model.AlgebraicEquation method), 364  
 set\_val() (amici.de\_model.AlgebraicState method), 365  
 set\_val() (amici.de\_model.ConservationLaw method), 366  
 set\_val() (amici.de\_model.Constant method), 366  
 set\_val() (amici.de\_model.DifferentialState method), 368  
 set\_val() (amici.de\_model.Event method), 369  
 set\_val() (amici.de\_model.EventObservable method), 370  
 set\_val() (amici.de\_model.Expression method), 370  
 set\_val() (amici.de\_model.LogLikelihoodRZ method), 371  
 set\_val() (amici.de\_model.LogLikelihoodY method), 371  
 set\_val() (amici.de\_model.LogLikelihoodZ method), 372  
 set\_val() (amici.de\_model.ModelQuantity method), 373  
 set\_val() (amici.de\_model.Observable method), 374  
 set\_val() (amici.de\_model.Parameter method), 374  
 set\_val() (amici.de\_model.SigmaY method), 375  
 set\_val() (amici.de\_model.SigmaZ method), 376  
 set\_val() (amici.de\_model.State method), 377  
 setAbsoluteTolerance() (amici.amici.Solver method), 320  
 setAbsoluteToleranceB() (amici.amici.Solver method), 320  
 setAbsoluteToleranceFSA() (amici.amici.Solver method), 320  
 setAbsoluteToleranceQuadratures() (amici.amici.Solver method), 320  
 setAbsoluteToleranceSteadyState() (amici.amici.Solver method), 321  
 setAbsoluteToleranceSteadyStateSensi() (amici.amici.Solver method), 321  
 setAddSigmaResiduals() (amici.amici.Model method), 291  
 setAllStatesNonNegative() (amici.amici.Model method), 291  
 setAlwaysCheckFinite() (amici.amici.Model method), 291  
 setFixedParameterById() (amici.amici.Model method), 291  
 setFixedParameterByName() (amici.amici.Model method), 292  
 setFixedParameters() (amici.amici.Model method), 292  
 setFixedParametersByIdRegex() (amici.amici.Model method), 292  
 setFixedParametersByNameRegex() (amici.amici.Model method), 292  
 setInitialStates() (amici.amici.Model method), 292  
 setInitialStateSensitivities() (am-

*ici.amici.Model* method), 292

setInternalSensitivityMethod() (*amici.amici.Solver* method), 321

setInterpolationType() (*amici.amici.Solver* method), 321

setLinearMultistepMethod() (*amici.amici.Solver* method), 321

setLinearSolver() (*amici.amici.Solver* method), 321

setMaxSteps() (*amici.amici.Solver* method), 321

setMaxStepsBackwardProblem() (*amici.amici.Solver* method), 321

setMaxTime() (*amici.amici.Solver* method), 321

setMinimumSigmaResiduals() (*amici.amici.Model* method), 292

setNewtonDampingFactorLowerBound() (*amici.amici.Solver* method), 321

setNewtonDampingFactorMode() (*amici.amici.Solver* method), 321

setNewtonMaxSteps() (*amici.amici.Solver* method), 321

setNewtonStepSteadyStateCheck() (*amici.amici.Solver* method), 322

setNMaxEvent() (*amici.amici.Model* method), 292

setNonlinearSolverIteration() (*amici.amici.Solver* method), 322

setObservedData() (*amici.amici.ExpData* method), 274

setObservedDataStdDev() (*amici.amici.ExpData* method), 274

setObservedEvents() (*amici.amici.ExpData* method), 275

setObservedEventsStdDev() (*amici.amici.ExpData* method), 276

setParameterById() (*amici.amici.Model* method), 293

setParameterByName() (*amici.amici.Model* method), 293

setParameterList() (*amici.amici.Model* method), 294

setParameters() (*amici.amici.Model* method), 294

setParametersByIdRegex() (*amici.amici.Model* method), 294

setParametersByNameRegex() (*amici.amici.Model* method), 294

setParameterScale() (*amici.amici.Model* method), 294

setReinitializationStateIdxs() (*amici.amici.Model* method), 294

setReinitializeFixedParameterInitialStates() (*amici.amici.Model* method), 294

setRelativeTolerance() (*amici.amici.Solver* method), 322

setRelativeToleranceB() (*amici.amici.Solver* method), 322

setRelativeToleranceFSA() (*amici.amici.Solver* method), 322

setRelativeToleranceQuadratures() (*amici.amici.Solver* method), 322

setRelativeToleranceSteadyState() (*amici.amici.Solver* method), 322

setRelativeToleranceSteadyStateSensi() (*amici.amici.Solver* method), 322

setReturnDataReportingMode() (*amici.amici.Solver* method), 322

setSensiSteadyStateCheck() (*amici.amici.Solver* method), 322

setSensitivityMethod() (*amici.amici.Solver* method), 322

setSensitivityMethodPreequilibration() (*amici.amici.Solver* method), 323

setSensitivityOrder() (*amici.amici.Solver* method), 323

setStabilityLimitFlag() (*amici.amici.Solver* method), 323

setStateIsNonNegative() (*amici.amici.Model* method), 294

setStateOrdering() (*amici.amici.Solver* method), 323

setSteadyStateComputationMode() (*amici.amici.Model* method), 294

setSteadyStateSensitivityMode() (*amici.amici.Model* method), 295

setSteadyStateSensiToleranceFactor() (*amici.amici.Solver* method), 323

setSteadyStateToleranceFactor() (*amici.amici.Solver* method), 323

setT0() (*amici.amici.Model* method), 295

setTimepoints() (*amici.amici.ExpData* method), 276

setTimepoints() (*amici.amici.Model* method), 295

setUnscaledInitialStateSensitivities() (*amici.amici.Model* method), 295

severity (*amici.amici.LogItem* property), 279

show\_model\_info() (in module *amici.petab\_import*), 338

sigma\_res (*amici.amici.ReturnData* property), 305

sigma\_res (*amici.amici.ReturnDataPtr* property), 311

sigma\_ys() (*amici.de\_export.DEModel* method), 358

sigma\_zs() (*amici.de\_export.DEModel* method), 358

sigmay (*amici.amici.ReturnData* property), 305

sigmay (*amici.amici.ReturnDataPtr* property), 311

SigmaY (class in *amici.de\_model*), 374

sigmaz (*amici.amici.ReturnData* property), 305

sigmaz (*amici.amici.ReturnDataPtr* property), 311

SigmaZ (class in *amici.de\_model*), 375

simulate() (*amici.petab\_simulate.PetabSimulator* method), 344

simulate\_petab() (in module *amici.petab\_objective*), 342

simulate\_without\_noise() (*amici.petab\_simulate.PetabSimulator* method), 345



simulation (*amici.amici.FixedParameterContext* attribute), 278  
 simulation\_status\_to\_str() (in module *amici.amici*), 325  
 SimulationParameters (class in *amici.amici*), 313  
 simultaneous (*amici.amici.InternalSensitivityMethod* attribute), 278  
 sllh (*amici.amici.ReturnData* property), 305  
 sllh (*amici.amici.ReturnDataPtr* property), 311  
 smart\_is\_zero\_matrix() (in module *amici.de\_export*), 361  
 smart\_jacobian() (in module *amici.de\_export*), 361  
 smart\_multiply() (in module *amici.de\_export*), 362  
 smart\_subs() (in module *amici.import\_utils*), 349  
 smart\_subs\_dict() (in module *amici.import\_utils*), 349  
 smoothness (*amici.splines.AbstractSpline* property), 400  
 smoothness (*amici.splines.CubicHermiteSpline* property), 403  
 Solver (class in *amici.amici*), 316  
 SolverPtr (class in *amici.amici*), 323  
 sparseeq() (*amici.de\_export.DEModel* method), 358  
 sparsesym() (*amici.de\_export.DEModel* method), 358  
 SPBCG (*amici.amici.LinearSolver* attribute), 279  
 species\_to\_parameters() (in module *amici.petab\_import*), 338  
 SPGMR (*amici.amici.LinearSolver* attribute), 279  
 spline\_user\_functions() (in module *amici.splines*), 404  
 SPTFQMR (*amici.amici.LinearSolver* attribute), 279  
 squared\_L2\_norm\_of\_curvature() (*amici.splines.AbstractSpline* method), 400  
 squared\_L2\_norm\_of\_curvature() (*amici.splines.CubicHermiteSpline* method), 403  
 sres (*amici.amici.ReturnData* property), 305  
 sres (*amici.amici.ReturnDataPtr* property), 311  
 srz (*amici.amici.ReturnData* property), 306  
 srz (*amici.amici.ReturnDataPtr* property), 311  
 ssigmay (*amici.amici.ReturnData* property), 306  
 ssigmay (*amici.amici.ReturnDataPtr* property), 312  
 ssigmaz (*amici.amici.ReturnData* property), 306  
 ssigmaz (*amici.amici.ReturnDataPtr* property), 312  
 staggered (*amici.amici.InternalSensitivityMethod* attribute), 278  
 staggered1 (*amici.amici.InternalSensitivityMethod* attribute), 278  
 start (*amici.splines.UniformGrid* property), 404  
 State (class in *amici.de\_model*), 376  
 state\_has\_conservation\_law() (*amici.de\_export.DEModel* method), 358  
 state\_has\_fixed\_parameter\_initial\_condition() (*amici.de\_export.DEModel* method), 359  
 state\_is\_constant() (*amici.de\_export.DEModel* method), 359  
 states() (*amici.de\_export.DEModel* method), 359  
 status (*amici.amici.ReturnData* property), 306  
 status (*amici.amici.ReturnDataPtr* property), 312  
 SteadyStateComputationMode (class in *amici.amici*), 323  
 SteadyStateSensitivityMode (class in *amici.amici*), 323  
 SteadyStateStatus (class in *amici.amici*), 324  
 SteadyStateStatusVector (class in *amici.amici*), 324  
 step (*amici.splines.UniformGrid* property), 404  
 stop (*amici.splines.UniformGrid* property), 404  
 StringDoubleMap (class in *amici.amici*), 324  
 StringVector (class in *amici.amici*), 324  
 strip\_pysb() (in module *amici.import\_utils*), 349  
 substitute() (*amici.de\_export.TemplateAmici* method), 360  
 success (*amici.amici.SteadyStateStatus* attribute), 324  
 SUNDIALS, 55  
 SuperLUMT (*amici.amici.LinearSolver* attribute), 279  
 SWIG, 55  
 SwigPtrView (class in *amici.numpy*), 390  
 sx (*amici.amici.ReturnData* property), 306  
 sx (*amici.amici.ReturnDataPtr* property), 312  
 sx0 (*amici.amici.ExpData* property), 276  
 sx0 (*amici.amici.ExpDataPtr* property), 277  
 sx0 (*amici.amici.ReturnData* property), 306  
 sx0 (*amici.amici.ReturnDataPtr* property), 312  
 sx0 (*amici.amici.SimulationParameters* property), 315  
 sx\_ss (*amici.amici.ReturnData* property), 306  
 sx\_ss (*amici.amici.ReturnDataPtr* property), 312  
 sy (*amici.amici.ReturnData* property), 306  
 sy (*amici.amici.ReturnDataPtr* property), 312  
 sym() (*amici.de\_export.DEModel* method), 359  
 sym\_names() (*amici.de\_export.DEModel* method), 359  
 sym\_or\_eq() (*amici.de\_export.DEModel* method), 359  
 symbol\_with\_assumptions() (in module *amici.import\_utils*), 349  
 sympify\_noeval() (in module *amici.splines*), 404  
 sz (*amici.amici.ReturnData* property), 306  
 sz (*amici.amici.ReturnDataPtr* property), 312

## T

t0() (*amici.amici.Model* method), 295  
 t\_presim (*amici.amici.ExpData* property), 277  
 t\_presim (*amici.amici.ExpDataPtr* property), 278  
 t\_presim (*amici.amici.SimulationParameters* property), 315  
 TemplateAmici (class in *amici.de\_export*), 359  
 toposort\_symbols() (in module *amici.import\_utils*), 349  
 ts (*amici.amici.ReturnData* property), 306  
 ts (*amici.amici.ReturnDataPtr* property), 312  
 ts\_ (*amici.amici.ExpData* property), 277

`ts_` (*amici.amici.ExpDataPtr* property), 278  
`ts_` (*amici.amici.SimulationParameters* property), 315  
`tstart_` (*amici.amici.ExpData* property), 277  
`tstart_` (*amici.amici.ExpDataPtr* property), 278  
`tstart_` (*amici.amici.SimulationParameters* property), 315

## U

`ubw` (*amici.amici.Model* property), 295  
`ubw` (*amici.amici.ModelDimensions* property), 298  
`ubw` (*amici.amici.ModelPtr* property), 299  
`ubw` (*amici.amici.ReturnData* property), 306  
`ubw` (*amici.amici.ReturnDataPtr* property), 312  
`UniformGrid` (class in *amici.splines*), 403  
`unscale_parameter()` (in module *amici.parameter\_mapping*), 387  
`unscale_parameters_dict()` (in module *amici.parameter\_mapping*), 387  
`unscaleParameters()` (in module *amici.amici*), 326  
`uses_thread_clock` (*amici.amici.CpuTimer* attribute), 268

## V

`val()` (*amici.de\_export.DEModel* method), 359  
`values()` (*amici.numpy.ExpDataView* method), 389  
`values()` (*amici.numpy.ReturnDataView* method), 390  
`values()` (*amici.numpy.SwigPtrView* method), 390  
`values_at_nodes` (*amici.splines.AbstractSpline* property), 400  
`values_at_nodes` (*amici.splines.CubicHermiteSpline* property), 403  
`var_in_function_signature()` (in module *amici.de\_export*), 362

## W

`w` (*amici.amici.ReturnData* property), 306  
`w` (*amici.amici.ReturnDataPtr* property), 312  
`with_traceback()` (*amici.AmiciVersionError* method), 264  
`with_traceback()` (*amici.import\_utils.CircularDependencyError* method), 346  
`with_traceback()` (*amici.import\_utils.SBMLException* method), 346  
`with_traceback()` (*amici.sbml\_utils.SbmlAnnotationError* method), 392  
`with_traceback()` (*amici.sbml\_utils.SbmlDuplicateComponentIdError* method), 392  
`with_traceback()` (*amici.sbml\_utils.SbmlInvalidIdSyntax* method), 392

`with_traceback()` (*amici.sbml\_utils.SbmlMathError* method), 393  
`with_traceback()` (*amici.sbml\_utils.SbmlMissingComponentIdError* method), 393

## X

`x` (*amici.amici.ReturnData* property), 306  
`x` (*amici.amici.ReturnDataPtr* property), 312  
`x0` (*amici.amici.ExpData* property), 277  
`x0` (*amici.amici.ExpDataPtr* property), 278  
`x0` (*amici.amici.ReturnData* property), 306  
`x0` (*amici.amici.ReturnDataPtr* property), 312  
`x0` (*amici.amici.SimulationParameters* property), 316  
`x_ss` (*amici.amici.ReturnData* property), 306  
`x_ss` (*amici.amici.ReturnDataPtr* property), 312  
`xdot` (*amici.amici.ReturnData* property), 306  
`xdot` (*amici.amici.ReturnDataPtr* property), 312

## Y

`y` (*amici.amici.ReturnData* property), 307  
`y` (*amici.amici.ReturnDataPtr* property), 312  
`y_scaled()` (*amici.splines.AbstractSpline* method), 400  
`y_scaled()` (*amici.splines.CubicHermiteSpline* method), 403

## Z

`z` (*amici.amici.ReturnData* property), 307  
`z` (*amici.amici.ReturnDataPtr* property), 313