Bioinformatics, 37(20), 2021, 3676–3677
doi: 10.1093/bioinformatics/btab227
Advance Access Publication Date: 2 April 2021
Applications Note

Systems biology

AMICI: high-performance sensitivity analysis for large ordinary differential equation models

Fabian Fröhlich[1,*], Daniel Weindl [2], Yannik Schälte[2,3], Dilan Pathirana[4], Łukasz Paszkowski[5], Glenn Terje Lines[5], Paul Stapor[2,3] and Jan Hasenauer [2,3,4,*]

[1]Department of Systems Biology, Harvard Medical School, Boston, MA 02115, USA, [2]Institute of Computational Biology, Helmholtz Zentrum München – German Research Center for Environmental Health, Neuherberg 85764, Germany, [3]Department of Mathematics, Technische Universität München, Garching 85748, Germany, [4]Faculty of Mathematics and Natural Sciences, University of Bonn, Bonn 53113, Germany and [5]Simula Research, Lysaker 1325, Norway

*To whom correspondence should be addressed.
Associate Editor: Anthony Mathelier

Received on December 24, 2020; revised on March 18, 2021; editorial decision on March 24, 2021; accepted on April 1, 2021

Abstract

Summary: Ordinary differential equation models facilitate the understanding of cellular signal transduction and other biological processes. However, for large and comprehensive models, the computational cost of simulating or calibrating can be limiting. AMICI is a modular toolbox implemented in C++/Python/MATLAB that provides efficient simulation and sensitivity analysis routines tailored for scalable, gradient-based parameter estimation and uncertainty quantification.

Availabilityand implementation: AMICI is published under the permissive BSD-3-Clause license with source code publicly available on https://github.com/AMICI-dev/AMICI. Citeable releases are archived on Zenodo.

Contact: jan.hasenauer@uni-bonn.de or fabian_froehlich@hms.harvard.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Ordinary Differential Equation (ODE) models are widely used in systems biology, for example, to elucidate dynamic processes and to predict response to perturbations. Model parameters have to be inferred from data, which can be computationally intensive as thousands of simulations may be required. This is challenging for large models, with many state variables and parameters, where simulations take seconds to minutes (Fröhlich *et al.*, 2018). Inference can benefit from accurate sensitivities (Villaverde *et al.*, 2018), i.e., derivatives of model outputs with respect to model parameters. Accurate sensitivities can be computed using forward, adjoint or steady-state approaches, but benefit from symbolic derivatives of model equations, which are labor-intensive and error-prone to compute manually.

The recent surge of genome-scale perturbation data, as well as respective comprehensive models, has increased the demand for methods for scalable sensitivity computation for ODE models. To address this demand, we introduce AMICI, a high-performance simulation and sensitivity analysis library. AMICI is implemented in C++ and Python, and provides model import from widely used formats such as the Systems Biology Markup Language (SBML) (Hucka *et al.*, 2003), BioNetGen Language (BNGL) (Harris *et al.*, 2016) and Kappa (Boutillier *et al.*, 2018), and generates high-performance-computing (HPC) ready modules. These modules provide model-specific simulation and sensitivity computation routines, which can be accessed from Python, C++ and MATLAB. For parameter estimation problems specified in PEtab (Schmiester *et al.*, 2021), AMICI can evaluate the objective function and its gradient.

2 Materials and methods

For high simulation performance, AMICI reads models from high-level formats, then translates the model and symbolically derived expressions to C++ code. As symbolic processing can be computationally intensive, AMICI symbolically only computes partial derivatives; total derivatives are computed through (sparse) matrix multiplication and addition at runtime. This expedites model compilation and simulation.

To simulate stiff, large models, efficient linear solvers are crucial. AMICI features several direct dense, direct sparse and implicit solvers. For sensitivity analysis, AMICI implements forward, adjoint (Fröhlich *et al.*, 2017) and steady-state (Lines *et al.*, 2019) methods and combinations thereof (see Supplementary Table S1). To enable simulation of complex experimental setups, such as growth factor addition or drug washouts, AMICI supports pre-equilibration and pre-simulation with distinct condition-specific parameters.

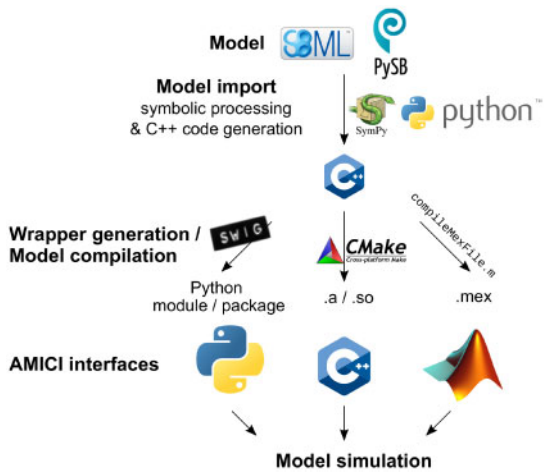

Fig. 1. Outline of model compilation in AMICI. Models, specified in SBML or PySB, are imported in Python. This generates efficient C++ code for model equations and simulation bindings for Python, C++ and MATLAB

3 Implementation

The AMICI library is implemented in C++14 and relies on SUNDIALS (Hindmarsh *et al.*, 2005) and SuiteSparse (Davis and Palamadai Natarajan, 2010) for simulation and sensitivity computation.

Model import is implemented in Python and supports the widely used formats (Fig. 1) SBML [via libSBML (Bornstein *et al.*, 2008)] and BNGL/Kappa [via PySB (Lopez *et al.*, 2013)]. Symbolic processing is performed using SymPy (Meurer *et al.*, 2017). Compilation has been tested with GCC, Clang, Intel and MinGW on Linux, MacOS and Windows platforms. For interoperability, AMICI provides CMake files and Dockerfiles.

AMICI ships with a MATLAB interface and implements a Python interface using SWIG, which can be extended to other languages. These interfaces execute simulation and sensitivity computation through the C++ library, enabling high performance, competitive with other toolboxes such as COPASI (Städter *et al.*, 2021). Simulation and compilation are highly configurable with the API documented on Read the Docs (https://amici.readthedocs.io/) and in example notebooks. For HPC-readiness, AMICI implements OpenMP parallelization over experimental conditions and is thread-safe.

Simulation and sensitivity analysis rely on intricate theory that is complex, error-prone to implement. Accordingly, we implemented an extensive continuous integration test pipeline. Simulation results are verified for the SBML semantic test suite, where AMICI passes 996 out of 1780 tests (appropriate error messages to indicate unsupported features for remaining tests). Simulation results are compared against PySB simulations for 17 BNGL validation models and examples, where AMICI passes all tests. Sensitivity results are checked by regression tests covering, e.g. forward, adjoint and steady-state sensitivities. Performance tests check computation time for model import, simulation and sensitivity analysis of a large model. Unit and documentation tests, static code analysis and memory leak checks are included, and code review is enforced for all contributions.

4 Discussion

There is a rich ecosystem of tools for model simulation. To avoid duplication, we ensured good interoperability with other tools: AMICI does not provide a model development environment, but permits model import from standard formats. Similarly, AMICI is not part of an integrated parameter estimation framework, but features a flexible, well-documented API. Currently, several parameter estimation tools can interface with AMICI, including pyPESTO (Schälte *et al.*, 2020) and parPE (Schmiester *et al.*, 2020). This modular design is geared toward researchers developing new methods or tools for parameter estimation that would benefit from high-performance simulation and sensitivity computation routines. Specifically, adjoint and steady-state sensitivity analysis as well as sparse linear solvers are currently only supported in a small, disparate set of tools, which highlights the unique capabilities of AMICI (see Supplementary Table S1).

AMICI has been in development since 2015 and has so far been used in at least 50 publications, and is continuously being developed by 4 core contributors at 3 different institutions. In the future, we plan to improve SBML support and extend interoperability with other tools.

Funding

This work was supported by the European Union's Horizon 2020 research and innovation program [CanPathPro; 686282 to J.H., D.W., P.S., Ł.P., G.T.L., F.F.], the Federal Ministry of Education and Research of Germany [01ZX1916A to D.W., 01ZX1705A to J.H., 031L0159C to J.H.], the German Research Foundation [HA7376/1-1 to Y.S., Germany's Excellence Strategy—EXC-2047/1–390685813 to D.P.], the Human Frontier Science Program [LT000259/2019-L1 to F.F.], the National Institute of Health [U54-CA225088 to F.F.] and the Federal Ministry of Economic Affairs and Energy [16KN074236 to D.P.].

Conflict of Interest: none declared.

References

Bornstein,B.J. *et al.* (2008) LibSBML: an API library for SBML. *Bioinformatics*, **24**, 880–881.

Boutillier,P. *et al.* (2018) The Kappa platform for rule-based modeling. *Bioinformatics*, **34**, i583–i592.

Davis,T.A. and Palamadai Natarajan,E. (2010) Algorithm 907: KLU, a direct sparse solver for circuit simulation problems. *ACM Trans. Math. Softw.*, **37**, 1.

Fröhlich,F. *et al.* (2018) Efficient parameter estimation enables the prediction of drug response using a mechanistic pan-cancer pathway model. *Cell Syst.*, **7**, 567–579.e6.

Fröhlich,F. *et al.* (2017) Scalable parameter estimation for genome-scale biochemical reaction networks. *PLoS Comput. Biol.*, **13**, e1005331.

Harris,L.A. *et al.* (2016) BioNetGen 2.2: advances in rule-based modeling. *Bioinformatics*, **32**, 3366–3368.

Hindmarsh,A.C. *et al.* (2005) SUNDIALS: suite of nonlinear and differential/algebraic equation solvers. *ACM Trans. Math. Softw.*, **31**, 363–396.

Hucka,M. *et al.*; SBML Forum. (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, **19**, 524–531.

Lines,G.T. *et al.* (2019) Efficient computation of steady states in large-scale ode models of biochemical reaction networks. IFAC-PapersOnLine. In: *8th Conference on Foundations of Systems Biology in Engineering FOSBE 2019*, Vol. 52, pp. 32–37.

Lopez,C.F. *et al.* (2013) Programming biological models in python using pysb. *Mol. Syst. Biol.*, **9**, 646.

Meurer,A. *et al.* (2017) SymPy: symbolic computing in Python. *PeerJ Comput. Sci.*, **3**, e103.

Schälte,Y. *et al.* (2020) ICB-DCM/pyPESTO: pyPESTO 0.0.11. doi: 10.5281/zenodo.3715448.

Schmiester,L. *et al.* (2020) Efficient parameterization of large-scale dynamic models based on relative measurements. *Bioinformatics*, **36**, 594–602.

Schmiester,L. *et al.* (2021) PEtab—interoperable specification of parameter estimation problems in systems biology. *PLoS Comput. Biol.*, **17**, e1008646.

Städter,P. *et al.* (2021) Benchmarking of numerical integration methods for ODE models of biological systems. *Sci. Rep.*, **11**, 2696.

Villaverde,A.F. *et al.* (2018) Benchmarking optimization methods for parameter estimation in large kinetic models. *Bioinformatics*, **35**, bty736.