The Journal of Open Source Software

pyodesys: Straightforward numerical integration of ODE systems from Python

Björn Dahlgren[1]

1 KTH Royal Institute of Technology

DOI: 10.21105/joss.00490

Software

- Review
- Repository
- Archive

Submitted: 05 December 2017
Published: 03 January 2018

Summary

The numerical integration of systems of ordinary differential equations (ODEs) is very common in most scientific fields. There exists a large number of software libraries for solving these systems, each requiring the user to write their code in slightly different forms. Furthermore, it is sometimes necessary to perform variable transformations in order for the solution to proceed efficiently.

pyodesys provides a unified interface to some existing solvers. It also provides an interface to represent the system symbolically. This allows *pyodesys* to derive the Jacobian matrix symbolically (which is both tedious and error prone when done manually). In addition, this representation allows the user to manipulate the mathematical representation symbolically. This is achieved by using SymPy (Meurer et al. 2017) (although the coupling is loose and other symbolic backends may be used).

pyodesys enables the user to write his or her code once, and leave the library specific adaptions for *pyodesys* to handle. This allows the user to evaluate both different solvers (which implement different integration methods and algorithms for step size control) and alternative formulations of the system (from variable transformations, including scaling of variables).

Features

- Unified interface to ODE solvers from Sundials (Hindmarsh et al. 2005), GNU Scientific Library (Galassi 2009) and odeint (Ahnert et al. 2011) in boost.
- Convenince methods for working with solutions (plotting trajectories, interpolation, inspecting invariants).
- Automatic derivation of the Jacobian matrix for use with implicit steppers.
- Symbolic variable transformations in the system of ODEs.
- Symbolic reduction of degrees of freedom by variable elimination using linear invariants.
- Symbolic rewriting of system based on (possibly approximate) analytic solutions to a subset of the dependent variables.
- Code-generation (C++) and on-the-fly compilation for enhanced performance (including automatic common subexpression elimintation).
- Parallel execution for parameter variations (multithreading using OpenMP). This feature is only availble in conjuction with code-generation.

References

Ahnert, Karsten, Mario Mulansky, Theodore E. Simos, George Psihoyios, Ch. Tsitouras, and Zacharias Anastassi. 2011. "Odeint - Solving Ordinary Differential Equations in C++." In. AIP. https://doi.org/10.1063/1.3637934.

Galassi, Mark, ed. 2009. *GNU Scientific Library: Reference Manual.* 3. ed., for GSL version 1.12, 1. print. A GNU Manual. s.l.: Network Theory.

Hindmarsh, Alan C, Peter N Brown, Keith E Grant, Steven L Lee, Radu Serban, Dan E Shumaker, and Carol S Woodward. 2005. "SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers." *ACM Transactions on Mathematical Software (TOMS)* 31 (3). ACM:363–96.

Meurer, Aaron, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, et al. 2017. "SymPy: Symbolic Computing in Python." *PeerJ Computer Science* 3 (January). PeerJ:e103. https://doi.org/10.7717/peerj-cs.103.