The Journal of Open Source Software

pyneqsys: Solve symbolically defined systems of non-linear equations numerically

Björn Dahlgren[1]

1 KTH Royal Institute of Technology

DOI: 10.21105/joss.00531

Software

- Review
- Repository
- Archive

Submitted: 08 January 2018
Published: 22 January 2018

Summary

Solving systems of non-linear equations numerically is a common task in scientific modeling work. Many software libraries have the capability to solve these kinds of systems, however, each require slightly different forms of input. In addition, it is often important that the user formulates the system in a manner which is suitable for the numerical algorithm. Finding an effective formulation is often an iterative process, which is facilitated if the system can be transformed symbolically.

pyneqsys offers a common interface to a handful of solvers. It furthermore provides tools to input and work with such systems symbolically. Having a symbolic representation allows *pyneqsys* to automatically derive the Jacobian matrix, which is a task which is laborious and a source of error when performed by hand. By relying on a computer algebra system, *pyneqsys* allows the user to apply e.g. variable transformations or generate representations in LaTeX, MathML etc. By default SymPy (Meurer et al. 2017) is used as the symbolic back-end, but other libraries are also supported.

Adapting *pyneqsys* to use new third party solvers is straightforward and some example solvers are provided with the library. Together with its ability to perform variable transformations symbolically *pyneqsys* allows the users to write code for their problem *once* and then easily test different formulations and solvers. This greatly lowers the burden of validation and speeds-up the iterative finding of the best method for solving the problem.

Features

- Unified interface to the KINSOL solver from SUNDIALS (Hindmarsh et al. 2005), SciPy's solvers (Jones et al. 2001–2001--), levmar (Lourakis 2004), NLEQ2 (Weimann 1991) and mpmath (Johansson and others 2013).
- Convenience methods for solving and plotting solutions as parameters of the system are varied.
- Automatic derivation of the Jacobian matrix.
- Symbolic variable transformations.
- Symbolic removal of linearly dependent equations by rewriting (linear parts) in reduced row echelon form.
- Carrying over the solution as initial guess in parameter variations.
- Facility for defining meta-algorithms (e.g. solve the system for one formulation first, and refine the solution by solving it as another formulation).
- Solve non-linear systems containing conditional equations (different equations governing different domains).

References

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. https://doi.org/10.1145/1089014.1089020.

Johansson, Fredrik, and others. 2013. *Mpmath: A Python Library for Arbitrary-Precision Floating-Point Arithmetic (Version 0.18)*.

Jones, Eric, Travis Oliphant, Pearu Peterson, and others. 2001–2001--. "SciPy: Open Source Scientific Tools for Python." http://www.scipy.org/.

Lourakis, M.I.A. 2004. "Levmar: Levenberg-Marquardt Nonlinear Least Squares Algorithms in C/C++." http://www.ics.forth.gr/~lourakis/levmar/.

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.

Weimann, UN. 1991. "A Family of Newton Codes for Systems of Highly Nonlinear Equations." ZIB Technical Report TR-91-10, ZIB, Berlin, Germany.