JOSEcon

The Journal of Open Source Economics

Publishing economic software

Fedor Iskhakov
Australian National University

DSE2019 Chicago

Current practice is substandard

- At large, economists are not too sophisticated with software development
- Programming language, coding style and code is often inherited from the supervisors
- Inefficient code sharing
- “Reinventing the wheel” + replication of bugs
- Workflow can be greatly improved by learning from other disciplines

The brighter future

- Modularization, sharing and reuse of developed code
- Collection of canonical implementations
- Reliability through independent use by many researchers + means of communication
- Agreement on good software engineering practices
- ⇒ Accumulation of knowledge and skills in software development in/for economics

Value of good software engineering

“The systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software”

- Version control and the ability to undo and trace any problem back
- Version control and efficient collaboration
- Incremental development and continuous integration
- Unit testing and test driven development

Significant increase in productivity

A number of lone warriors

1. Econ-ARK econ-ark.org/
2. QuantEcon quantecon.org/
3. Projects run by Richard Evans
4. VFI toolkit by Robert Kirkby
5. Dolo language by Pablo Winant
6. NFXP code + manual by John Rust
7. OpenSourceEconomics by Philipp Eisenhauer
8. Atomized developers of open source code in economics (publishing specific projects code)

What are the right incentives

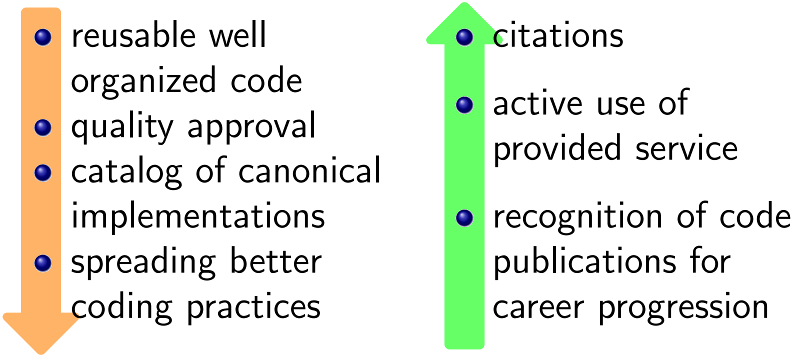
Profession does not acknowledge community work such as good code development in tenure process

PUBLICATIONS

- Add to publication list
- Collect citations
- Trouble-free publication process conditional on well written code
- CrossRef + DOI = proper publication

Community supported by the profession

Enthusiastic community of developers



- reusable well organized code
- quality approval
- catalog of canonical implementations
- spreading better coding practices

- citations
- active use of provided service
- recognition of code publications for career progression

Professional societies, journals, departments

Use cases

Computational economist/Developer

- Go extra mile in own code development
- Submit main paper in parallel with code paper

Student

- Replication project → learn + code publication
- Start own project from bug-free verified code

Editor/Referee

- Consult the community for quality check
- Outsource code review for accepted papers

Modern open source journals

Open Journals theoj.org

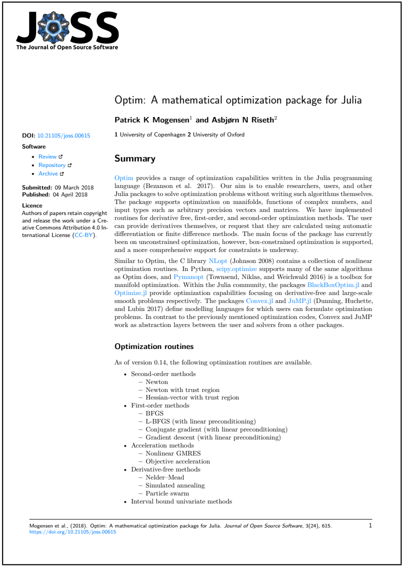
The J of Open Source Software

- The Journal of Open Source Education
- The Open Journal of Astrophysics
- The Journal of Brief Ideas

The Open Journals

- Collection of open source, open access journals
- Creates infrastructure for open access publishing under MIT license
- Fiscally sponsored project of the NumFOCUS
- Individual journals are responsible for their own editorial process, but must
    - Be open access
    - Have an open review process
    - Use the Open Journals open source toolchain

10.21105.joss.00615



The Journal of Open Source Software

Optim: A mathematical optimization package for Julia

Patrick K Mogensen[1] **and Asbjørn N Riseth**[2]

1 University of Copenhagen **2** University of Oxford

DOI: 10.21105/joss.00615

Software

- Review
- Repository
- Archive

Submitted: 09 March 2018
Published: 04 April 2018

Summary

Optim provides a range of optimization capabilities written in the Julia programming language (Bezanson et al. 2017). Our aim is to enable researchers, users, and other Julia packages to solve optimization problems without writing such algorithms themselves. The package supports optimization on manifolds, functions of complex numbers, and input types such as arbitrary precision vectors and matrices. We have implemented routines for derivative free, first-order, and second-order optimization methods. The user can provide derivatives themselves, or request that they are calculated using automatic differentiation or finite difference methods. The main focus of the package has currently been on unconstrained optimization, however, box-constrained optimization is supported, and a more comprehensive support for constraints is underway.

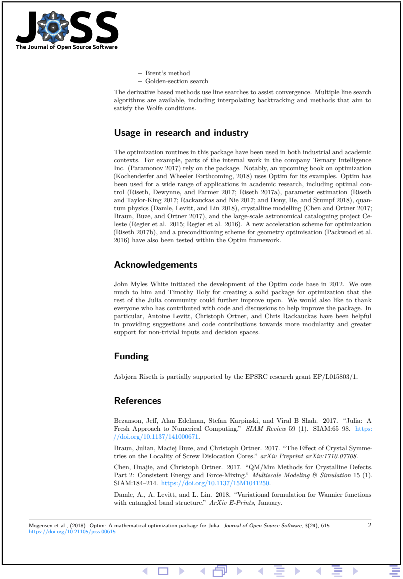
Similar to Optim, the C library NLopt (Johnson 2008) contains a collection of nonlinear optimization routines. In Python, scipy.optimize supports many of the same algorithms as Optim does, and Pymanopt (Townsend, Niklas, and Weichwald 2016) is a toolbox for manifold optimization. Within the Julia community, the packages BlackBoxOptim.jl and Optimize.jl provide optimization capabilities focusing on derivative-free and large-scale smooth problems respectively. The packages Convex.jl and JuMP.jl (Dunning, Huchette, and Lubin 2017) define modelling languages for which users can formulate optimization problems. In contrast to the previously mentioned optimization codes, Convex and JuMP work as abstraction layers between the user and solvers from a other packages.

Optimization routines

As of version 0.14, the following optimization routines are available.

- Second-order methods
  - Newton
  - Newton with trust region
  - Hessian-vector with trust region
- First-order methods
  - BFGS
  - L-BFGS (with linear preconditioning)
  - Conjugate gradient (with linear preconditioning)
  - Gradient descent (with linear preconditioning)
- Acceleration methods
  - Nonlinear GMRES
  - Objective acceleration
- Derivative-free methods
  - Nelder–Mead
  - Simulated annealing
  - Particle swarm
- Interval bound univariate methods

Mogensen et al., (2018). Optim: A mathematical optimization package for Julia. *Journal of Open Source Software*, 3(24), 615. 1



The Journal of Open Source Software

  - Brent's method
  - Golden-section search

The derivative based methods use line searches to assist convergence. Multiple line search algorithms are available, including interpolating backtracking and methods that aim to satisfy the Wolfe conditions.

Usage in research and industry

The optimization routines in this package have been used in both industrial and academic contexts. For example, parts of the internal work in the company Ternary Intelligence Inc. (Paramonov 2017) rely on the package. Notably, an upcoming book on optimization (Kochenderfer and Wheeler Forthcoming, 2018) uses Optim for its examples. Optim has been used for a wide range of applications in academic research, including optimal control (Riseth, Dewynne, and Farmer 2017; Riseth 2017a), parameter estimation (Riseth and Taylor-King 2017; Rackauckas and Nie 2017; and Dony, He, and Stumpf 2018), quantum physics (Damle, Levitt, and Lin 2018), crystalline modelling (Chen and Ortner 2017; Braun, Buze, and Ortner 2017), and the large-scale astronomical cataloguing project Celeste (Regier et al. 2015; Regier et al. 2016). A new acceleration scheme for optimization (Riseth 2017b), and a preconditioning scheme for geometry optimisation (Packwood et al. 2016) have also been tested within the Optim framework.

Acknowledgements

John Myles White initiated the development of the Optim code base in 2012. We owe much to him and Timothy Holy for creating a solid package for optimization that the rest of the Julia community could further improve upon. We would also like to thank everyone who has contributed with code and discussions to help improve the package. In particular, Antoine Levitt, Christoph Ortner, and Chris Rackauckas have been helpful in providing suggestions and code contributions towards more modularity and greater support for non-trivial inputs and decision spaces.

Funding

Asbjørn Riseth is partially supported by the EPSRC research grant EP/L015803/1.

References

Bezanson, Jeff, Alan Edelman, Stefan Karpinski, and Viral B Shah. 2017. "Julia: A Fresh Approach to Numerical Computing." *SIAM Review* 59 (1). SIAM:65–98. https://doi.org/10.1137/141000671.

Braun, Julian, Maciej Buze, and Christoph Ortner. 2017. "The Effect of Crystal Symmetries on the Locality of Screw Dislocation Cores." *arXiv Preprint arXiv:1710.07708.*

Chen, Huajie, and Christoph Ortner. 2017. "QM/Mm Methods for Crystalline Defects. Part 2: Consistent Energy and Force-Mixing." *Multiscale Modeling & Simulation* 15 (1). SIAM:184–214. https://doi.org/10.1137/15M1041250.

Damle, A., A. Levitt, and L. Lin. 2018. "Variational formulation for Wannier functions with entangled band structure." *ArXiv E-Prints*, January.

Mogensen et al., (2018). Optim: A mathematical optimization package for Julia. *Journal of Open Source Software*, 3(24), 615. 2

joss.00615 editorial process

- Submission received
    - Submission form
- Referee report
- Revision
- Acceptance
- Publication

JOSEcon Fundamental principles

- Open source + open access
- Multilingualism, including proprietary languages
- Open reviewing process with objective publication criteria
- Novelty in implementation rather than method

Draft of the JOSEcon project charter

JOSEcon Publication criteria

Necessary conditions

- Code documentation
- Proper coding style
- Illustrative examples
- Open source license

Good to have

- Public repository (where updates will appear)
- Unit tests, continuous integration, etc.

Sign up at josecon.org

JOSEcon Advisory board

1. Christopher Carroll
2. John Stachurski
3. John Rust
4. Felix Kubler
5. Serguei Maliar

Sign up at josecon.org

Where we are

Years prior to 2018

The idea of publishing economics code discussed by Chris Carroll and others

2018

Discussions and collection of thoughts and opinions from various target groups

2019

Setting up and testing the infrastructure
Start production by the end of the year

Sign up at josecon.org

Thank you!