Introduction to scientific computing

OpenSourceEconomics[*]

March 18, 2020

We teach basic software engineering, numerical methods, and computational engineering skills. They allow you to leverage tools from computational science and increase the transparency and extensibility of our implementations. In doing so, you expand the set of possible economic questions that you can address and improve the quality of your answers.

We organize the course around the two flagship codes of our group. We use two codes to explore selected issues in software engineering, numerical methods, and computational engineering.

- `respy`

  We maintain a `Python` package for the simulation and estimation of a prototypical finite-horizon dynamic discrete choice model based on Keane and Wolpin (1997).

GitHub	`OpenSourceEconomics/respy`
Docs	`respy.readthedocs.org`

- `estimagic`

  We maintain a `Python` package that helps to build high-quality and user-friendly implementations of (structural) econometric models. `estimagic` provides a consistent interface to a large set of global and local optimizers. Complicated optimizations can be monitored using an interactive browser-based dashboard.

GitHub	`OpenSourceEconomics/estimagic`
Docs	`estimagic.readthedocs.org`

[*]Corresponding author: Philipp Eisenhauer, peisenha@uni-bonn.de.

Throughout the course, we will make heavy use of `Python` and its `SciPy` ecosystem and `Jupyter` Notebooks. Basic knowledge of this toolchain is a prerequisite. There exist numerous introductory resources, and we provide a curated list at `ose-resources.readthedocs.io`. We will use `zulip` for all course communications, please be sure to join our workspace at `ose.zulipchat.com`.

- **Introduction**

  We provide a general introduction to computational modeling in economics and our OpenSourceEconomics initiative. We outline opportunities for students to get involved.

  – OpenSourceEconomics (2020b). OpenSourceEconomics website

- **Testing**

  We discuss different types of automated tests and how they can be applied to scientific software projects. After presenting the basics, we show some examples in `estimagic` and `respy`.

  – Okken, B. (2017). *Python testing with pytest: Simple, rapid, effective, and scalable.* Pragmatic Bookshelf, Raleigh, NC

  – Arbuckle, D. (2010). *Python testing: Beginner's guide.* Packt Publishing, Birmingham, UK

- **Collaboration**

  We introduce continuous integration features of `GitHub` and discuss workflows for different team sizes.

  – Chacon, S. and Straub, B. (2014). *Pro git.* Apress, New York, NY

  – Blischak, J. D., Davenport, E. R., and Wilson, G. (2016). A quick introduction to version control with git and github. *PLoS Computational Biology*, 12(1)

- **Numerical optimization**

  After a brief theoretical introduction to local and global optimization, students will solve straightforward optimization problems with `scipy`. We then discuss the limitations of `scipy.optimize` in typical econometric workflows and introduce the students to the powerful optimization tools of the `estimagic` package.

  – Locatelli, M. and Schoen, F. (2013). *Global optimization.* SIAM, Philadelphia, PA

  – Nocedal, J. and Wright, S. J. (2006). *Numerical optimization.* Springer, New York, NY

  – Gabler, J. (2019). A Python tool for the estimation of (structural) econometric models

- **Numerical derivatives**

  After a brief introduction to numerical differentiation, we show how to calculate gradients, Jacobian and Hessian matrices using estimagic.

  – Brodtkorb, P. A. and D'Errico, J. (2019). numdifftools

  – Ridout, M. S. (2009). Statistical applications of the complex-step method of numerical differentiation. *The American Statistician*, 63(1):66–74

  – Gabler, J. (2019). A Python tool for the estimation of (structural) econometric models

- **Numerical integration**

  We first introduce the participants to quadrature, Monte-Carlo, and Quasi-Monte-Carlo methods and provide them with (over)simplified guidelines to choose between the different methods. We then show the convergence speed of different approaches in a discrete choice dynamic programming model.

  – Davis, P. J. and Rabinowitz, P. (2007). *Methods of numerical integration.* Dover Books on Mathmatics, Cambridge, MA

  – Heiss, F. and Winschel, V. (2008). Likelihood approximation by numerical integration on sparse grids. *Journal of Econometrics*, 144(1):62–80

  – Judd, K. L. and Skrainka, B. S. (2011). High performance quadrature rules: How numerical integration affects a popular model of product differentiation. *CEMMAP Working Paper*

- **Discrete choice dynamic programming models**

  We first outline the underlying economic, mathematical, and computational model. We then provide an overview of its applications in economics. Finally, we use `respy` to showcase the application of earlier lessons for numerical integration, parallelization strategies, version control, software testing, and continuous integration.

  – OpenSourceEconomics (2020a). Eckstein-Keane-Wolpin models: An invitation for transdisciplinary collaboration. *Handout*

  – Keane, M. P., Todd, P. E., and Wolpin, K. I. (2011). The structural estimation of behavioral models: Discrete choice dynamic programming methods and applications. In Ashenfelter, O. and Card, D., editors, *Handbook of Labor Economics*, volume 4A, pages 331–461. Elsevier Science, Amsterdam, Netherlands

  – Aguirregabiria, V. and Mira, P. (2010). Dynamic discrete choice structural models: A survey. *Journal of Econometrics*, 156(1):38–67

- **Execution speed**

  Execution speed `Python` is often called a slow language. Nevertheless, it is one of the most widely used languages for high-performance computing. We teach students how to make their `Python` code fast with `numpy` and `numba` and how to parallelize it using `multiprocessing` and `joblib`. After the students solve simple examples, we look at some implementations is `respy`.

  - Gorelick, M. and Ozsvald, I. (2014). *High performance Python.* O'Reilly Media, Sebastopol, CA
  - Lanaro, G. (2017). *Python high performance.* Packt Publishing, Birmingham, UK