High Performance Computing for Economists

Spring 2024 – Syllabus

1 Course Overview

The course introduces students to basic techniques required for high-performance computing in economics. The course is designed for students from all fields of economics and covers a broad range of topics including: basic tools of software engineering (e.g. the shell, version control, editors, IDEs, etc.), concepts and tools about programming languages and coding (e.g. functional programming, object-orientation, linters, debuggers, profilers, etc.), as well as theory and applications on parallel programming.

The course also provides a hands-on introduction to `Julia`, a modern and open-source programming language for high-performance computing, which will allow us to demonstrate how to use many of the tools presented during the course. Finally, we will also see how to access and use computational resources such as Princeton's Adroit clusters, which allows to tap into powerful multi-core processors.

While we will cover the basics of numerical optimization, we will mostly focus on the tools that you need to *implement* algorithms (e.g. how to write efficient, bug-free code that is both reusable and easily replicable). This class is specifically designed to ease your life when working with such tools.

The course is designed with a mixture of frontal and hands-on material. Students are therefore recommended to bring their own laptop to follow along. However, that is not strictly necessary.

2 Prerequisites

The course assumes no prior knowledge of any of the topics being covered. Some familiarity with a scripting language (e.g. `Matlab, R, Julia, Stata, Python` etc.) will prove useful. Ideally you would also want to know how to open a bash shell on your local machine.

3 Self-learning

Since the course covers a broad range of topics, there is not enough time to go in great depth in each topic. The material is therefore designed to give plenty of references that the student can then go over in his own time and when the necessity arises.

4 Outline

Here is an overview of the content we will cover in 3 days:

1. **Day 1**
   - Tools Part I
     - Basics and motivation
     - Introduction to OS and the shell
     - Editors and IDEs
     - Make
     - Version control with Git
     - Programming languages and paradigms
   - Basics of Numerical Optimization and Root-finding
     - Useful root-finding algorithms
     - Gradient-based approaches
     - Non-gradient-based approaches
     - Auto-differentiation
2. **Day 2**
   - Languages
     - Languages: C++, Python, Julia, R, Matlab, C, Fortran, Scala, Mathematica, Scala
     - Compilers
     - Libraries
   - Tools Part II
     - Linting
     - Debugging,
     - Profiling
     - Unit testing
   - Style, parallelization, and clusters
3. **Day 3**
   - Julia Tutorial and Parallelization
     - Basics
     - Variables and control structures
     - Functions and types
     - Advanced topics: type hierarchy and function composition
     - Advanced topics: parallel programming in `Julia`