Why High-performance Computing?

(Lectures on High-performance Computing for Economists I)

Jesús Fernández-Villaverde\textsuperscript{1} and Pablo Guerrón\textsuperscript{2}

November 21, 2021

\textsuperscript{1}University of Pennsylvania

\textsuperscript{2}Boston College
Computing has become a central tool in economics:

1. **Macro** → solution and estimation of dynamic equilibrium models, policy evaluation and forecast, ...

2. **Micro** → computation of games, labor/life-cycle models, models of industry dynamics, study of networks, bounded rationality and agent-based models, ...

3. **Econometrics** → machine learning, non-standard estimators, simulation-based estimators, large datasets, ...

4. **International/spatial economics** → models with heterogeneous firms and countries, dynamic models of international trade, spatial models, economic consequences of climate change and environmental policies, ...

5. **Finance** → asset pricing, non-arbitrage conditions, VaR, ...

6. **Economic history** → processing of large sets of non-standard information, library data, historical counterfactuals, ...
• Computation helps, complements, and extends economic and econometric theory. Judd (1997).

• Economics is not different from other fields (if anything, economics has been slow to embrace computation).

• Widespread movement across all scientific and engineering fields: *On Computing* by Paul S. Rosenbloom.

• Nowadays, computation in economics is also becoming key in:
  1. Policy making institutions.
  2. Regulatory agencies.
  3. Industry.
Past, present, and future

- Move towards computation started already in the 1950s (estimation of simultaneous equations models, simple static GE models, ...).

- But it gathered speed after the 1980s (RBC research program, first-generation simulation estimators, structural estimation, interest rate models, ...).

- Most likely, the computational trend will increase over time:
  1. Drop in computing costs.
  2. Big data.
  3. Machine learning and AI.
  4. New hardware.
  5. Change in composition of the profession.
Consequences for students

- This means that you will spend a substantial share of your professional career:
  1. Coding.
  2. Dealing with coauthors and research assistants that code.
  3. Reading and evaluating computational papers.
  4. Supervising/regulating people using computational methods.

- You want to lay solid foundations for future: concrete tools will change, fundamental ideas will not.
High-performance computing

- High-performance computing (HPC) deals with scientific problems that require substantial computational power.

- Even simple problems in economics generate HPC challenges:
  1. Dynamic programing with several state variables.
  2. Highly non-linear DSGE models with many shocks.
  3. Problems with occasionally binding constraints.
  4. Complex asset pricing.
  5. Structural estimation.
  6. Frontier estimators without close-forms formulae.
  7. Handling large datasets.
Parallel processing

• Usually, but not always, HPC involves the use of several processors:
  1. Multi-core/many-core CPUs (in a single machine or networked).
  2. Many-core coprocessors.
  3. GPUs (graphics processing units).
  4. TPUs (tensor processing units).
  5. FPGAs (field-programmable gate arrays).

• Most of these machines are available to all researchers at low prices.

• Nevertheless, we will also think about how to produce efficient serial code (although, following most recent developments, we will not emphasize much vectorization).
Parallel paradigms

FIGURE 1.2
Typical usage of APIs for parallel programs on specific computing devices/architectures (other configurations might be possible depending on particular compilers/software)

1. OpenMP
2. MPI
3. Intel Xeon Phi coprocessor and/or many-core CPUs
4. CUDA for writing parallel multithreaded programs running on NVIDIA GPUs.
5. OpenCL for writing parallel multithreaded programs running on multi-core CPUs and/or GPUs.
6. OpenACC for writing parallel multithreaded programs running on accelerators such as GPUs.
GPUs
Coprocessor
TPUs
FPGAs
Often HPC is framed regarding running time.

In practice, coding and debugging time is likely to be more relevant than running time.

We will spend considerable effort in discussing proper coding.

Savings in development time are often first-order. Savings in running time are most times second-order.

Adapted from Gen. Robert H. Barrow, USMC (27th Commandant of the US Marine Corps)

Amateurs talk about the speed of their processors, but professionals study coding techniques.
Some resources

- HPC carpentry: [https://hpc-carpentry.github.io/](https://hpc-carpentry.github.io/).
- Livermore documentation and tutorials: [https://hpc.llnl.gov/training/](https://hpc.llnl.gov/training/).
- HPC Wire: [https://www.hpcwire.com/](https://www.hpcwire.com/).
- Inside HPC: [https://insidehpc.com/](https://insidehpc.com/).
- *Parallel and High Performance Computing* by Robert Robey and Yuliana Zamora.
- *Introduction to High Performance Computing for Scientists and Engineers* by Georg Hager and Gerhard Wellein.
This set of notes does NOT cover:

1. Theory of computation and complexity theory.
2. Automata theory.
3. Computer architecture.
5. Numerical analysis.
6. Solution, estimation, and parallelization algorithms applied to economics.
7. \LaTeX and \BibTeX.