EC 410/510 – Computational Economics

David Evans

2026-11-13

About Me

About Me

Education

- PhD in Economics, New York University, 2015
- Associate Professor, Department of Economics, UO

Research Interests

- Optimal Government Policy
- Inequality
- Computational Economics
- Bounded Rationality

Contact

- Email: devans@uoregon.edu
- Office: PLC 538
- Office Hours: Th 10am–11am and Fridays 1pm–2pm and by appointment

About My Teaching Style

- I like to work with **a lot** of examples, and I will ask you to work through examples in class
- I'll switch between slides and code in class
  - All slides will be posted to Canvas
- We will be using **algebra and math** in this course
- I **really** enjoy teaching economics, so if I start going too fast
  - Raise your hand and ask me to slow down
  - Ask me to repeat material you didn't get
  - **"I have no idea what you're doing"** is a reasonable and *valuable* comment

Course Logistics

Meetings

Term	Spring 2026
Schedule	TTh 12:00–1:20pm
Room	TBD

Grading (Undergraduate)

Component	Weight
Problem Sets	40%
Midterm Exam	30%
Final Exam	30%

Grading (Graduate 510)

Component	Weight
Problem Sets	40%
Midterm Exam	25%
Final Exam	25%
Final Project	20%

Text

No formal text — many lectures draw on QuantEcon by Sargent & Stachurski. All lecture notes via Canvas.

Problem Sets

- Approximately every **two weeks**, covering the topic we've been studying
- **Do not expect to complete a problem set in one day** — especially if you are new to programming
  - Designed to take time and be completed via trial and error
- Graded on a check plus / check / check minus / zero scale
- **Late policy:** Problem sets can be turned in late, but you must contact me first with a concrete plan
  - Each problem set builds on prior material — falling behind is hard to recover from
  - Please communicate with me **early** if you are struggling

> **Tip**
>
> Working in groups is permitted, but all submitted work must be your own.

If You Feel Sick

> **Important**
>
> **I'm here for you** — contact me if you have issues or concerns. I've planned this course to be flexible.

- If you feel too sick to attend class or do the homework:
  - I'll post recordings of every class and notes generated in lecture
  - Rest up and send me an email to let me know
- You can always contact me if you need to turn in an assignment late

What Is Computational Economics?

Three Styles of Economics

	Empirical	Theoretical	Quantitative
Goal	Measure correlations in data	Prove existence/uniqueness results	Answer numerical questions with structural models
Strength	Grounded in data	General, rigorous	Quantitatively precise
Limitation	Hard to extrapolate to counterfactuals	Results often too general for quantitative significance	Requires calibration assumptions

⚠ **Warning**

Counterfactuals are not in the data. You cannot observe the economy under a policy that was never implemented — you need a model.

Qualitative vs. Quantitative Economics

Qualitative Analysis

- Deals with existence and uniqueness of equilibria, comparative statics, qualitative explanation of data patterns
- Often too stylized to be applicable to the real world
- Results too general to have quantitative significance: e.g. $\text{corr}(x, y) > 0$

Quantitative Economics

- Uses computational techniques to solve and simulate a structural economic model
- Allows for a more comprehensive characterization of equilibrium
  - *Can you match the sign **and magnitude** of the correlation?*
- Computational tools allow for solving far more complex and relevant cases
  - Explore all points in parameter space, not just analytically tractable ones

Features of Quantitative Macro

1. Questions are about **measurement** — answers are numerical
   - How much of Y can be explained by X?
   - By how much does policy P improve welfare?

2. Use a **structural theory** of the economy (a model) to derive quantitative implications

3. Parameters of the model are **calibrated** along some dimensions of the data and used to explain **other dimensions**

4. The **computer solves the equilibrium** and runs the experiment to answer the question

> ⚠ **Warning**
>
> Point 3 (calibration) is **controversial** — the rest are widely accepted. We will discuss this tension throughout the course.

Why Not Just Use Big Data / Machine Learning?

- ML is very good at prediction within the support of the data
- But macroeconomics often asks questions **outside** the data:
  - What happens if the Fed raises rates by 200 basis points?
  - What is the welfare cost of business cycles?
  - How does a new unemployment insurance policy affect job search?
- These **counterfactuals require a model** with disciplined assumptions about the structure of the economy

> **Tip**
>
> Structural models and empirical methods are complements, not substitutes.

Goals and Tools

Goals for This Course

By the end of this course you will:

1. Learn how to **model and simulate random variables** on a computer

2. Understand how to write dynamic optimization problems recursively as a **Bellman equation** and solve it numerically

3. Identify how to break down complicated economic models into **component parts** that can be solved on a computer

4. Understand how to use economic data to **calibrate parameters** of an economic model

> **Tip**
>
> The goal is not just to learn programming — it is to learn to think like a computational economist.

Course Topics

Week	Topic
1	Julia Basics and Linear Algebra
2	Finite State Markov Chains
3–4	McCall Search Model (Bellman Equations)
5	Rust (1987) Optimal Stopping + **Midterm**
6	Rust (1987) continued
7–8	Kydland & Prescott (1982): Real Business Cycles
9–10	Sovereign Default

Why Julia?

There are many languages to choose from: Matlab, Python, R, C++, Fortran…

Julia offers the best of all worlds:

- **Clean syntax** like Python — readable and expressive
- **Linear algebra first** like Matlab — natural for economic models
- **JIT compilation** — performance on par with C, C++, Fortran

```
# Example: compute a simple present value
β = 0.96
T = 50
pv = sum(β^t for t in 0:T)
```

> **Tip**
>
> Learning at least two general-purpose programming languages over your career pays large dividends. Julia is a great second language after Python, or a great first language for scientific computing.

Getting Set Up

What You Need to Install

Install these **in order** before the first problem set:

Software	Purpose
Julia	The language we use — download from julialang.org
VS Code	Code editor — download from code.visualstudio.com
Julia + Jupyter extensions	Run Julia inside VS Code
Git	Download the course notebooks

> **Tip**
>
> A detailed step-by-step guide is posted on Canvas (`GettingStarted.html`).

Installing Git

Mac — Git may already be installed. Open a terminal and run:

```
1 git --version
```

If not installed, macOS will prompt you to install the **Xcode Command Line Tools**.

Windows — download and run the installer from git-scm.com/download/win. Use the default settings.

Downloading the Course Notebooks

Open a terminal (Mac) or Git Bash (Windows) and run:

```
git clone https://github.com/dgevans/UndergraduateComputationalNotebooks
```

This creates a folder called `UndergraduateComputationalNotebooks` with all course notebooks and problem sets.

Setting Up the Julia Environment

The repository includes a `Project.toml` with all required packages. Install them once:

1. Open the `UndergraduateComputationalNotebooks` folder in VS Code
2. Click **Julia env:** in the bottom toolbar and select the course environment
3. Open the Julia REPL: `Alt+J` then `Alt+O`
4. Press `]` to enter package mode, then run:

```
instantiate
```

> **Tip**
>
> This only needs to run **once** and may take 5–10 minutes. After that, all packages load automatically when you open a notebook from this folder.

Closing Thoughts

- This course will be **hard**
  - But if you stick with it you will learn **a lot**
- Don't be afraid to stop me during lecture
  - "I have no idea what you're doing" is a reasonable and *valuable* comment
- Problem sets are due approximately every two weeks
  - Start them **early**, come to office hours with questions
- **Computers are changing economics** — the tools you learn here are used in academic research, central banks, and private sector consulting

> **Tip**
>
> See github.com/FRBNY-DSGE/DSGE.jl — the NY Fed runs its entire DSGE model in Julia.