AGEC 652 - Lecture 1.1

A course introduction

Diego S. Cardoso

Spring 2022

Welcome to AGEC 652!

- Formally: *Application of Quantitative Analysis: Mathematical Programming*

- Informally: **Computational methods for applied economics**

What you will learn in this course

1. How to setup and manage computational projects for your research

2. Understand why and when computational methods can help

3. And, most importantly, how to apply many techniques to solve economic problems using your computer

What we will cover

3 parts

1. An introduction to Scientific Computing for Applied Economics

2. Numerical methods to solve economic problems

3. Structural estimation of static models

    ○ We leave out dynamic models. Those you'll see in AGEC 654

What we will cover: Part 1

First 2 weeks

- Reproducibility

- Version control (git) and project organization

- Julia programming language

What we will cover: Part 2

Week 2 until about Spring Break

- Fundamentals of numerical methods
  - Computer architecture and arithmetic
  - Numerical differentiation and integration
  - Function approximation
- Systems of equations
  - Methods for linear and nonlinear systems
  - Economic models and applications
- Optimization
  - Methods for constrained and unconstrained
  - Economic models and applications

What we will cover: Part 3

From after Spring Break until second-to-last week

- Introduction to structural estimation
  - Structural vs. reduced-form
  - Data in Julia
  - Recap of MLE and GMM
- Estimation of models with a single agent
- Estimation of models with multiple (interacting) agents

Dates are tentative: This is a "new" course. Your feedback and understanding is appreciated.

Course materials

- Papers, articles, and tutorials
  - Links on *Brightspace > Content > Materials*
- Lecture slides, coding examples
  - Links on *Brightspace > Content > Materials*
  - Hosted on the GitHub class repository
- Books
  1. Miranda and Fackler (2002)
  2. Judd (1998)
  3. Nocedal and Wright (2006)
  - All available at Purdue Library, links on Brightspace

What you need to succeed in this course

- AGEC 552 or ECON 615: matrix algebra and differential calculus. And some basic game theory.

- Previous coding experience or willingness to spend some time learning as you go

- A laptop to use in class (come talk to me if you do not have one available)

What you have to do

- Come to class (unless unable)

- 6 individual problem sets

- Term paper (up to 3 people)

  - Proposal: March 11
  - Presentation: April 26 and 28
  - The actual paper: May 6

- A 15-minute presentation of a literature paper/package to the class

Grading

- Six problem sets: 60% (10% each)

- 15-minute literature presentation: 10%

- Term paper (proposal, presentation, final version): 30%

Problem sets (60%)

- Problem sets will be where you implement the techniques we learn
  - But we will be doing our fair share of coding in class
- You **must use Julia** and write .jl scripts (no Jupyter notebooks)
- Individual submissions on GitHub
  - This is intended for you to write code on your own and learn from the process
  - **But you can (and should) discuss solutions with classmates!**

Problem sets (60%)

- Why am I making you do problem sets this way?
- If you want to publish in AEA journals you need to have good practices
  - Other journals are doing the same
- Julia is very good for reproducibility

15-minute presentations (10%)

- Everyone will present a paper starting in March
- You can present (as long as it is related to the methods we see in this course)
  - An applied paper that uses computational methods
  - A methods paper
  - A tutorial of a package
- I will soon post a list of suggested papers and a survey of presentation dates
- You must consult with me at least 1 week prior to your scheduled presentation date to ensure the paper is appropriate for a presentation

Term paper (30%)

- You can work by yourself or in teams of up to 3

- Two options:

  - A paper that is the beginning of a **computationally-driven** research project
  - or an extension of an existing paper with new methods and analyses

- The only requirement is that the project **cannot be computationally trivial** (i.e., not something you could easily do with Stata commands)

Term paper (30%)

- A 2-page proposal is due right before Spring Break

- Everyone will present their final projects in the last week of class

- More details on the syllabus and to come later

Course and University policies

Attendance

- We follow standard Purdue regulations on attendance. But things are quite weird with COVID...

- If you must miss class for health or any other personal reason, please let me know as far in advance as possible by e-mail or Brightspace message

- Please respect the Protect Purdue Pledge

- And please **take care of yourself**

Course and University policies

Course evaluation

- There is the regular course evaluation by Purdue at the end of the semester

- In addition, I will ask for your anonymous and voluntary opinion halfway thorugh the course

  - This is a new course with inevitable rough edges
  - Your sincere feedback is appreciated and will help me deliver a better course for everyone

Please read the syllabus for other policies

Office hours

- Tuesdays from 2:30 to 4 PM via Zoom or by appointment
  - A link is available on Brightspace > Start Here

Let's stop here for a while and introduce ourselves!

Before we continue, some

Acknowledgments

These slides build on materials provided by Ivan Rudik (Cornell University) and on the textbooks and other sources referenced in the syllabus/reading list.

Why computational methods?

Why do we need computational methods?

Everything you've done so far has likely been solvable analytically

Including OLS: $\hat{\beta} = (X'X)^{-1}X'Y$

Not all economic models have closed-form solutions, and others can't have closed-form solutions with losing important economic content

What can we compute?

We can use computation + theory to answer **quantitative** questions

Theory can't give us welfare in dollar terms

Theory can't tell us the value of economic primitives

What can we compute?

Theory often relies on strong assumptions like:

- log utility (income loss vs substitution)
- no transaction costs (important friction)
- strictly concave objectives (natural phenomena don't follow this)
- static decision-making

It can be unclear what the cost of these assumptions are

Some apparently simple questions

Example 1

Suppose we have a constant elasticity demand function:

$$q(p) = p^{-0.2}$$

In equilibrium, quantity demanded is $q^* = 2$

Q: What price clears the market in equilibrium?

Example 1

A: Just invert the demand function

$$2 = p^{-0.2}$$

$$p^* = 2^{-5}\checkmark$$

Your calculator can do the rest

Example 2

Suppose the demand function is now

$$q(p) = 0.5p^{-0.2} + 0.5p^{-0.5},$$

a weighted average of two CE demand functions

In equilibrium, quantity demanded is $q^* = 2$

Q: What price clears the market in equilibrium?

First, does a solution exist?

Yes, why?

Example 2

$q(p)$ is continuous and monotonically decreasing

$q(p)$ is greater than 2 at $p = 0.1$ and less than 2 at $p = 0.2$

$\Rightarrow$ by intermediate value theorem $q(p) = 2$ somewhere in $(0.1, 0.2)$

Example 2

```
# We know solution is between .1 and .2
x = collect(range(.1, stop = .2, length = 10)) # generate evenly spaced grid
q_d = ones(size(x)).*2 # generate equal length vector of qd=2

# Define price function
price(p) = p.^(-0.2)/2 .+ p.^(-0.5)/2

# Get corresponding quantity values at these prices
y = price(x)
```

Now plot q_d and $q(p)$

Example 2
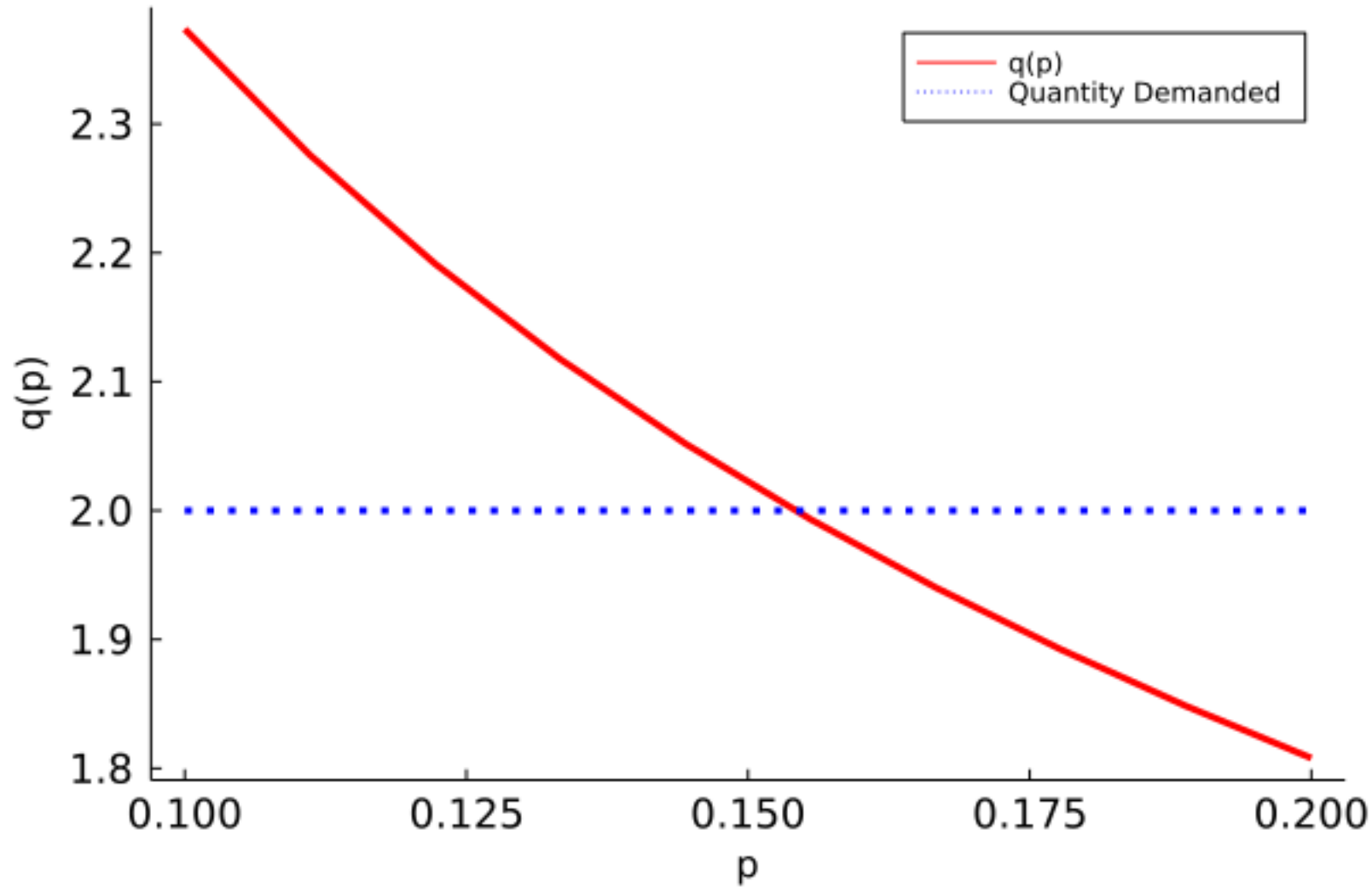
Example 2

Notice: if we let $t = p^{-0.1}$ then

$$q(t) = 0.5t^2 + 0.5t^5$$

Can we solve for t now?

No! Closed-form solutions to fifth order polynomials are not guaranteed to exist!

So how do we solve the problem?

Newton's method

Iteratively do the following:

1. Guess solution to: $q(p) - q^* = 0 \Rightarrow q(p) - 2 = 0$
2. Approximate the function with local 2nd-order polynomial around guess
3. Solve this easier equation
4. Solution is the new guess
5. Stop if previous guess and new guess are sufficiently close

We will learn more about this and why it works in a later class

Coding Newton's method

```
# Define demand functions
demand(p) = p^(-0.2)/2 + p^(-0.5)/2 - 2      # demanded quantity minus target
demand_grad(p) = .1*p^(-1.2) + .25*p^(-1.5) # demand gradient
```

```julia
function find_root_newton(demand, demand_grad)
    p = .3          # initial guess
    deltap = 1e10 # initialize stepsize

    while abs(deltap) > 1e-4
        deltap = demand(p)/demand_grad(p)
        p += deltap
        println("Intermediate guess of p = $(round(p,digits=3)).")
    end
    println("The solution is p = $(round(p,digits=3)).")
    return p
end;
```

Newton code

```
# Solve for price
find_root_newton(demand, demand_grad)
```

```
## Intermediate guess of p = 0.068.
## Intermediate guess of p = 0.115.
## Intermediate guess of p = 0.147.
## Intermediate guess of p = 0.154.
## Intermediate guess of p = 0.154.
## Intermediate guess of p = 0.154.
## The solution is p = 0.154.
```

```
## 0.15419764093200633
```

Example 3

Consider a two period ag commodity market model

- Period 1: Farmer makes acreage decisions for planting
- Period 2: Per-acre yield realizes, equilibrium crop price clears the market

The farmer's policy function is:

$$a(E[p]) = \frac{1}{2} + \frac{1}{2}E[p]$$

Example 3

Yield $\hat{y}$ is stochastic and only realized after planting period. It follows $\hat{y} \sim \mathcal{N}(1, 0.1)$

Crop production is

$$q = a\hat{y}$$

Demand is given by

$$p(q) = 3 - 2q$$

Q: How many acres get planted?

Example 3

Q: How many acres get planted?

A: We plug in the stochastic quantity and take expectations

$$p(\hat{y}) = 3 - 2a\hat{y}$$

$$a = \tfrac{1}{2} + \tfrac{1}{2}\left(3 - 2aE[\hat{y}]\right)$$

Rearrange and solve:

$$a^* = 1$$

Example 3

Now suppose the government implements a price floor such that $p > 1$.

We now have that

$$p(\hat{y}) = \max(1, 3 - 2a\hat{y})$$

Q: How many acres get planted with a price floor?

Example 3

Q: How many acres get planted with a price floor?

This is analytically intractable

The max operator is non-linear so we can't pass the expectation through

$$E[\max(1, 3 - 2a\hat{y})] \neq \max(1, E[3 - 2a\hat{y}])$$

We need to solve this numerically

Function iteration

We can solve this using a technique called **function iteration**

```julia
# Function iteration method to find a root
function find_root_fi(mn, variance)

    y = randn(1000)*sqrt(variance) .+ mn # draws of the random variable
    a = 1.                                # initial guess
    differ = 100.                         # initialize error
    exp_price = 1.                        # initialize expected price

    while differ > 1e-4
        a_old = a                         # save old acreage
        p = max.(1, 3 .- 2 .*a.*y)        # compute price at all distribution points
        exp_price = mean(p)               # compute expected price
        a = 1/2 + 1/2*exp_price           # get new acreage planted given new price
        differ= abs(a - a_old)            # change in acreage planted
        println("Intermediate acreage guess: $(round(a,digits=3))")
    end

    return a, exp_price
end
```

Function iteration

```
acreage, expected_price = find_root_fi(1, 0.1)
```

```
## Intermediate acreage guess: 1.122
## Intermediate acreage guess: 1.086
## Intermediate acreage guess: 1.095
## Intermediate acreage guess: 1.093
## Intermediate acreage guess: 1.093
## Intermediate acreage guess: 1.093
## Intermediate acreage guess: 1.093
```

```
## (1.0932445776485156, 1.1864891552970311)
```

```
println("The optimal number of acres to plant is $(round(acreage, digits = 3)).\nThe expected price
```

```
## The optimal number of acres to plant is 1.093.
## The expected price is 1.186.
```

Don't leave just yet!

Things to do before next class

1. Read Gentzkow and Shapiro (2014) *Code and Data for the Social Sciences*

2. Watch the 5-minute video *What is version control?*

   ○ Links for both are on Brightspace > Content > Materials

3. Create a GitHub account with your Purdue email (or add it to your existing account)

4. Install GitHub Desktop