Topics in Macroeconomics

Lecture - Introduction

Diego de Sousa Rodrigues
de_sousa_rodrigues.diego@uqam.ca

ESG-UQAM

Fall 2025

Course Information

- **E-mail:** de_sousa_rodrigues.diego@uqam.ca.
- **This course:** Provides techniques for the analysis and evaluation of dynamic economic models, e.g. DSGE, firm and industry models.
- **Target group:** Students who intend to do quantitative research in macro, applied micro, or structural econometrics.
- **Background:** Dynamic programming / Recursive macro.
- **Main programming languages:** `MATLAB` and `Julia`.

Road Map

1. Why Study Numerical Methods?
2. Issues in numerical methods and programming
3. Useful course information

Why Study Numerical Methods?

- **Paper-and-pencil** has its limits.
- Closed-form solutions are often hard or impossible.
- Increasing computational power allows us to study complex models.
- Techniques are useful in other fields (applied micro, trade, labour) — structural estimation.
- **Drawbacks:** Only approximate solutions, no theorems/proofs, room for human error, ...

A Computational Experiment

- A researcher poses a quantitative question, uses theory to construct a model economy, and solves it on the computer to answer the question.
- Computational experiments are used in other disciplines (physics, engineering, etc.).

Five steps of a computational experiment:

A Computational Experiment

- A researcher poses a quantitative question, uses theory to construct a model economy, and solves it on the computer to answer the question.
- Computational experiments are used in other disciplines (physics, engineering, etc.).

Five steps of a computational experiment:

1. **Pose a question**.
2. **Use a theory and construct a model**.
3. **Calibrate (estimate) and solve numerically the economy**.
4. **Run the experiment**.

1. Pose a Question

- The first (and perhaps most important and difficult) step is to come up with a **well-defined question**.
- A well-posed question should not be vague or too general and should have **quantitative implications (e.g., for policy evaluation)**.

1. Pose a Question

- The first (and perhaps most important and difficult) step is to come up with a **well-defined question**.
- A well-posed question should not be vague or too general and should have **quantitative implications (e.g., for policy evaluation)**.

Examples:

- *What are the welfare consequences of a tax reform?* (Gravelle and Kotlikoff, 1995).
- *Is growth good for the poor?* (Dollar and Kraay, 2002).
- *What is the quantitative nature of fluctuations induced by technology shocks?* (Kydland and Prescott, 1982).
- *Do barriers to technology adoption account for income disparities across countries?* (Parente and Prescott, 1999).

2. Use a theory and construct a model

- *"An explicit set of instructions for building a mechanical imitation system"* (Lucas, 1980). It should have **internal and external consistency**.
- Example: the neoclassical growth model.
- Of course,there are plenty of issues that the growth model fails to answer, but recall that as long as it serves well in analyzing the question posed this is satisfactory.
- Old, well-tested theories were once new and untested — sometimes it's worth exploring new theories.

3. Calibrate (estimate) and solve numerically the economy

- Use data from a true economy (e.g., Canada) to **tie down parameters so the model mimics the true economy** — at least in dimensions important for your question.
- Solve the model numerically with appropriate techniques (we will cover this in the course).

4. Run the Experiment

- With the replicated laboratory, you can run **counterfactual experiments**.
- Keep in mind limitations of computers:
  - Model uncertainty vs. computer pseudorandomness.
  - Discrete finite machine: approximations (grids, etc.).
  - Computational constraints: improve algorithms/computing power/language.

Programming and Computing: Jargon

- **Algorithm:** sequence of steps for doing a calculation.
- **Programming language:** translates algorithms into executable code (e.g., MATLAB, Fortran, Julia, Gauss).
- **Routine/program/code:** algorithm translated into a language (MATLAB: `.m` files).
- **Built-in functions:** provided by the language (e.g., `mean` in MATLAB).
- **Iteration/recursion/loop:** repeated steps (e.g., `for`, `while`).

Algorithms: Example

Basic structure:

```
define parameters/variables
statement 1
statement 2
...
end
```

Example: compute $4!$:

```
x = 1
x = x * 2
x = x * 3
x = x * 4
```

What about 200!?

Calculating 200! with a Loop

Pseudocode:

```
x = 1
for n = 2 to 200
  x = x * n
end
```

MATLAB:

```
x = 1;
for n = 2:200
  x = x * n;
end
```

Computing Issues

- **Accuracy:** computers are finite; cannot capture the continuum of real numbers.
- **Efficiency:** for heavy algorithms, coding "tricks" can be crucial.
- **Direct vs iterative methods:** trade off precision and speed.
- **Dealing with infinity:** approximate limits with stopping rules.

Pseudorandom Number Generators

- Computers generate **pseudorandom** (deterministic-looking) sequences.
- MATLAB examples:

```
% Uniform[0,1], 5 elements (column)
x = rand(5,1);

% Normal(0,1), 5 elements
y = randn(5,1);

% Normal(10, 4), 5 elements
r = 10 + 2*randn(5,1);

% Fixed seed / repeatability
s = rng;               % current settings
x = rand(1,5);         % some values
rng(s);                % restore settings
y = rand(1,5);         % same values: x == y
```

Simulation: Coin Toss Example

Goal: repeat 100 tosses, then repeat the experiment 1000 times.

Simulation: Coin Toss Example

Goal: repeat 100 tosses, then repeat the experiment 1000 times.

```
repeat 1000 times
for i = 1 up to 100 r = RAND
 if r < 0.5
    record H
else
    record T
end
stop when i becomes 100
      stop after repeating 1000 times
```

Simulation

- In the same way, **after computing the equilibrium of a model economy, you could in principle produce manually realisations of the economy**.
- However,you might be interested in calculating statistics for the economy that would involve simulating it many thousands of times.

Stopping Rules

- For an iterative sequence $x_n \to x^*$, stop when

$$\|x_n - x_{n+1}\| < \varepsilon.$$

- MATLAB's floating-point epsilon: $\texttt{eps} = 2^{-52}$.
- Often better to use relative tolerance:

$$\frac{\|x_n - x_{n+1}\|}{\|x_n\|} < \varepsilon, \qquad \frac{\|x_n - x_{n+1}\|}{\|x_n\| + 1} < \varepsilon.$$

Programming Style

Bad:

```
n=100
theta=0.8
u=randn(n,1);
y=zeros(n,1);
for t=2:n
y(t)=theta*y(t-1)+u(t);
end
```

Programming Style

Bad:

```
n=100
theta=0.8
u=randn(n,1);
y=zeros(n,1);
for t=2:n
y(t)=theta*y(t-1)+u(t);
end
```

Better (comments, spacing, indentation):

```
% AR(1) with N(0,1) errors
n = 100;               % sample size
theta = 0.8;         % persistence
u = randn(n,1);      % shocks
y = zeros(n,1);      % init
for t = 2:n
  y(t) = theta*y(t-1) + u(t);
```

Debugging

- **Bugs:** syntax vs logic errors.

Debugging

- **Bugs:** syntax vs logic errors.
- Syntax bugs are often caught by the interpreter; logic bugs require careful checking.
- Tips:
  - Test special cases with known answers.
  - Use intuition about expected results.
  - Step through code; modularize into small testable parts.

Our Running Example for a Large Fraction of the course: Standard RBC Model

- **Representative agent** with preferences:

$$\mathbb{E}_{t=0}\left[\sum_{t=0}^{\infty}\beta^t u(c_t)\right].$$

- **Representative firm** with CRS technology:

$$Y_t = z_t F(K_t, L_t).$$

- **Capital accumulation**:

$$K_{t+1} = (1-\delta)K_t + I_t.$$

- **Resource constraint**:

$$c_t + I_t = F(K_t, L_t).$$

- No government; competitive equilibrium; work with planner's problem (2nd welfare theorem).

Log-Utility, Cobb–Douglas, Full Depreciation

If $u(c) = \log c$, $F(K, L) = K^{\alpha} L^{1-\alpha}$, and $\delta = 1$, we know the closed-form solution — useful for comparing numerical methods. With $L_t = 1$, write $F(K, 1) = f(K)$.

Planner's problem:

$$V(K, z) = \max_{c, K'} \left\{ u(c) + \beta \mathbb{E}[V(K', z')] \right\}$$
$$\text{s.t. } c + K' = zf(K) + (1 - \delta)K.$$

Consumer solves

$$\max_{\{c_t, k_{t+1}\}_{t=0}^{\infty}} \quad \sum_{t=0}^{\infty} \beta^t u(c_t), \quad 0 < \beta < 1$$

$$\text{s.t.} \quad k_{t+1} + c_t \le f(k_t) + (1-\delta)k_t, \quad k_0 > 0,\ c_t \ge 0.$$

Assume u and f strictly concave; f satisfies INADA.

Concave Programming & FOCs

For finite horizon T, interior solution (INADA) satisfies Kuhn–Tucker/FOCs:

$$u'(c_t) = \beta\big(f'(k_{t+1}) + (1-\delta)\big)u'(c_{t+1}), \quad t = 0, 1, \ldots, T.$$

$$u'\big(f(k_t) + (1-\delta)k_t - k_{t+1}\big) = \beta\big(f'(k_{t+1}) + (1-\delta)\big)u'\big(f(k_{t+1}) + (1-\delta)k_{t+1} - k_{t+2}\big).$$

Transversality (finite horizon):

$$\beta^T\, u'\big(f(k_T) + (1-\delta)k_T - k_{T+1}\big)\, k_{T+1} = 0.$$

Assume $u(c) = \ln c$, $\delta = 1$, $f(k) = k^\alpha$.

$$\frac{1}{c_t} = \beta\alpha k_{t+1}^{\alpha-1}\frac{1}{c_{t+1}}, \quad c_t = k_t^\alpha - k_{t+1},$$
$$k_{t+1}^\alpha - k_{t+2} = \beta\alpha k_{t+1}^{\alpha-1}\big(k_t^\alpha - k_{t+1}\big).$$

Let $z_t \equiv \dfrac{k_{t+1}}{k_t^\alpha}$. Then $[1 - z_{t+1}] = \beta\alpha\left(\dfrac{1}{z_t} - 1\right)$.

Pencil-and-Paper Example

Recursive solution (finite T):

$$
\begin{aligned}
z_T &= 0, \\
z_{T-1} &= \frac{\alpha\beta}{1+\alpha\beta}, \\
z_{T-2} &= \frac{\alpha\beta(1+\alpha\beta)}{1+\alpha\beta+(\alpha\beta)^2}, \\
z_t &= \frac{\alpha\beta\big(1+\alpha\beta+\cdots+(\alpha\beta)^{T-t-1}\big)}{1+\alpha\beta+\cdots+(\alpha\beta)^{T-t}}.
\end{aligned}
$$

Pencil-and-Paper Example III

Closed form:

$$z_t = \alpha\beta \, \frac{1-(\alpha\beta)^{T-t}}{1-(\alpha\beta)^{T-t+1}}, \qquad k_{t+1} = z_t \, k_t^{\alpha}.$$

As $T \to \infty$, the limit coincides with the infinite-horizon solution: $k_{t+1} = \alpha\beta \, k_t^{\alpha}$.

Different Methods & Helpful Techniques

- Multiple solution methods to solve the problem above and a lot more; suitability depends on problem structure.
- Helpful techniques: **root-finding, interpolation, discretization of stochastic processes, etc**.
- Extensions: **stationary distributions, idiosyncratic/aggregate risk, OLG, calibration, simulation-based estimation**.

Some Helpful Material

References:

- Adda and Cooper, *Dynamic Economics*.
- Judd, *Numerical Methods in Economics*.
- Marimon and Scott, *Computational Methods for the Study of Dynamic Economies*.
- Miao, *Economic Dynamics in Discrete Time*.
- Sauer, *Numerical Analysis*.

Online resources:

- Wouter den Haan: `http://www.wouterdenhaan.com`
- Karen Kopecky: `http://www.karenkopecky.net`
- Makoto Nakajima: `https://sites.google.com/site/makotonakajima/`