Quantitative Macroeconomics and Numerical Methods

Maciej Sztachera
Goethe University Frankfurt

Tutorial A
Introduction

Plan for the tutorial

- Learn techniques to solve discrete-time dynamic optimization problems
- Implement these techniques on the computer
- Apply them to canonical macro models:
  - Neoclassical growth model
  - Canonical incomplete markets model (Aiyagari)
  - Overlapping Generations (OLG) models
  - Search models

Assignments

- Five assignments where you solve models numerically
- You can work on assignments in groups of two (three allowed as an exception)
- You can freely choose a programming language
- Coleman, Lyon, Maliar, and Maliar (2021) compare programming languages for economists
  - Matlab - great built-in functions and documentation, but closed-source and slow in some applications
  - Python - general programming language with a mature package system that is open source, but slow without additional effort
  - Julia - built-in JIT-compilation, code easy to parallelize, but JIT-compilation incurs a fixed cost at every run, less mature than Python
- Office hours 16:00-17:00 on Mondays in room RuW 4.255

Today: Solve a dynamic consumption-savings problem

- Dynamic model

$$v(a) = \max_{c,a'} \left\{ u(c) + \beta v(a') \right\}$$

$$\text{s.t. } c = (1+r)a + y - a'$$

where

- c: consumption (control)
- a': assets tomorrow (control)
- a: assets today (endogenous state)
- $v(a)$: value function

- Does this recursive problem have a unique solution for the value function? Why?
- What methods can we use to solve this problem?

Let $X \subseteq \mathcal{R}^l$ and let $C(X)$ be a space of bounded functions $f\colon X \to \mathcal{R}$, with the supremum metric. Let $T\colon C(X) \to C(X)$ be an operator satisfying two conditions.

1. **Monotonicity** If f, $g \in C(X)$ and $f(x) \leq g(x)$ for all x, then $T[f](x) \leq T[g](x)$ for all $x \in X$

2. **Discounting** There exists $\delta \in (0,1)$ such that for any constant c

$$T[f+c](x) \leq T[f](x) + \delta c \quad \forall f \in C(X), x \in X$$

If these two conditions are satisfied, T is a contraction mapping.

The Bellman operator $T[v] = \max_{c,a'} \log c + \beta v$ satisfies Blackwell's conditions.

1. **Monotonicity**

$$v \geq u \implies T[v] = \log c_v^* + \beta v(a_v^*) \geq \log c_u^* + \beta v(a_u^*) \geq \log c_u^* + \beta u(a_u^*) = T[u]$$

2. **Discounting**

$$\begin{aligned} T[v+c](a) &= \max_{c,a'} \log c + \beta \left(v(a') + c \right) \\ &= \left[\max_{c,a'} \log c + \beta v(a') \right] + \beta c \\ &\leq T[v](a) + \underbrace{\frac{1+\beta}{2}}_{\delta<1} c \end{aligned}$$

Therefore, by the Contraction Mapping Theorem, the Bellman operator has a unique fixed point.
We can recover it using value function iteration.

Value Function Iteration: Iterate on the Bellman equation

Since the Bellman operator T is a contraction mapping, one can apply it iteratively to recover its fixed point.

$$
\begin{aligned}
T[v](a) &= \max_{a'} \log\left((1+r)a + y - a'\right) + \beta v(a') \\
T[T[v]](a) &= \max_{a'} \log\left((1+r)a + y - a'\right) + \beta T[v](a') \\
T[T^2[v]](a) &= \max_{a'} \log\left((1+r)a + y - a'\right) + \beta T^2[v](a') \\
\vdots \qquad &= \qquad \vdots \\
T[T^n[v]](a) &= \max_{a'} \log\left((1+r)a + y - a'\right) + \beta T^n[v](a')
\end{aligned}
$$

We know that $\lim_{n\to\infty} T^n[v] = v^*$. It means that iterating on the value function sufficiently many times should bring us arbitrarily close to the fixed point v^*.

Value Function Iteration with Full Discretization: Pseudocode

1. Choose a set of grid points for assets $\mathcal{A} = \{a_{min}, ..., a_{max}\}$
2. Make an initial guess for the value function, e.g., $v^0(a) = 0$ for all $a \in \mathcal{A}$
3. Iterate on the Bellman equation
4. At each iteration step k, do the following:

   I Compute $\phi_{ij}^k = \log\left((1+r)a_i + y - a_j\right) + \beta v^{k-1}(a_j)$ for assets today $a_i \in \mathcal{A}$ and assets tomorrow $a_j \in \mathcal{A}$

   II Find the index j^* that maximizes the value function today

$$\phi_{ij^*} = \max_{j \in 1,\ldots,n_a} \log\left((1+r)a_i + y - a_j\right) + \beta v^{k-1}(a_j)$$

   III Optimal index j^* gives the location of optimal savings (the policy function) at the grid $\mathcal{A}$, and $v^k(a_i)$ is the new guess for the value function

$$v^k(a_i) = \phi_{ij^*}$$

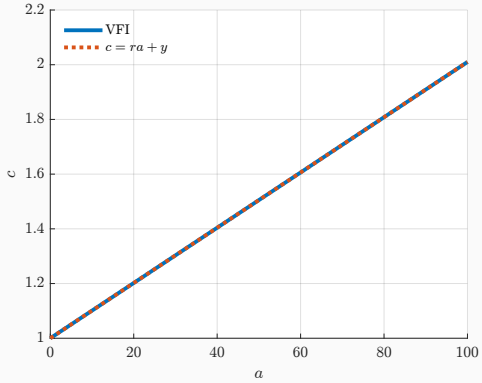
   IV Check if the new value function is sufficiently close to the old one.

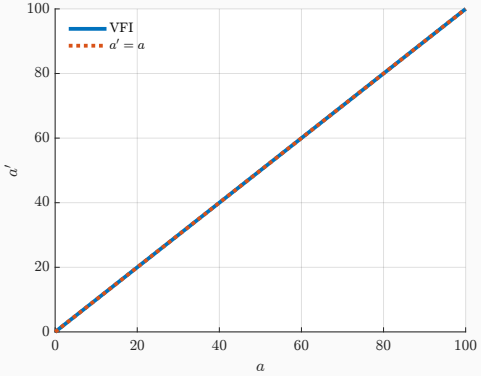
$$\|v^k - v^{k-1}\| \leq \left(1 + \|v^k\|\right) \cdot \varepsilon$$

Useful commands

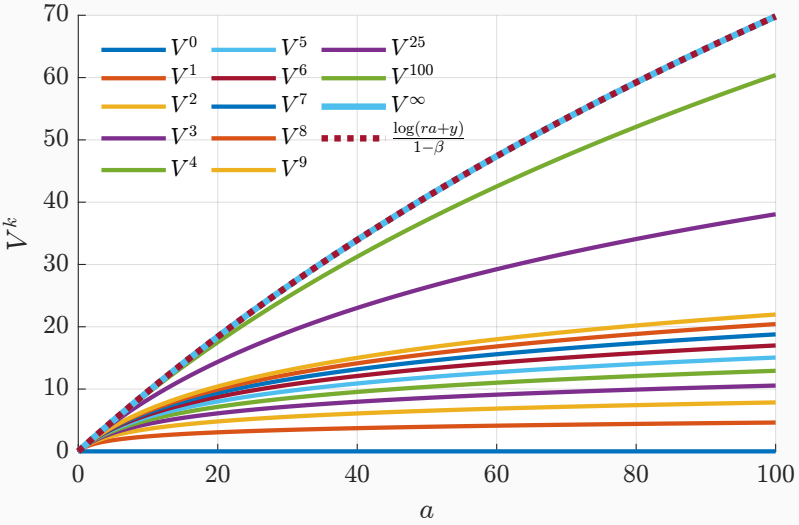
1. Julia
   - `zeros(na)` for an initial value function guess
   - `argmax(u+beta*v,dims=2)` to find optimal assets tomorrow
2. Matlab
   - `zeros(na)` for an initial value function guess
   - `[vnew,idx] = max(u+beta*v,[],2)` to find optimal assets tomorrow
3. Python
   - `np.zeros(na,1)` for an initial value function guess
   - `np.argmax(u+beta*v,axis=1)` to find optimal assets tomorrow

Value Function Iteration with Full Discretization: Policy functions





Value Function Iteration with Full Discretization: Value function convergence



References

Coleman, C., S. Lyon, L. Maliar, and S. Maliar (2021): "Matlab, Python, Julia: What to Choose in Economics?" *Computational Economics*, 58, 1263–1288.