Introduction and Overview

Summer School in Economics and Finance 2023

César Salinas

Indiana University

Spring 2023

Introduction

- Instructor: César Salinas
- Email: `salinasdcs@gmail.com`
- Lectures: from January 3 to January 6, 2023
- Lecture time: 7pm-10pm
- Zoom link, syllabus, lecture notes and code for tutorials:

  Posted at Canvas.

Introduction

- I assume prior background in dynamic programming and Python
- Additional resources to learn more about Python:
  - https://python-programming.quantecon.org/intro.html
  - http://scipy-lectures.org/intro/language/python_language.html
  - https://jakevdp.github.io/PythonDataScienceHandbook/

Why Python?

Table 1: Average and Relative Run Time (Seconds)

	Mac		
Language	Version/Compiler	Time	Rel. Time
C++	GCC-7.3.0	1.60	1.00
	Intel C++ 18.0.2	1.67	1.04
	Clang 5.1	1.64	1.03
Fortran	GCC-7.3.0	1.61	1.01
	Intel Fortran 18.0.2	1.74	1.09
Java	9.04	3.20	2.00
Julia	0.7.0	2.35	1.47
	0.7.0, fast	2.14	1.34
Matlab	2018a	4.80	3.00
Python	CPython 2.7.14	145.27	90.79
	CPython 3.6.4	166.75	104.22
R	3.4.3	57.06	35.66
Mathematica	11.3.0, base	1634.94	1021.84

Source: From Aruoba and Fernández-Villaverde (2018).

Why Python?

- Python is free and open source.
- Python is a general-purpose language used and supported extensively by companies and government agencies.
- Python is particularly popular within the scientific community.
- Python is also very beginner-friendly and is found to be suitable for students learning programming and recommended to introduce computational methods.
- More about Python: `https://python-programming.quantecon.org/about_py.html`

A Very Typical Problem

A classic problem we will be focusing throughout the semester:

$$
\begin{aligned}
V(z,k) &= \max_{c,k'} u(c) + \beta \mathbb{E} V(z',k') \\
s.t. \quad & c + k' = F(z,k) + (1-\delta)k
\end{aligned}
$$

Many questions here:

1. Does a solution exist?
2. Is there a unique solution or not?
3. What are the relevant z, k?
4. How can V be represented on a computer?
5. How can we compute $\mathbb{E}_z$?
6. How can we find the optimal c and k'?

The Bellman equation is a functional equation because it can be written as $V = T(V)$ or

$$V = TV.$$

Two useful theorems. 2 crucially theorems in Chapter 3 of Stokey and Lucas: (i) the Blackwell's Theorem (BT), and (ii) the Contraction Mapping Theorem (CMT).

- To understand the BT you need to be familiarized with metric spaces, norms and contraction mappings.

- The BT gives sufficient conditions for an operator T to be a contraction mapping (it lets us to invoke the CMT).

- The latter theorem establishes that if an operator T is a contraction mapping, then
  1. it has a unique fixed point, i.e. there exists a unique function V such that $TV = V$.
  2. $T^n V_0 \to V$ as $n \to \infty$.

- The beauty of CMT is powerful. It not only tell us the existence and uniqueness of V^* but it also shows us how to find it!
- CMT suggests:
  1. Set $n = 0$. Choose an initial guess $V_0 \in S$
  2. Find V_{n+1} by solving
  $$V_{n+1} = TV_n = \max_{c,k'} u(c) + \beta \mathbb{E} V(z', k')$$
  3. Set $n = n + 1$ and repeat step 2 until $|V_{n+1} - V_n| < \varepsilon$
- We still need to address some other questions

Discretization. The easiest, flexible and most robust way to solve this and similar problems.

$$
\begin{aligned}
V(z,k) = \max_{c,k'} u(c) + \beta \sum_{z' \in \mathcal{Z}} F(z'|z) V(z',k') \\
s.t. \quad c + k' = F(z,k) + (1-\delta)k, \\
\text{for all } k \in \mathcal{K}, z \in \mathcal{Z}.
\end{aligned}
$$

With all this in mind the answers to the previous questions are

1. Does a solution exist? Yes
2. Is there a unique solution or not? Yes
3. What are the relevant z, k? $\mathcal{K} \times \mathcal{Z}$
4. How can V be represented on a computer? Matrix
5. How can we compute $\mathbb{E}_z$? Markov Chain
6. How can we find the optimal c and k'? Checking all z, k

A Simple Example

Let's consider the Neoclassical Growth model with: log utility, Cobb-Douglas production, full depreciation and non-stochastic

$$V(k) = \max_{c,k'} \left\{ \log(c) + \beta V(k') \right\}$$
$$s.t. \quad c + k' = Ak^{\alpha}$$

or more compactly

$$V_{n+1}(k) = TV_n = \max_{k'} \left\{ \log(Ak^{\alpha} - k) + \beta V_n(k') \right\}$$

We want to solve for the value function $V(k)$ and the policy function $k' = g(k)$

Value Function Iteration

1. Make a guess on $V_0(k) = 0$ for all k, which gives us

$$V_1 = \max_{k'} \left\{ \log(Ak^\alpha - k') \right\} \Rightarrow k' = 0 \Rightarrow V_1(k) = \log A + \alpha \log k$$

2. V_2 becomes

$$V_2 = \max_{k'} \left\{ \log(Ak^\alpha - k') + \beta(\log A + \alpha \log k') \right\},$$

   and the FOC implies $k' = \frac{\alpha\beta A k^\alpha}{1+\alpha\beta}$

3. Substitute k' to obtain V_2 and keep iterating until convergence. We can verify that in the limit $T \to \infty$

$$k' = \alpha\beta k^\alpha$$

Projection Methods

- From the previous slide, we can notice that the value function takes the form of $V(k) = a + b \log k$ where a and b are unkown coefficients.
  Then, all we need to solve for is

$$a + b \log k = \max_{k'} \left\{ \log(Ak^{\alpha} - k') + \beta(\log A + \alpha \log k') \right\}$$

- This can be solved with pencil and paper and we should get

$$a = \frac{\log A + (1 - \alpha\beta) \log(1 - \alpha\beta) + \alpha\beta \log \alpha\beta}{(1 - \beta)(1 - \alpha\beta)}$$
$$b = \frac{\alpha}{1 - \alpha\beta}$$

Projection Methods

- This method can also be applied to solve for the policy function directly.
  When FOCs are sufficient, then we can find the solution to the above problem using the FOC:
  $$\frac{1}{Ak^{\alpha} - g(k)} - \frac{\alpha\beta A(g(k))^{\alpha-1}}{A((g(k))^{\alpha} - g(g(k)))} = 0,$$
  where we want to soplve for the functional equation $g(k)$
- Guessing $g(k) = sAk^{\alpha}$ and substituting in the equation above we should be able to find
  $$s = \alpha\beta$$
- The functional form will not be as simple and trivial as this one, but we will cover methods to approximate functions with pretty flexible basis functions later

Content

1. Discretizing the State and Choice Space
   1.1 Tauchen
   1.2 Rouwenhorst
   1.3 Brute Force Grid Search
2. Function Approximation
   2.1 Endogenous Grid Method
3. Root Finding
   3.1 Value Function Iteration with first order conditions
4. Optimization Methods
   4.1 Value Function Iteration with optimization methods