Advanced Macroeconomic Analysis

Release 0.9a

Timothy Kam

Apr 29, 2019

CONTENTS

Lecture Notes

Other formats (limited functionality)

- `PDF`
- `Epub`

CHAPTER

ONE

ABOUT THIS COURSE

This is a course for the student intending to continue on to a serious Master's degree involving economics in general, and macroeconomics in particular. The target student is one who will be a deep thinker and a thought leader who will be capable of understanding modelling techniques. This is not a course for the student intending to get another diploma for the sole purpose of career promotion, nor for the consumer of superficial or imprecise economic "intuitions".

We will be introduced to some core macroeconomic models that help to address some of the major empirical and policy questions in macroeconomics. These question relate to business cycles, long term growth, short-run fiscal and monetary policies, long-run or intergenerational fiscal policies, and to institutions such as fiat money and financial intermediation that facilitate trade.

1.1 What will we study?

Business Cycles and Monetary Policies: Two Flavors. We will begin our study of some common stories behind observed data on economic fluctuations. Technically, the simplest approach would be to model economic behavior/outcomes as they get revealed in observational data over time or over the business cycle. For example, a most naive approach would be to write down a statistical correlation model that fits the observed data: This is the hallmark of most textbook models of Keynesianism. Keynes' was an approach that attempted to divorce Macroeconomics from Classical Economics (i.e., microeconomics or price theory).

Note: There are later generations of this approach with slightly more microeconomic foundations, known as the New Neoclassical Synthesis or New Keynesianism. We won't have the time or the technical apparatus, yet, to look into this more recent literature. This will be taken up in a subsequent Macroeconomic Theory course.

Once we're done with the Keynesian way of thinking, we will revisit some similar business cycle questions again using a model that has more detail in terms of how agents' behavior arise in the aggregate. The simplest version we'll use is an overlapping generations (OLG) model, where there is an explicit mapping from agents' underlying choices, through their choice interactions in markets, to the resulting aggregate behavior of the economy. A crucial and contrasting lesson here is as follows: The aggregate rules (functions) governing behavior in the Keynesian model is assumed unchanged in the presence of policy change, whereas in the more microfounded model (monetary) policy change can confound what a Keynesian would have predicted to be stable aggregate behavior. We shall see that the linchpin in the conflict of these models is how one goes about modelling the crucial relations that may affect a policy maker's trade off: In this instance, it is the so-called Phillips curve. Should one assume that it is a stable statistical relation, or, should one suspect that that relation may be transient and may depend on something deeper and reactive to policy changes?

Warning: In practice, the policy consultant has a tendency to take the short cut of making policy conclusions via statistical relations. The lessons here will present a grave warning to such temptations of instant gratification in policy modelling and practice.

Fiscal Policies: Two Flavors. We will also study various forms of fiscal policy in the short run, initially in the Keynesian setting, and later, in the OLG setting. In the latter case, the model provides an explicit and natural environment to rationalize government intervention because the model has a natural sense of agent heterogeneity and market imperfection.

Facilitating Trade in Missing Markets: Dynamic Redistribution and Other Institutions. We'll also learn about how various forms of institutions: money, financial intermediation or banks, social security and fiscal policies can play a role in making economic outcomes "better" for the agents. This is studied through the lens of the OLG model economy.

1.2 How do we do it?

In this course, we'll often begin our stories with pictorial or geometric representations of models. We will then get you ready for more advanced study using more rigorous mathematical representations, analysis and arguments. The purpose of doing so is to ensure that our logic is well-disciplined, so we don't end up making rationalizations and policy conclusions outside of the premise of our underlying model. Another purpose of this is to prepare you for the serious careers that will require modelling, analytical and computational skills—i.e., insuring you to be more future-proof.

The student will be expected to be comfortable with elementary algebra, calculus, optimization (mathematical programming) and some linear algebra. The instructor will not be teaching you these basic skillsets again here. All these were taught to you in the *Mathematics for Economists A* course. In fact, much of the calculus and optimization applications here have almost identical counterparts in your study in Diploma Microeconomics. If you are weak on microeconomics, then you are advised to either first enrol in Diploma Microeconomics or to take that course concurrently. The student should also have an aptitude to learn a programming language.

These seminars have been based on several books and other sources. There is not one set textbook. The successful student in this course will be one who attends classes physically, who engages in discussions, and who independently follows up on readings referred to from these notes. Last-minute study before the end of term is a bad idea for this course. The instructor in this course is not responsible for lecture recordings.

Warning: Students are encouraged to hone their live-listening and note-taking skills and not to rely on lecture recording technologies that promise instant gratification but weaken deep learning.

CHAPTER

TWO

A BRIEF HISTORY

Key Learning Points

- Historical foundations of Macroeconomics
- Lucas Critique:
  - Economic equilibrium as statistical model
  - But statistical model may not be an equilibrium

Some say the field of Macroeconomics began with John Maynard Keynes' 1936 *magnum opus*, *The General Theory of Employment, Interest and Money*. Keynes had catalyzed a metaphysical split between the study of individuals and firms at the microeconomic level (i.e., microeconomics or price theory) from the study of an economy as a whole, and its systemic relation to aggregate government policy.

Keynes' verbal narrative alluded to the idea of the economy as a mechanical, dynamical system consisting of interacting sectors of the economy (e.g., consumption, investment, government). Elsewhere in Eastern Europe, a similar idea was in place (Michal Kalecki). But since Kalecki wrote in Polish, nobody in the English-speaking world paid attention! [Ka1935] had also applied the field of mathematical dynamics to think about the Business Cycle (or Trade Cycle as people were wont to call it in the old days).

The early formalization of Keynes' idea took the form of the famous IS/LM model. This stuff is taught in elementary undergraduate macroeconomics courses today, albeit in modified forms as the IS-LM-MP or AD-AS model. This modelling device was invented by Sir John R. Hicks, in [Hi1937]. While it was an instructive mathematical device, it was a static one. As a consequence, one could not explicitly analyze dynamic phenomena like the evolution of capital, equity, government debt and so on. From Hick's toy IS/LM representation, Jan Tinbergen and others (e.g., Lawrence Klein and Arthur Goldberger) grew larger econometric beasts from the original Keynes-Hicks macroeconomic paradigm [HH1989]. These models were used in many countries for macroeconomic forecasting and counterfactual policy experiments. A crucial feature of these macroeconometric models was that they did not require specifications of economic decision making at the level of agents: households and firms. This allowed for *unrestricted parametrization* of the model

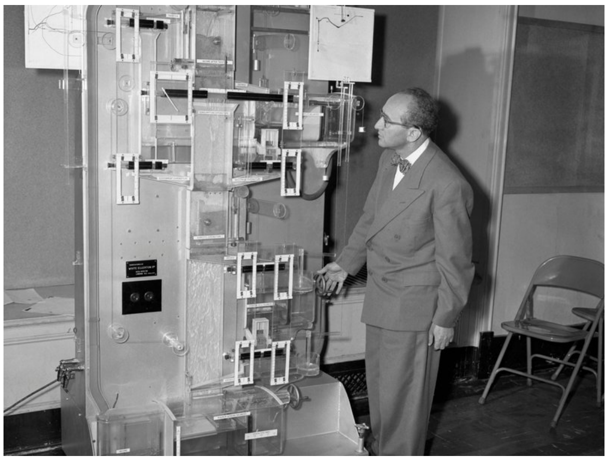


Fig. 1: The MONIAC IS/LM model Source: Smithsonian Magazine [Fe2014]

structure. In a way, this gave a lot of flexibility to the econometrician, since one could parametrize the model freely (e.g., by adding arbitrary causal channels and dependency on lagged information) in order to fit historical data.

On another spectrum—and in fact, chronologically preceding the works of Kalecki, Keynes and Hicks—the mathematician Frank Ramsey had solved an important problem on optimal saving in 1928: see [Ra1928].

> This was a dynamic optimal decision-making problem. Its solution was an optimal savings *function*, as a function of time. Central to the characterization of this solution was a mathematical object called the Euler-Lagrange *functional equation*. (This was a differential equation.) Thus, being able to solve this equation meant one was able to figure out the "best" path a decision maker in the model should take over time.

Ramsey's contribution to Economics would not impact on the practice of modern macroeconomics until the 1960's when people like David Cass, Tjalling Koopmans, Kenneth Arrow and Mordecai Kurz paid attention to Ramsey's pathbreaking application of the calculus of variations in his 1928 paper.

Much of modern macroeconomic theory is still based on similar functional analytic problems. However, our modern problems will be generalized to much more sophisticated settings (e.g., with many actors making dynamic decisions) and economically more difficult questions (e.g., how to characterize optimal government policy in the face of dynamic informational and incentive constraints; and how to solve a dynamic economy when the notion of a state variable includes an infinite-dimensional object, a distribution over agents with heterogeneous attributes).

You might ask: Why are there so many different macroeconomic models? How are they different? First, horses for courses: It is an extremely difficult task to have a single model of everything; it will be just too complex to understand what's going on in the model.

> Try building a map of your city with every detail in it: from every blade of grass to every dog peeing on a tree. See how useful such a map of your world

will be!

Second, what can vary, from one model to another, will be part of the economists storyboard:

- the set of actors (private and public agents),
- agent preference sets (and their functional representations),
- assumptions about information (including sources of uncertainty),
- commodity spaces (and production sets),
- market pricing mechanisms (e.g., perfect competition, monopoly, bilateral bargaining).

By the 1970s, economists were concerned with the failure of macroeconomic policy in alleviating recessions during the Oil Crises. The more governments tried using fiscal and monetary policy to stabilize aggregate demand, the more recessionary their economies became. Some economists began looking to a new bechmark paradigm where aggregate economic outcomes were consistent with one's story of microeconomic behavior and the micro-agent's expectations of future events (including government policy plans). This led to the so-called *rational expectations* revolution.

The notion of rational expectations was not meant to be taken to be a definitive concept of how people formed beliefs about future policies and events. It was nevertheless a useful equilibrium concept for a modeller to capture the idea that an an estimated "reduced-form" econometric model (along the lines of Tinbergen's early work) may not be stable in the face of policy changes. You see, up until then, policy analysts were doing conterfactual policy experiments on their estimated Keynesian macroeconometric models, while presuming that the structure of their models (i.e., magnitude and direction of causal relations) were invariant to their policy experiments. In short, they treated the economy and market participants like a physical engineering system where the underlying atoms are not sentient (i.e., they do not think or make decisions).

However, in a sentient economy, agents can look ahead and form beliefs about future events, including policy changes. This means that they can form best responses to anticipated policy changes. The resulting equilibrium representation—i.e., the statistical law or mathematical counterpart to Keynesian macroeconometric structures—may actually shift, in both magnitude and direction when policies change. That is, the statistical process for the economy may not be policy invariant. For example, up until then, economist presumed that there was a stable statistical inflation-unemployment trade-off given by the Phillips curve. This statistical relationship formed the basis for most Keynesian models' monetary policy trade-off: Policy makers decide on a particular policy outcome and locate the economy as a point on that trade-off curve. However, [Lu1972] gave an example of the Phillips curve inverting, when agents anticipate prolonged government intervention in monetary policy, thus nullifying the effectiveness of the policy. (See also [Lu1976].)

Thus, the so-called Lucas Critique and its related rational expectations equilibrium concept, was not so much a push to indoctrinate economists and the general public with some right-wight economic policy agenda, as many popular writers mistakenly label it. Lucas' point was to provide a counter-example showing the perils of policymaking based on mechanical/statistical models of the economy; i.e., econometric models that do not build in the contingency of sentient, forward-looking and optimally reactive agents (i.e., "the markets" in popular parlance), in the face of policy change.

In contrast to the Old Keynesian macroeconometric modelling philosophy, the modern microfoundation approach now places a lot of *logical discipline*, or *restrictions on (equilibrium-determined) parameters* of the resulting probability model vis-a-vis observed data. While a benefit in terms of delivering consistency in narratives in terms of internal model logic, these models made the life of the econometrician very difficult. (Can you think why?) As a consequence, for a few decades, there had been a marked bifurcation in research paths between macroeconomic modellers and time-series macroeconometricians. However, we now live in exciting times: With recent technological advancements in computing and also in econometric theory, researchers have again found new interest in estimating and testing microfounded macro models with data.

In this course, we will learn about some of these modern approaches to macroeconomics, by example. As a consequence, we will learn to tame different horses for different courses. We close this chapter with two questions: So why is the practice of macroeconomics so controversial? Does that not make the field exciting to study and to work in?

CHAPTER

THREE

ECONOMIES AS DYNAMICAL SYSTEMS

Is there a common theme underlying the different models that we will study, models that spanned so many decades and modelling philosophies?

Key Learning Points

- State Variables
- What is a Markovian or recursive map?
- Deterministic and Stochastic recursive maps
- Continuous and Discrete random variables

3.1 Reduced-form representation

Fortunately, the answer is **yes**.

The similarity is in the "final" mathematical or statistical form, i.e., the "reduced-form" representation of a model's dynamics.

What may differ within the generic form are the different theories' implied *magnitudes* and *directions* of causal relationships.

Even if the models vary in their underlying assumptions or economic-political philosophies, at the end of the day, after we have solved our agents' decision problems and their interactions through markets, our model's (dynamic) equilibrium can be *represented by* a *recursive* or *Markovian* state-space form.

This vital *recursive mapping* will suffice to describe the evolution of the model economy. End users can then use their preferred model and its resulting *recursive system* to do things like, fitting the model to the data and testing the model empirically; or performing counterfactual policy analyses and simulations.

3.2 Recursive maps by example

Throughout this course we will focus on models that will imply a linear probability model form.

Let's begin with a simple one. Suppose for now, we let a macroeconomy be represented by a simple probability model.

Warning: Note that there is no *explicit economic theory* behind this model (yet)! The (implicit) workflow here, as was the case of old-fashioned policy modelling, is

- Write down a statistical model
- Obtain values for *unconstrained parameters*. (Hold this thought for now; we'll come to this point later).
- Then weave an *economic/policy narrative* around the statistical relationship.

So, for the moment, we are undertaking a modelling exercise like that of the policy modellers from the 1950s-1960s. A reincarnation of such policy modelling practice still survives today as statistical Vector Auto Regression models.

To make sure we all start from the same page, consider an linear probability model,

$$Y_t = \beta X_t + \gamma \varepsilon_t, \qquad \varepsilon_t \sim \varphi$$

where

- β and γ are given parameters;
- Y_t is some endogenous variable, say, real GDP (let's call this the *state of the economy*);
- X_t is some variable that is observable—to both the allegorical agent in the model economy and to the statistician/modeller—at data location t. Suppose the index variable t keeps track of natural time. Assume time is countable: $t \in \mathbb{N} := \{0, 1, 2, ...\}$; and
- ε_t is an exogenous shifter to the state of the economy which is governed by a given probability distribution φ.

This linear probability model should be familiar to the reader with a first-course in statistics or econometrics training.

What if we make the content of X_t, explicit?

Example 1

$X_t := Y_{t-1}$.

Plug this into the previous linear probability model to get

$$Y_t = \beta Y_{t-1} + \gamma \varepsilon_t, \qquad \varepsilon_t \sim \varphi$$

This is a scalar first-order linear *stochastic difference equation* (LSDE). Let's keep working with the reduced-form model of an economy for now. Consider:

Example 2

$X_t := Y_{t-1}$ and $\varepsilon_t = 1$ for all t.

Plug this into the previous linear probability model to get

$$Y_t = \beta Y_{t-1} + \gamma,$$

As long as $\beta \neq 1$, a *particular solution* to this *deterministic* difference equation exists, and takes the form of a constant solution:

$$Y_t = Y = \frac{\gamma}{1-\beta},$$

for all $t \in \mathbb{N}$.

We often call this solution Y a *stationary point*, or, a *steady state*.

Exercise

Show that the *general solution* to this difference equation is

$$Y_t = (Y_0 - Y)\,\beta^t + Y.$$

1. Explain in words what this function says.
2. Pick some numerical values for the parameters (β, γ). Write a Python code and plot the *general solution* as a graph in (t, Y_t)-space.
1. Show what happens to your graph if $|\beta| > 1$?
2. Show what happens to your graph if $|\beta| < 1$?

3.3 Linear stochastic difference equation systems

Consider now the generalization of the scalar (univariate) example above. We often call these *linear stochastic difference equations* (LSDE):

Note:

- READ: `PDF notes on LSDE`

Many empirical and policy models with have (approximately) solutions of this form. In this course, we will begin with models where their individual solution is either an exact, or an approximate, linear recursive self-map. As in all the toy examples above.

What if we have a model with arbitrary dependency of its *current state* on an arbitrarily long record of its past? Not a problem. We will see that in general, we can re-define the problem to make it Markovian or recursive again. The trick will be in expanding the notion of the model's state space and re-defining appropriate "dummy" or auxiliary state variables.

3.4 Postscript

Now take a look back to what we've done here; then look ahead to the rest of this course. Many models will generally have *solutions* to their respective decision-making or equilibrium concept in the form of a recursive function:

$$x_{t+1} = F(x_t, w_t)$$

where $(x, w) \mapsto F(x, w)$ is some general (possibly nonlinear) recursive map, x_t represents the state vector of the model economy, and, w_t is some forcing process exogenous to the model system. To reiterate, we will only focus on the case that the mapping F is of a linear form.

You might ask: What is the point of all this "mathiness"? The goal is use this resulting recursive map to simulate and study the dynamic behavior of the model economy, under various economic policy scenarios. In short, the model is a laboratory for potential policy experiments, albeit an imperfect or inaccurate one. Most economists have to resort to inaccurate models to study policy counterfactuals simply because randomized control trials are difficult, especially when the subjects under study is a who economy.

CHAPTER

FOUR

KEYNESIAN BUSINESS CYCLES AND POLICY

We begin with the orthodoxy in macroeconomic textbooks and in most central-bank policy thinking: Keynesianism. The goal here is to ensure that we all understand and master the methods and insights from this school of thought, before we begin wondering about alternatives.

Like a good mix tape, we'll start with some easy tracks and then we'll ramp it up. Finally, we tear it all down with some critical (and open) questions. There are still new riffs to be played . . .

Key Learning Points

- "Short run" and "business cycles" metaphysical construct
- The textbook Keynesian view of the "short run" reality
- A working (numeric) version of your Keynesian IS-MP-PC model
- O Phillips Curve, where art thou?
- Pitfalls of reduced-form policy modelling

4.1 Cartoon theorizing

We first re-visit a version of the textbook Keynesian model geometrically. The following slides based on material found in an intermediate textbook like [Mi2015] or [Jo2014] will help set the stage in terms of developing intuition and basic lessons.

1. `Empirical Motivations`
2. `Demand Side (IS)`
3. `Supply Side (PC)`
4. `Monetary Policy (MP)`
5. `Ensemble c'est tous (IS-PC-MP as AD-AS)`

6. `Applications 1`
7. `Applications 2`
8. `Application 3`

4.2 Learning the nuts and bolts

Once we've got these bad boys nailed down, we'll start developing slightly more R-rated (R for "rigorous") versions of these ideas using some mathematical and computational tools. We'll do so in our TutoLabo sessions.

Note: Why the mathematical discipline? Think/Discuss.

- How would you take your IS-MP-PC ideas to the data directly?
- The intermediate textbook versions of IS-MP-PC give you *qualitative* predictions. How would you attempt to deduce any *quantitative* results from it?
- Dani Rodrik: "[W]e use math not because we are smart, but because we are not smart enough".

"

4.3 Caveat emptor

Recall how we mentioned that Macroeconomics is a difficult subject? We ask big questions, so we set ourselves up for a big fall too. No matter how *precise* we get in our modelling exercises as economists, our models are very likely to be *inaccurate* when confronted with reality:

> Indeed, the textbook Keynesian IS-PC-MP logical paradigm for macroeconomic policy thinking may have some flaws—both in its internal logic and also in terms of its external validity (with respect to observational data). In the following set of slides, we'll first attack the question of observed data and the Keynesian theory's external consistency. Then we'll study a simplified version of the famous Lucas Phillips curve model and show that the so-called Phillips curve correlation may not be a stable one, as seen in some recent data.

The mantra of the week here is

> "In macro data and policy, WYSMNBWYG".

(WYSMNBWYG stands for "What You See May Not Be What You Get".) Lucas' lesson from 1972 warns us about modelling economic behavior as fixed statistical correlation models. To paraphrase the Greek story from Virgil's *Aeneid* or Homer's *Odyssey*, one should beware of (some behavioral) economic modellers bearing free parameters (i.e., a model whose assumption—as opposed to result—is "too close" to the observed statistical behavior to be modelled.)

1. `Slides on recent "Phillips curve" data`
2. `Notes on Williamson's article`
3. [Lu1972] `Lessons from 1972`
4. Steve Williamson's critique

IN ORDER TO DETERMINE WHAT ECONOMIC INTERVENTIONS ARE BEST IN DEVELOPING COUNTRIES, WE MUST USE RANDOMIZED CONTROLLED TRIALS.
THESE TRIALS ARE BASED ON THE SUCCESSFUL METHODS USED TO EVALUATE MEDICAL DRUGS. THUS, FOR OUR STUDY, WE LOOKED AT TWO GROUPS.
ONE GROUP WAS GIVEN DIRECT CASH INJECTIONS. ANOTHER WAS GIVEN "PLACEBO CASH," WHICH WERE PIECES OF PAPER WITH HASTILY-DRAWN PICTURES OF GEORGE WASHINGTON ON THEM.
THE PLACEBO GROUP POSTED PHOTOS ONLINE OF US GIVING THEM FAKE MONEY, CAUSING WORLDWIDE OUTRAGE.
THIS LED TO MASSIVE CONTRIBUTIONS OF CLOTHING, FOOD, EDUCATIONAL RESOURCES, MEDICINE, AND NON-PLACEBO CASH TO THE PLACEBO GROUP.
AS A RESULT, THEY EXPERIENCED A SIGNIFICANT DECREASE IN CHILD MORTALITY AND AN INCREASE IN HOUSEHOLD WEALTH, FAR OUTPACING THE CASH-INJECTION GROUP DURING THE FIVE YEAR PERIOD OF STUDY.
THE POLICY IMPLICATION IS CLEAR:
GIVEN THE HIGH COST OF BEING EMPIRICALLY WRONG, WE RECOMMEND A RETURN TO BEING THEORETICALLY WRONG.
WILL THAT IMPROVE LIFE FOR THE POOR?
OUR MODEL SAYS YES!
smbc-comics.com

CHAPTER

FIVE

ECONOMIC GROWTH

Key Learning Points

- Revisit our first encounter with economic dynamics—Solow-Swan Model
- Appreciate relevance of Solow-Swan in accounting for observed empirical regularities in long run economic data
- Use the Solow-Swan Model as a starting point for more modern extensions later
- Learn basic techniques of dynamic economics including scientific computing
- **Ac**, Ch. 2-3; See the growth chapters in Charles I. Jones' undergraduate textbook for a refresher.

Let's fast-forward a bit from Keynes' world of (short-run) macroeconomics, for now, to the 1950s. We will revisit the prototype model of economic growth, due to Robert Solow and Trevor Swan. This framework later served as the backbone of modern business-cycle and macroeconomic policy modelling. In fact, if you looked at the flow of modern undergraduate textbooks, e.g., [Jo2010], and also the trend in modern macroeconomic and policy research, you'd see this chronologically nonlinear, but logically linear, pattern in the history of macroeconomic thinking and policy modelling:

- Solow model (and its other endogenous growth variants) as the basis for the long run growth path of economies.
- Aggregate shocks (New Classical) and policy-relevant short run market frictions (Keynesian, New Keynesian, Non-Walrasians or New Monetarists) as the source of temporary departures from the long run growth path—i.e., business cycle fluctuations.
- All of the above plus individual shocks to preferences, technologies, life-cycle problems, or incomplete market trading opportunities, as the source of agent heterogeneity, wealth and consumption inequality over the long run and/or over the business cycle. (This opens up new doors for the analysis of redistributive policies across individuals within a period and across time and generations.)

Note: The recent speech of Janet Yellen (US FRB President), entitled "Macroeconomic Research After the Crisis", is a good synopsis of what's important in macroeconomics today.

So let's come back to the basics again: The long run growth model and its implied dynamic outcomes. Let's motivate our study by recalling some empirical motivations.

5.1 Empirical Regularities

Around the time Trevor Swan (then at the ANU) and Robert Solow (MIT) were thinking about a theory of growth, Nicholas Kaldor [Ka1961] documented these facts on long-run data (for industrialized countries):

- The shares of labor and physical capital in GDP are nearly constant
- The ratio of physical capital to output is nearly constant
- The rate of return to capital (or the real interest rate) is nearly constant
- Physical capital per worker grows over time
- Per capita output grows over time, and its growth rate does not tend to diminish
- The growth rate of output per worker differs substantially across countries, but the rates tend to converge

See also an updated version of Kaldor's exercise by Chad Jones [Jo2010] which also deals with new facts on human capital, R&D, institutions and etc.

5.2 Population Growth

Let's tame this creature. I'm pretty sure many of you have learned about this model *ad nauseum*. But let's do this once more!

First, we list down the basic notations in a version of Solow-Swan:

- Time is denumerable. We index each date/period by $t \in \mathbb{N} := \{0, 1, ...\}$.
- The stock of population/workers at date $t \in \mathbb{N}$ is L_t. Without loss, assume $L_0 = 1$. Population grows at constant rate $n \geq 0$.
- The measure L_t of identical agents work, consume (C_t), saves/invests (I_t) and produces a single good (Y_t).
- The economy is closed.

5.2.1 Technology

There is a production technology that converts capital stock and labor into output:

$$Y_t = F(K_t, L_t) \tag{5.1}$$

We assume available capital at the start of date $t \in \mathbb{N}$ is used for producing output Y_t and in that process a fraction δ of capital disappears or wears out. There is a technology that converts the remaining proportion $(1-\delta) \in (0,1)$ of date-t initial capital stock (K_t) and investment flow I_t into next period productive capital, K_{t+1}:

$$K_{t+1} = (1-\delta)K_t + I_t \tag{5.2}$$

5.2.2 Saving

The agents consume according to a rule of thumb:

$$C_t = (1-s)Y_t \tag{5.3}$$

The parameter $(1-s) \in (0,1)$ is called the *propensity to consume*.

The agents are not wasteful; so what is not consumed must be saved:

$$S_t = sY_t \tag{5.4}$$

5.2.3 Characterizing Solution

Since the economy is closed, all savings must be invested, and the only investment opportunity is into capital formation. Recall the circular flow diagram in first-year macroeconomics? Here we have:

$$S_t = I_t \tag{5.5}$$

So there we have it. Combine (5.5), (5.4) and (5.1) into (5.2), and we have a description of the Solow-Swan growth dynamics:

$$K_{t+1} = (1-\delta)K_t + sF(K_t, L_t). \tag{5.6}$$

Don't forget the description of population dynamics:

$$L_{t+1} = (1+n)L_t \tag{5.7}$$

So together, given some initial states (K_0, L_0), the difference equations (5.6) and (5.7) characterize the trajectory of the Solow-Swan model economy, $\tau(K_0, L_0) := \{(K_{t+1}, L_{t+1})(K_0, L_0)\}_{t\in\mathbb{N}}$.

5.2.4 Technology and equilibrium

It turns out that the solution (trajectory) of the Solow-Swan economy depends quite a lot on the assumption on the function $F : \mathbb{R}_+^2 \to \mathbb{R}_+$. So let's put some more structure on it.

We assume that the function F has these properties:

1. F is twice-continuously differentiable on the set $\mathbb{R}_+^2$.
2. F is homogeneous of degree one in its inputs. (What is another economics jargon for this mathematical property?)
3. The value of F is increasing in each input at a decreasing rate. How would you formalize these properties in terms of derivative functions of F?
4. F satisifies the Inada conditions:

$$\begin{aligned}
\lim_{a \searrow 0} \frac{\partial F(a,b)}{\partial a} &= +\infty \\
\lim_{a \nearrow +\infty} \frac{\partial F(a,b)}{\partial a} &= 0 \\
\lim_{b \searrow 0} \frac{\partial F(a,b)}{\partial b} &= +\infty \\
\lim_{b \nearrow +\infty} \frac{\partial F(a,b)}{\partial b} &= 0
\end{aligned}$$

Exercise Technology and equilibrium (1)

1. An example of F that satisfies these assumptions is the Cobb-Douglas production function, where $F(K, L) = K^{\alpha}L^{1-\alpha}$ and $\alpha \in (0, 1)$. If the real rate of return on renting capital (labor) is equal to the marginal product of capital (labor), show that in this instance, capital income as a share of total production is α. Likewise, show that labor's share is $1 - \alpha$.
2. Show that these assumptions imply that if at least one input is zero then output is also zero. ("It takes two to Tango" as it were!)

Now given the assumption that F is homogeneous of degree one, we have the following property:

$$F(K, L) \times \frac{1}{L} = F\left(\frac{K}{L}, 1\right) := f(k),$$

where $k := K/L$. Using this fact, do the following exercise:

Exercise Technology and equilibrium (2)

1. Show that the two difference equations (with two state variables), (5.6) and (5.7), can be re-written in terms of a scalar state variable k_t as

$$k_{t+1} = g(k_t) := \frac{(1-\delta)k_t + sf(k_t)}{1+n}. \tag{5.8}$$

Observe that the function $g : \mathbb{R}_+ \to \mathbb{R}_+$ is a self-map. It takes points in the set of possible capital stocks back into the same set. Conventionally, we write $g^t(k)$ to mean the value of a t-fold application of the map g on the point k; or equivalently, this is written out as the value of the composite function $g \circ g^{t-1}(\cdot)$ at the point k, where $t \geq 1$ and $g^0(k) = k$.

We define a *deterministic steady-state point*, k^*, to be an outcome such that

$$k^* = g(k^*). \tag{5.9}$$

That is, it is a *fixed point* of the mapping g. In other words, if the economy is at such a point k^*, and repeated application of the mapping $g^t(k^*)$ for any $t \geq 1$ results in again k^*, then we say the economy is at a (deterministic) steady state. Where the context is understood below, we will use the term *steady state* in lieu of *deterministic steady state*.

The next exercise requires a bit of high-school mathematical analysis on the real line.

Exercise Technology and equilibrium (3)

1. Prove that given the restrictions on F above, there exists a unique steady state equilibrium in terms of the sufficient state variable k that is nontrivial (i.e. there is positive production).
2. We say that the (scalar) dynamical system (5.8) has a fixed point $k^* \in (0, \infty)$ that is *stable* if for an $\varepsilon > 0$ there is some $\delta \in (0, \varepsilon)$ such that for all $T \geq t \in \mathbb{N}$, $|g^t(k_0) - k^*| < \delta$ implies that $|g^T(k_0) - k^*| < \varepsilon$. A more restricted notion of stability is as follows: We say that the state k^* is an *asymptotically stable* fixed point of the (scalar) dynamical system (5.8) if it is *stable*, and, there is a $\delta > 0$ such that if $|g^t(k_0) - k^*| < \delta$ for any $t \in \mathbb{N}$, then $|g^T(k_0) - k^*| \to 0$ as $T \to \infty$. Can you try verbalizing these definitions, using words your sixteen-year-old self would have understood?
3. Argue that the self-map in (5.8) is increasing. Using the definitions of stability above, prove that the nontrivial steady state equilibrium is asymptotically stable and that it is unique.
4. Now verify that the unique nontrivial steady state $k^* \in (0, +\infty)$ solves

$$(n+\delta)k^* = sf\left(k^*\right).$$

5. Can you explain in words what the last condition says? (Be mindful of the economic interpretation here.) What about another $k^{*\prime} = 0$? Is this also a steady state of the economy?

6. Can you construct a counterexample of F where the steady-state equilibrium is not unique?

Now let's make use of the property that F is homogeneous of degree one in the following warm-up exercises. The purpose of these exercises is for developing the computational side of our skillset, so you are strongly encouraged to work through them carefully.

Exercise Technology and equilibrium (4)

1. Outline an algorithm for solving for the trajectory $\tau(K_0, L_0)$.
2. You can actually solve for $\tau(K_0, L_0)$ given parameter values by hand (maybe with the help of a calculator or spreadsheet). Try "coding" this up in a spreadsheet software. All we need is to instantiate the production function $(K, L) \mapsto F(K, L)$. Let's pick a Cobb-Douglas example: $F(K, L) = K^{\alpha}L^{1-\alpha}$. Pick some arbitrary numbers for the parameters for now: $\alpha = 1/3$, $\delta = 0.10$, $n = 0.015$, and $s = 0.15$.
3. Now, let's do this instead using a programming language like Python.
4. Once you get the Python example running, try different parameter values. Provide economic explanations disciplined by the Solow-Swan model structure for:
   - What happens if s is increased?
   - What happens if α is bigger?
   - What happens if δ is bigger?

5.3 Decentralized Economy

Now let's return to the Solow-Swan model. Notice how we had a story as if there were a single agent/planner for the whole economy? Or equivalently, economists call this a centralized problem. In most countries, the economy is decentralized and allocations of resources are somehow coordinated through some market pricing mechanism(s). Let's see how that looks like in this artificial economy.

Now consider the same model environment but with a different structure of ownership:

- There are three markets: final good, capital rental and labor rental markets
- Households own the initial capital stock and supply all their labor endowment to firms
- A continuum of identical and perfectly competitive firms (in both output and input markets) rent capital and labor services from households
- The representative firm maximizes profits—i.e. the difference between its total

revenue and total cost

Without loss, let's normalize the absolute price of the final good to unity. So then, we can let r and w, respectively, denote the relative prices of renting capital and labor determined in the competitive input markets. These relative prices are sometimes called the real return on capital and the real wage, respectively.

Note: A Matter of Accounting

If the rental rate r_t is what the competitive firm faces in the capital rental market, then from the household/capital-owner's accounting perspective, each additional unit of capital rented out should return a rate r_t^h over and above the depreciation rate of capital (δ), the latter being a consequence of use in production over date t. So we have:

$$r_t = r_t^h + \delta.$$

Exercise Decentralized Economy (1)

1. Write down the profit function of the firm in terms of claims to units of the final good.
2. Show that the first order conditions characterizing the firm's optimal demand for capital and labor services are, respectively:

$$r_t = \frac{\partial F(K_t, L_t)}{\partial K_t},$$
$$w_t = \frac{\partial F(K_t, L_t)}{\partial L_t}.$$

3. Define a *competitive equilibrium* for this economy. This will be an infinite sequence of allocations and relative prices that must satisfy certain restrictions. Spell them all out clearly.
4. National Accounting done Three Ways: Show that in a competitive equilibrium the household's total income must be equal to the total production of the economy. In turn these must all be equal to the total expenditure demand in the economy.
5. Verify that in competitive equilibrium the firm earns zero economic profits.
6. Can you modify your Python programs earlier to now incorporate this more general version of the Solow-Swan model?

5.4 Exogenous Growth

Consider a variation now where the production technology is represented by

$$F(K_t, A_t L_t). \tag{5.10}$$

The new variable, A_t is assumed exogenous and is often called Harrod-neutral or labor augmenting technical progresss.

Assume

$$A_{t+1} = (1+g)A_t; \tag{5.11}$$

and A_0 is known.

For there to exist a steady state fixed point in this model's competitive equilibrium, we will have to work with a change of variables. Define original level X_t of a variable to be in efficiency units by $\tilde{x} := X_t / A_t L_t$, where $X \in \{C, I, Y, K\}$.

Exercise Exogenous Growth (1)

Assume again F is Cobb-Douglas with $\alpha \in (0,1)$.

1. Show that the competitive equilibrium you defined just now can be distilled into a single difference equation in terms of capital per efficiency units of labor $\tilde{k}$:

$$\tilde{k}_{t+1} = \frac{(1-\delta)\tilde{k}_t + sf\left(\tilde{k}_t\right)}{(1+n)(1+g)}; \qquad f(\tilde{k}) := \tilde{k}^{\alpha}. \tag{5.12}$$

2. Show that there is a unique non-trivial fixed point representing the transformed model's steady state equilibrium indexed by:

$$\tilde{k}^* = \left(\frac{s}{(1+n)(1+g) - (1-\delta)} \right)^{\frac{1}{1-\alpha}}. \tag{5.13}$$

3. Verify that this model, along the steady state equilibrium path, can address all six of the stylized facts of growth in industrialized nations, listed at the beginning of this chapter. (This is a longer exercise worth doing!)

5.5 Golden Rule

Notice that in a steady state, sometimes called a *balanced growth path*, we have capital in efficiency units in equation (5.13) being constant. But it is a function of the parameter s, *inter alia*. So let's make explicit this dependency for the purposes of our study here and write $\tilde{k}^* \equiv \tilde{k}^*(s)$.

Here's a normative economics question: Suppose a planner outside of this model competitive equilibrium were able to manipulate household's saving behavior—by changing the parameter s. Since we are talking about normative issues, you might wonder, in public finance or welfare economics one typically thinks in terms of a measure of agents' welfare or utility representations. In the Solow-Swan model, these objects are not defined. So our notion of "what ought to be better or best" will be measured in terms of consumption. In particular, we ask: What is the *steady-state consumption-maximizing* level of the savings rate, s_{gold}?

Exercise Golden Rule (1)

1. Assume our friendly Cobb-Douglas function again for production. Show that the golden-rule savings rate is $s_{gold} = \alpha$ in this example.
2. Dynamic (In)efficiency: Explain what happens to the balanced-growth path consumption if $s > s_{gold}$? Explain what happens to the balanced-growth path consumption if $s < s_{gold}$?

Note: Later, when we deal with the topic of optimal growth, we will revisit this exercise again to compare with the implications from the optimal growth model.

CHAPTER

SIX

HETEROGENEITY AND REDISTRIBUTION

In this chapter, we take baby steps to start thinking about fiscal and monetary policy (again) from the point of view of a model environment with agent heterogeneity. We'll use the famous overlapping generations (OLG) model. In contrast to our earlier studies of more "reduced-form" Keynesian theories of macroeconomic policy, now the economic insights are more profound as we can relate equilibrium outcomes to underlying taste, technology and policy features of the model environment. As a consequence our policy conclusions are more nuanced.

The OLG model is a natural metaphor for an economy where agents may behave differently as a function of their heterogeneous types. It is also a useful simple prototype to help us think about the problem of missing (i.e., incomplete) markets, whereby government policy intervention may have a welfare-improving and redistributive role to play.

In this course, we only focus on the dimension of heterogeneity that is age. The purpose of our study here is not to immediately have models that have empirical and quantitative policy impact.

> To get there, one has to banish that impatience and the lust for instant gratification. Instead, one must be long suffering and slowly appreciate the art of model building and scientific analysis along the way to being able to work with or develop fully-fledged *quantitative theory* models.

The simple models here will allow us to draw some *qualitative* and *more nuanced insights* into whether particular fiscal policies help restore economic efficiency, and if so, how they might do so. (More complex and computational versions of such models are used in academia and policy to not only think about age-dependent policies, but also wealth- and health-dependent tax/transfer schemes.)

Also, we'll use a version of the framework to also think about why money exists. In our monetary OLG setting, the lessons are somewhat more metaphorical, although the OLG model sets up the stage for later generations of deep monetary theory (a.k.a. New Monetarism) models that are more transparent about the connection between information/contractual frictions in the exchange process and the rise of money and/or other liquid financial assets. In many New Keynesian and the faddish behavioral/agent-based (or post-Keynesian) models, the underlying reasons for why governments should tax or control inflation, or regulate financial markets are often not precisely modelled. If they are, they are often modelled by free parameters. One ought to be very suspicious of economists who purvey policy salves for problems that they have not bothered to model and rationalize, but instead have conjured up from a free parameter.

Key Learning Points

- OLG model with age heterogeneity
- Focus on long term fiscal issues
- Social security and intergenerational redistribution
- Money and Banking

6.1 Diamond-Samuelson OLG model

1. Diamond-Samuelson OLG
2. Efficiency and OLG

6.2 Fiscal Policy and OLG

1. Fiscal Policy 1
2. Fiscal Policy 2
3. Fiscal Policy 3

6.3 Money and Inflation Control

1. Why Money? An OLG rationalization
2. Inflation
3. O Phillips Curve, where art thou?

6.4 Money and Banking

1. Financial Intermediation

6.5 A taster of business cycles modelling

1. Aggregate shocks 1
2. Aggregate shocks 2

CHAPTER

SEVEN

A QUICK TUTORIAL IN PYTHON TOOLS

Key Learning Points

- Editors and Ipython
- Jupyter Notebooks
- Python Lists
- Python Dictionaries
- NumPy and SciPy classes
- Plotting with MATPLOTLIB
- Conditionals
- Loops
- Statistics

A lot of models are quite sophisticated these days. Solving the models and evaluating the solutions are often non-analytic tasks. As a result, computational economics is a vital skillset for the modern economist to master. In this chapter, we will take a crash course on the basics of numerical computation in the Python language. While this is a very short and incomplete treatment on Python programming, fret not, as we will pick up more skills when we tackle specific problems. The best way to learn a language is while you're fearlessly writing and speaking it!

7.1 Getting started

1. Go to the Anaconda distribution of Python and down the installer. (Choose the latest *Python 3.xx* version.)
2. Follow the instructions to install.
3. When you're done you can start having fun with Python.

In this course, we will mostly use the Jupyter Notebook which allows you to write notes and do live computing within the notebook itself. I found this to be a great tool, not only for learning, but also for communicating works in progress with my research collaborators.

Jupyter should be installed with your Anaconda distribution. If not install it:

```
conda install jupyter
```

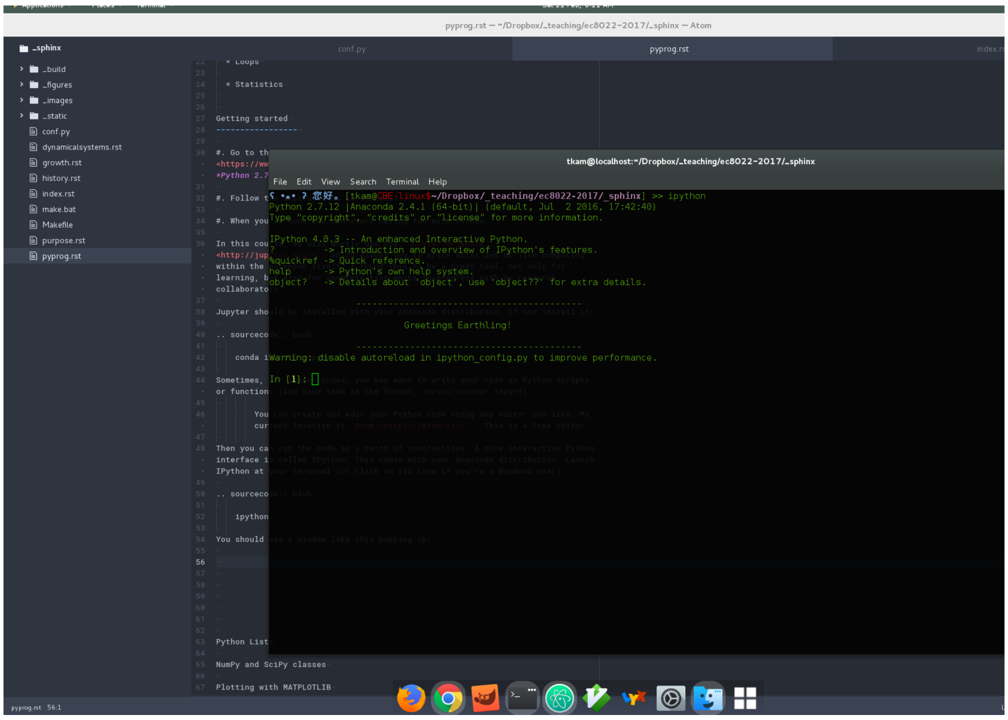
Sometimes, for longer codes, you may want to write your code as Python scripts or functions (and save them in the format, *mysourcecode*.py).

> You can create and edit your Python code using any editor you like. My current favorite is Atom. This is a free editor.

Then you can run the code as a batch of instructions. A nice interactive Python interface is called IPython. This comes with your Anaconda distribution. Launch IPython at your terminal (or click on its icon if you're a Windows user):

```
ipython
```

You should see a window like this popping up:



```
ʕ •ᴥ• ʔ 您好。[tkam@CBE-linux$~/Dropbox/_teaching/ec8022-2017/_sphinx] >> ipython
Python 2.7.12 |Anaconda 2.4.1 (64-bit)| (default, Jul  2 2016, 17:42:40)
Type "copyright", "credits" or "license" for more information.

IPython 4.0.3 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

        ------------------------------------------

                   Greetings Earthling!

        ------------------------------------------
Warning: disable autoreload in ipython_config.py to improve performance.

In [1]:
```

Fig. 1: *An Ipython terminal with Atom editor lurking behind*

We will walk through Ipython and Jupyter Notebooks soon when we get to the live **Tuto-Labo** sessions with my assistant Igor.

Python is an interpreted language. This means you write code that is very close to human language, and its interpreter converts that to machine-level code that your computer can operate on.

Some of you may have had experience with other scientific programming languages like R, MATLAB, Julia and etc. Well, Python is just like that. But in my opinion, Python's language is highly readable.

7.2 Data Types

In your modelling work you will often be dealing with different data types for variables. Typically, these are

- Floating-point numbers or "floats"
- Integers
- Boolean
- strings

We'll work through these by example.

7.2.1 Floats

Since a computer can only store discrete information, a continuous object like a *real number* can only be represented approximately.

Did you know?

The recent film *Hidden Figures* was a true story about black women working as "colored" computers and engineers for NASA in the 1950-60s. Back in the day, a computer was a human doing intensive calculations by hand. At the time, much of the calculations in solving numerical differential equations were done by hand with the aid of log tables. Later, when they began to replace human computers with IBM machines, there was a scene where the human computer detected huge errors in the machine computer's calculations of a space capsule trajectory. This was due to the limited precision of early machine computers. The human computer was also limited in precision!

(The film also highlighted the ugly face of institutionalized racism , sexism, and white privilege.)

★★★★

A floating point number is a computer's approximation of a real number, e.g., π, to some finite precision. Check this out in your Ipython terminal:

```
In [1]: import numpy as sweetpotato
In [2]: print(sweetpotato.pi)
3.14159265359
```

In Python you have to be pedantic when you intend to use floats. For example:

```
In [3]: x= 3.0

In [4]: type(x)
Out[4]: float
```

Notice the explicit use of a decimal or period after a number. This make the variable `x` a floating point representation of a real number.

7.2.2 Integers

From the previous example, supposed we had accidentally typed

```
In [7]: x= 3

In [8]: type(x)
Out[8]: int
```

Notice the lack of a decimal or period after a number. This make the variable `x` an integer now. We will quite often use integers for indexing elements in a Python list, or, a Numpy array.

7.2.3 Boolean

The Boolean data type can have values `True` or `False`. For example:

```
In [10]: x, y, z = True, False, True

In [11]: print(x,y,z)
(True, False, True)

In [12]: x+y
Out[13]: 1

In [14]: x+z
Out[14]: 2
```

Observe that we can do Boolean algebra with these data types.

7.2.4 Strings

Example:

```
In [16]: x = "Hello Dave"

In [17]: print(x)
Hello Dave
```

7.3 Python Lists, Indexing and Slicing

A Python list is a collection of Python objects. For example

```
In [16]: mylist = [ "Hello Dave", 2.0, 3.4 ]

In [17]: mylist[0]
Out[17]: 'Hello Dave'
```

The last instruction is an example of **indexing**, where we picked out the first element of `mylist` which is indexed by the integer 0.

Note: Python indexes begin from the integer 0.

Here's a quick way to get the last element of that list:

```
In [18]: mylist[-1]
Out[18]: 3.4
```

We can also do **slicing**, i.e., extract chunks of data we want from a Python list. Here's an application of this idea to the previous example's `mylist`.

```
In [19]: mylist[0:2]
Out[19]: ['Hello Dave', 2.0]
```

This example picked out the first and second elements of `mylist`.

Here's the syntax for **slicing** in general on a Python list `a`:

```
a[start:end] # members from arbitrary START to END-1
a[start:]    # members from arbitrary START through the rest of␣
↪the array
a[:end]      # items from the beginning of list through END-1
a[:]         # a copy of the whole array
```

Note: **Commenting your code lines**. Notice the `#` symbols above followed by text comments? The # decoration makes the stuff that come after it invisible to Python's interpreter and can only be read by the human.

Writing *comments* after or around your lines of code is a good programming practice. Can you imagine why? (Think about revising your code 6 years after you've first developed your code!)

We can also have a python **list of lists**, so on and so forth:

```
In [19]: mylist = [ "Hello Dave", 2.0, 3.4, [1,2,3,4] ]

In [20]: mylist[3]
Out[20]: [1, 2, 3, 4]
```

Python lists come overloaded with additional functions, like:

```
In [21]: mylist.pop(3)
Out[22]: [1, 2, 3, 4]
```

where `mylist.pop(3)` removes the last element of `mylist`. Alternatively, if you wanted to pop out the last element of a list, you could also have written `mylist.pop(3)`. Either way, we can now check what happened to the original list called `mylist`:

```
In [23]: print(mylist)
Out[24]: ['Hello Dave', 2.0, 3.4]
```

We can also do the reverse:

```
In [25]: mylist.append(["This is fake news"])

In [26]: mylist
Out[27]: ['Hello Dave', 2.0, 3.4, 'This is fake news']
```

7.4 Python dictionaries

Another useful thing is an object called a Python *dictionary*. This will come in handy when you are generating heaps of data and you want to label them and extract them later, without having to keep track of thousands of variables or objects as inputs or outputs to functions. To store these things neatly as a single object, you can use a Python dictionary.

Note: Think about your kitchen drawer: Are you the sort of cook who compartmentalizes your knives from forks, your ladles from your spatulas? If that's you, then creating a Python dictionary works the same way: It helps you organize things better.

Here's a simple example of a dictionary that maps keys to values

```
In [10]: d = dict(alice=0.1, bob=[1,2,3], christine='Is Not␣
↪Alice or Bob')

In [11]: d['christine']
Out[11]: 'Is Not Alice or Bob'

In [12]: d['alice']
Out[12]: 0.1
```

Another equivalent way to create the previous example is

```
d = {'alice': 0.1, 'bob': [1, 2, 3], 'christine': 'Is Not Alice␣
↪or Bob'}
```

7.5 NumPy and SciPy libraries

Two of the most used libraries for scientific computing are NumPy and SciPy. In this course, we will use predominantly tools contained in Numpy.

7.5.1 NumPy Lists, Indexing and Slicing

From a Python list we can convert it into a Numpy array:

```
In [39]: import numpy as np

In [40]: y = np.asarray(mylist)

In [42]: print(y)
['Hello Dave' '2.0' '3.4' 'This is fake news']

In [43]: type(y)
Out[43]: numpy.ndarray
```

We can convert it back to a Python list when required:

```
In [44]: z = y.tolist()

In [45]: type(z)
Out[45]: list
```

We will work with NumPy arrays a lot in our numerical computations later.

Note: Have a tour of NumPy's tutorial here and also this here for SciPy. We will use SciPy later for tasks like solving optimization problems, and, function approximation.

Here's an example for creating a two-dimensional matrix or NumPy array:

```
In [17]: A = np.array([[1, 2, 3], [4, 5, 6]])

In [18]: A
Out[18]:
array([[1, 2, 3],
       [4, 5, 6]])

In [19]: A.size
Out[19]: 6

In [20]: A.shape
Out[20]: (2, 3)
```

In the above example, what we've done is stack two 1×3 row vectors vertically in a list, and using the NumPy class' function called `array()`, we turn the list into a NumPy matrix. The last two command inputs, respectively, check the number of elements in the matrix `A` and the dimension of that matrix.

The reason we will often work with NumPy arrays is that in NumPy and SciPy, we can avoid writing loops to speed things up. NumPy arrays come overloaded with many useful functions that can be used to perform quick operations on an array itself. To see what these functions are, take the `A` matrix created earlier. Now, type a period after A, and hit the "Tab" key on your keyboard:

```
In [23]: A.
A.T               A.argsort         A.compress        A.cumsum          ␣
↪A.dumps            A.imag            A.min             A.prod          ␣
↪   A.reshape         A.shape           A.sum             A.tostring
A.all             A.astype          A.conj            A.data            ␣
↪A.fill             A.item            A.nbytes          A.ptp           ␣
↪   A.resize          A.size            A.swapaxes        A.trace
A.any             A.base            A.conjugate       A.diagonal        ␣
↪A.flags            A.itemset         A.ndim            A.put           ␣
↪   A.round           A.sort            A.take            A.transpose
A.argmax          A.byteswap        A.copy            A.dot             ␣
↪A.flat             A.itemsize        A.newbyteorder    A.ravel         ␣
↪   A.searchsorted    A.squeeze         A.tobytes         A.var
A.argmin          A.choose          A.ctypes          A.dtype           ␣
↪A.flatten          A.max             A.nonzero         A.real          ␣
↪   A.setfield        A.std             A.tofile          A.view
A.argpartition    A.clip            A.cumprod         A.dump            ␣
↪A.getfield         A.mean            A.partition       A.repeat        ␣
↪   A.setflags        A.strides         A.tolist
```

(continues on next page)

(continued from previous page)

Note the `tolist()` function that we've already discussed above. Here's a few useful ones for our purposes. The `ravel()` function basically gives a flattened view of the array `A` without actually flattening or vectorizing it; so no new memory real estate is being consumed here:

```
In [23]: A.ravel()
Out[23]: array([1, 2, 3, 4, 5, 6])
```

In contrast the `flatten()` command flattens or vectorizes the array `A` and requires new memory space:

```
In [23]: a = A.flatten()

In [27]: a
Out[27]: array([1, 2, 3, 4, 5, 6])
```

This is equivalent to doing:

```
In [29]: A.reshape(1,-1)
Out[29]: array([[1, 2, 3, 4, 5, 6]])
```

What this example does is to reshape the 2×3 array `A` into a row vector. (The argument 1 forces the row, and the second argument -1 says automatically fit everything else in.)

7.6 Plotting with MATPLOTLIB

Let's start with plotting a 2D graph:

- Construct a uniformly-spaced grid of 23 elements, ranging from 0 to 2π. This will be the domain of our function. In the code, we store them as the NumPy array `x`.
- Plot the graph of a cosine function evaluated at these points. This will be stored in memory as the array `y`.
- Don't forget to label your axes!

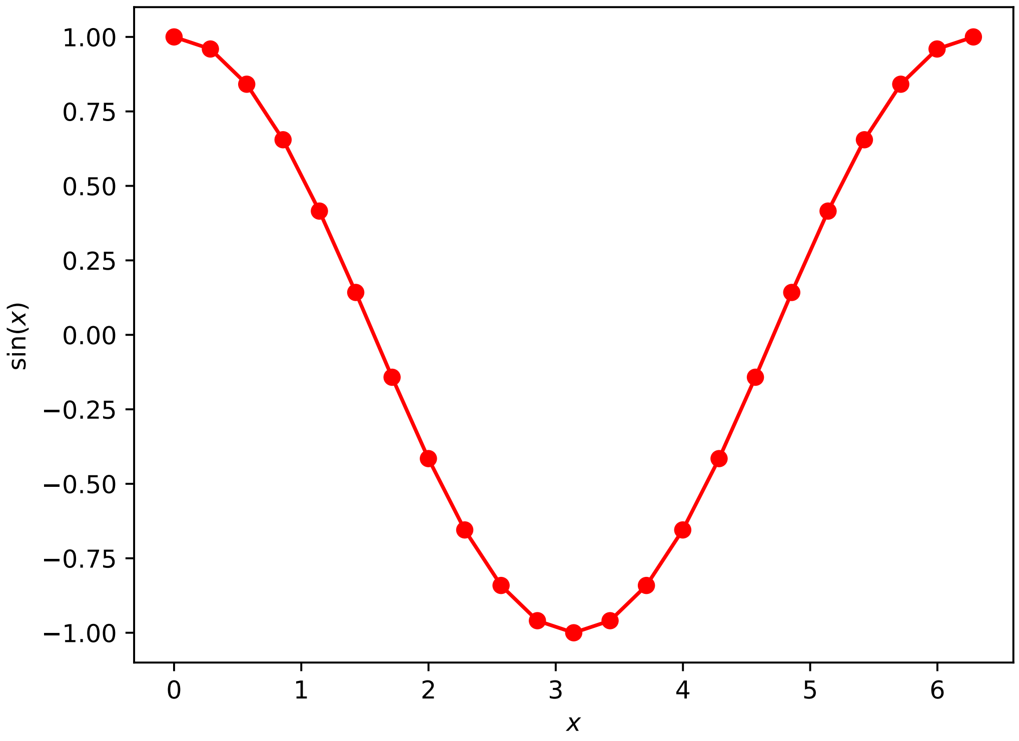
Here we go:

```
import matplotlib.pyplot as plt
import numpy as np
plt.figure()
x = np.linspace(0.0, 2.0*np.pi, 23)
y = np.cos(x)
plt.plot(x,y, 'o-r')
plt.xlabel('$x$')
```

(continues on next page)

(continued from previous page)

```
plt.ylabel('$\sin(x)$')
plt.show()
```



In the next example, we use NumPy to generate a sequence realizations of a Gaussian random variable and then we plot two figures:

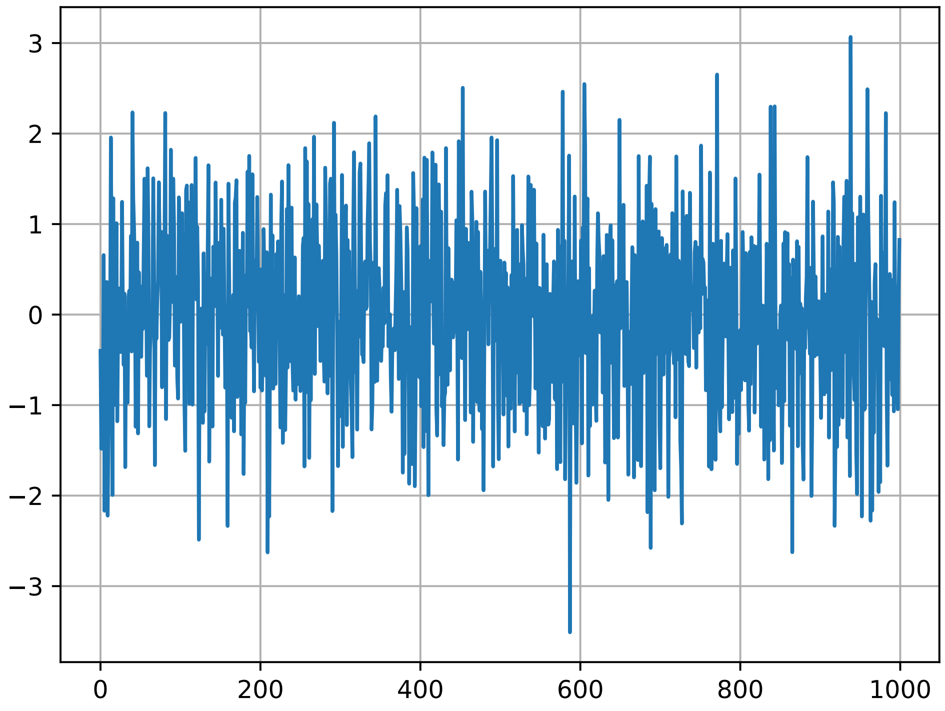
- The sample path.
- A histogram to illustrate the sample distribution.

```
import matplotlib.pyplot as plt
import numpy as np

# Random Gaussian data
x = np.random.randn(1000)

# First figure-sample sequence
plt.figure()
plt.plot(x)
plt.grid()

# Second figure-histogram of sample
plt.figure()
```

(continues on next page)

(continued from previous page)

```
plt.hist( x, 20)
plt.grid()
plt.xlabel('$x$')
plt.title(r'Normal: $\mu=%.2f, \sigma=%.2f$'%(x.mean(), x.
↪std()))

plt.show()
```



This example illustrates the use of the `plt.figure()` function to create a new figure without writing over an existing figure. This is useful in a script context when you are generally multiple plot at the same time.

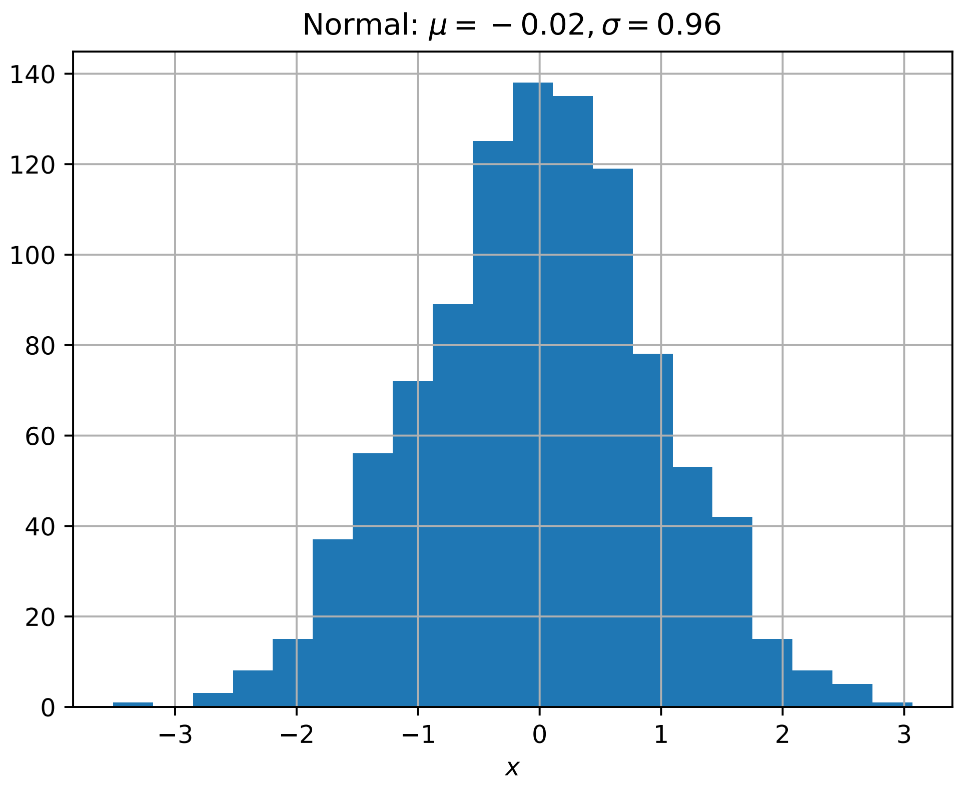
There are lots of additional examples in the MATPLOTLIB website gallery. You can learn from these and adapt them to your own applications later.

7.7 Useful Python and Numpy functions

To create a list of integers counting from 0 to $N-1$:

```
In [1]: N=5
```

(continues on next page)



(continued from previous page)

```
In [2]: range(N)
Out[2]: [0, 1, 2, 3, 4]
```

This is often useful when we do repeated evaluations using loop statements. See below.

7.8 Conditionals

Think about writing down a state contingent decision rule for your robot butler. For example: If the temperature is 18.1 degrees celcius or less, then turn on the heater. If the temperature is 29 degrees celcious or more, then turn on the cooling system. Otherwise, do nothing.

Similar, in talking to Python to execute your commands, quite often your commands will be state contingent. Here an example:

```
# import NumPy library
import numpy as np

# Generate realization of uniform random variable on [0,1]
x = np.random.rand(1)
```

(continues on next page)

(continued from previous page)

```
# Test x condition:
if x < 0.2:
  print("Jump to your left. x = %0.3f" % (x))
else:
  print("Jump to your right. x= %0.3f" % (x))
```

Try this out in your IPython console. See what you get.

Sometimes you may need to issue compound conditional statements. Doing this in Python is quite natural in terms of syntax.

Note: Just be careful of two things:

- **Indents**: Python uses indentations for statements that are nested within conditions.
- **Colons**: after the conditions following IF, ELIF or ELSE statements.

Let's see these is an example:

```
x = 0.214567
test = "apple"
if x <= 0.2 and test != "apple":
  print("Yolo!")
elif x >= 0.2 and test == "apple":
  print("Pine"+test+"pen")
else:
  print("Some of the complementary conditions hold")
```

Can you try deciphering what this says?

7.9 Loops

Python also comes with a useful function called `enumerate()`. This allows you to extract two arguments: First, the integer index for each element of the list we pass to `enumerate()`. Second, the element itself associated with the first argument. To fix ideas, let's look at this example:

```
# Create NumPy array of 15 evenly spaced numbers from .1 to 2.
x = np.linspace(0.1, 2.0, 15)

# Loop over this array and print each element out
for i in range(x.size):
  print(x[i])
```

The `range()` function creates a list of integer indexes for each element of the array `x`, which should count from 0 to 14, in this example.

A FOR loop is useful, as in the example above, when you know exactly when the loop will finish its iterations.

You should get:

```
0.1
0.235714285714
0.371428571429
0.507142857143
0.642857142857
0.778571428571
0.914285714286
1.05
1.18571428571
1.32142857143
1.45714285714
1.59285714286
1.72857142857
1.86428571429
2.0
```

A cooler way to do this is

```
for i, x_i in enumerate(x):
  print("%i \t %0.5f" % (i, x_i))
```

You should get this output:

```
0       0.10000
1       0.23571
2       0.37143
3       0.50714
4       0.64286
5       0.77857
6       0.91429
7       1.05000
8       1.18571
9       1.32143
10      1.45714
11      1.59286
12      1.72857
13      1.86429
14      2.00000
```

The `enumerate()` function gives you two things at once in a loop: the index or counter of the element of an array or list, and, the element itself. Note the fancy arguments in the usage of the `print()` command above? Can you decipher it?

Note: Just be careful of two things:

- **Indents**: Python uses indentations for statements that are nested within FOR or WHILE loops.
- **Colons**: after the conditions following FOR or WHILE statements.

Depending on your application, sometimes you might want to use a WHILE loop instead of a FOR loop. (Usually, this would be the case when you don't know when the loop will terminate.)

Here's an example with a convergent sequence of real numbers generated by the formula:

$$y = \frac{1}{5\,(1+x)}, \qquad x \geq 0$$

```
# Tolerance for zero
TOL = 1e-3
# Arbitrary initialization
x, y = 0.0, 1.0
# The while loop
while y > TOL:
    d = 1.0 + x
    y = 0.2/d
    x = d
    print("\ny = %0.5f" %(y))
```

which outputs:

```
y = 0.2000

y = 0.1000

y = 0.0667

y = 0.0500

y = 0.0400

y = 0.0333

y = 0.0286

y = 0.0250

y = 0.0222

y = 0.0200
```

(continues on next page)

(continued from previous page)

```
y = 0.0182

...

y = 0.0011

y = 0.0010
```

7.10 Statistics

Have a browse through this tutorial here at SciPy Tutorials

7.11 More Python resources for the economist

Since this is a course about macroeconomic modelling, not a course on Python programming, we will be introducing Python and its various usages on the fly. This chapter gave you some essential tools, by example. The only way you will begin to think and speak Python in your economic modelling and application, is to force yourself to put it into action: Code from scratch, don't be afraid to get stuck, and keep trying. We will supply you with basic model templates in Python, but you should not just take a code or library of codes as it is. Take it apart and try re-writing your own version line by line. (Take a peek at our given examples if you like, but try to think by yourself too.) You will appreciate the hard work at the end.

If you want to spend time learning NumPy and SciPy systematically, a great place to do this is at the Scipy Lectures.

The QuantEcon project is a great source to learn Python in the context of economic modelling, or vice-versa. I highly recommend it. It also comes with the same tutorials in the up and coming language Julia.

Don't worry if you don't remember every command or syntax when you use Python, NumPy or SciPy. You can always Google it! (I still do.) Here is a useful cheat sheet for commonly used commands for scientific computing.

For those coming over from the dark side of proprietary software like MATLAB to Python/NumPy/SciPy, check this cheat sheet out.

Finally, learning the syntax of a language is one thing. Actually being able to use the language in your work successfully comes with trying and getting your own hands dirty.

Warning: Keep in mind: If you don't have a good understanding of the theory behind a problem you're trying to compute, and, if you don't think creatively in your solutions approach, then having an elephant's memory of the language's syntax will not get you very far.

BIBLIOGRAPHY

[Fe2014] Fessenden, M. This Computer From 1949 Runs on Water. Link

[Hi1937] Hicks, J. (1937). Mr. Keynes and the "Classics"; A Suggested Interpretation. Econometrica, 5(2), 147-159. https://www.jstor.org/stable/1907242

[HH1989] Hughes Hallett, A. J. (1989). Econometrics and the Theory of Economic Policy: The Tinbergen-Theil Contributions 40 Years On. Oxford Economic Papers, 41(1), new series, 189-214. http://www.jstor.org/stable/2663189

[Ka1935] Kalecki, M. (1935). A Macrodynamic Theory of Business Cycles. Econometrica, 3(3), 327-344. https://www.jstor.org/stable/1905325

[Lu1972] Lucas, R. (1972). Expectations and the neutrality of money, Journal of Economic Theory, 4(2), 103-124. http://dx.doi.org/10.1016/0022-0531(72)90142-1

[Lu1976] Lucas, R. (1976). "Econometric Policy Evaluation: A Critique". In Brunner, K.; Meltzer, A. The Phillips Curve and Labor Markets. Carnegie-Rochester Conference Series on Public Policy. 1. New York: American Elsevier. pp. 19–46

[Ra1928] Ramsey, F. (1928). A Mathematical Theory of Saving. The Economic Journal, 38(152), 543-559. https://www.jstor.org/stable/2224098

[Jo2014] Jones, C. (2014). Macroeconomics, 3rd edition, Norton. http://library.anu.edu.au/record=b2879421

[Lu1972] Lucas, R. (1972). Expectations and the neutrality of money, Journal of Economic Theory, 4(2), 103-124. http://dx.doi.org/10.1016/0022-0531(72)90142-1

[Mi2015] Mishkin, F. (2015). Macroeconomics: Policy and Practice, 2nd edition, Pearson. http://library.anu.edu.au/record=b3578493

[Ka1961] Kaldor, Nicholas, "Capital Accumulation and Economic Growth," in F.A. Lutz and D.C. Hague, eds., The Theory of Capital, St. Martins Press, 1961, pp. 177–222.

[Jo2010] Charles I. Jones, "The New Kaldor Facts: Ideas, Institutions, Population, and Human Capital" (with Paul Romer) American Economic Journal: Macroeconomics, January 2010.