Lecture 1: Introduction and Key Concepts

Lukasz Rachel

ECON 0107, First Term 2022

Course logistics

- **Instructor:** Lukasz Rachel, e-mail: `l.rachel@ucl.ac.uk`
  - Office hours: Tuesdays 11am-12noon
  - via Zoom: `https://ucl.zoom.us/j/99461520125`
- **TA:** Allen Shen, e-mail: `allen.shen.16@ucl.ac.uk`
  - Tutorials on Monday 6pm-7pm and Thursday 9am-11am
  - Office hours: Thursday 4pm-5pm, room G13 PhD Wing Drayton House
- **Course Moodle:**
  `https://moodle.ucl.ac.uk/course/view.php?id=32022`

Readings

- **Textbooks:** Ljungqvist-Sargent, Stokey-Lucas-Prescott, Acemoglu
  - LS: Useful reference for recursive methods and applications.
  - SLP: Useful reference for dynamic programming in discrete time.
  - Acemoglu: Useful reference for continuous-time methods.

Requirements, grading, and prerequisites

Requirements and grading

- 5 problem sets, some with numerical parts. Collaboration encouraged.
  - Hand them in in groups of 3-4 people; Monday 9am.
  - Important for your knowledge building (and for the exam)
- Exam on 9 January (100% of the mark)
  - 24 hours to access the paper. Then 5 hours to submit.

Prerequisites:

- Multivariable calculus. Familiarity with finite dimensional optimization.
- Real analysis, linear algebra, and familiarity with numerical computation software (e.g., MATLAB), useful but not required.

Outline

1. A Bird's Eye View of the Course

2. Preview: A 2-Period Model
   - Euler equations and transversality conditions
   - Dynamic programming and the Principle of Optimality

3. Part I: Discrete time with certainty – Warm-Up

Dynamic optimization & applications

- We will analyze **optimization problems over time**, mostly infinite horizon. Many applications in macro (and elsewhere in economics):
  - Household choosing consumption/savings/portfolio/labor supply.
  - Firm making investment/pricing/inventory decisions.
  - ...
- Some similarities, some differences vs. static optimization
  - At a conceptual level it's the same: same good at different t = different good (Arrow-Debreu eq'm - you'll learn that in Part 2 of this course with Franck Portier)
  - But: flow of time imposes a lot of additional structure suggesting new mathematical techniques & economic intuitions
- **Objective**: how to derive optimality conditions & interpret them

Course structure

Techniques & applications for various kinds of optimization problems:

1. **Derive optimal decision rules**
   a) Uncertainty: deterministic vs. stochastic
   b) Formulation of the problem: discrete-time vs. continuous-time
   c) Solution approach: sequential problem vs. recursive problem
2. **Characterize the implied dynamic behavior of the system**
   a) throughout: both analytical & computational tools
   b) but focus on analytical results, for more on computation see Part 3 of the course (Vincent Sterk)

Course structure (cont.)

1. Deterministic discrete-time optimization
   - Sequential problem (variational approach, Euler equations)
   - Recursive problem (principle of optimality, dynamic programming, Bellman equations)
   - Characterizing deterministic dynamics (linear difference equations)
2. Stochastic discrete-time optimization
   - Sequential & recursive problem
3. Deterministic continuous-time optimization
   - Sequential problem (optimal control, maximum principle)
   - Recursive problem (principle of optimality, HJB equation)
   - Characterizing deterministic dynamics (phase diagrams)
4. Consumption saving problem in partial equilibrium
5. General equilibrium with incomplete markets

Outline

1. A Bird's Eye View of the Course

2. Preview: A 2-Period Model
   - Euler equations and transversality conditions
   - Dynamic programming and the Principle of Optimality

3. Part I: Discrete time with certainty – Warm-Up

Cake-eating problem with two periods

Consider a dynamic problem over two periods:

$$\begin{aligned} V^*(0, a_0) &= \max_{c_0, a_1, c_1, a_2} \log c_0 + \beta \log c_1, \\ \text{s.t.} \quad & c_0 + a_1 = a_0 \\ & c_1 + a_2 = a_1 \\ & c_0, c_1 \geq 0, \\ \text{and} \quad & a_2 \geq 0. \end{aligned}$$

- Consumer with log utility starts at $t = 0$ with assets, $a_0 > 0$.
- Has no income (simplicity). Can borrow/save at rate $r = 0$.
- Cannot leave behind negative assets. Why do we need this?
- Chooses consumption-savings path subject to constraints.

Aside: State variables & control variables

- In this problem: a is a *state*, c is a *control*
- Heuristic definitions:
  - **State**: a given input to a problem. can be either fully exogenous (e.g. productivity) or chosen in the past (e.g. assets)
  - **Control**: choice variable today
- Will formalize those concepts later using recursive methods

Problem can be approached in multiple ways

The problem can be approached in a number of ways. We will consider 2 ways, previewing what is coming in the rest of the course:

1. Eliminate $\{c_0, c_1\}$ and choose $\{a_1, a_2\}$ subject to constraints.
    - This is how we will typically solve the sequential problem in discrete time. It's called the variational approach.
2. Choose a_2 given a_1. Then choose a_1 given a_0 and the optimal decision rule for a_2.
    - This is the recursive approach to optimization.

Outline

1. A Bird's Eye View of the Course

2. Preview: A 2-Period Model
   - Euler equations and transversality conditions
   - Dynamic programming and the Principle of Optimality

3. Part I: Discrete time with certainty – Warm-Up

The first order/variational approach

- One approach is to simplify before attacking the problem.
- We can ignore the constraints, $c_0, c_1 \geq 0$ (since $\lim_{c \to 0} \log c = -\infty$).
- Substitute

$$c_0 = a_0 - a_1 \text{ and } c_1 = a_1 - a_2$$

to eliminate variables and constraints. The problem becomes:

$$\begin{aligned} V^*(0, a_0) &= \max_{a_1, a_2} \log(a_0 - a_1) + \beta \log(a_1 - a_2). \\ \text{and} & \quad a_2 \geq -\overline{D} \text{ (equivalently } -a_2 \leq \overline{D}) \end{aligned}$$

- As before, the problem makes the most (economic) sense when $\overline{D} = 0$. I made this an explicit parameter ("debt limit") to make a point.

The first order/variational approach

- This is a finite dimensional constrained optimization problem.
- The maximum satisfies the first order optimality conditions.
- The problem is concave. Thus, the converse also holds:
  - Any candidate that satisfies appropriate FOCs is also a maximum.
  - If we find a candidate, this also takes care of existence.

$\Longrightarrow$ We can use the first order conditions to characterize the solution.

Preview: second lecture will give analogous "sufficiency" result for infinite-dimensional problem.

The first order conditions

- Assign the Lagrange multiplier $\lambda_2 \geq 0$ to the constraint, $a_2 \geq -\overline{D}$.
- We can then write the Lagrangian as:

$$\mathcal{L} = \log(a_0 - a_1) + \beta \log(a_1 - a_2) + \lambda_2 \left(a_2 + \overline{D}\right).$$

- The FOCs:

$$\begin{aligned} &\frac{d\mathcal{L}}{da_1} = 0 \\ &\frac{d\mathcal{L}}{da_2} = 0, \\ \text{and} \qquad &a_2 \geq -\overline{D}, \lambda_2 \geq 0, \text{ with } \left(a_2 + \overline{D}\right)\lambda_2 = 0. \end{aligned}$$

- The last line is known as a complementary slackness condition.

The intuition for the Lagrange multipliers

- Lagrange multipliers are important in economics (and in this course).
  - They can be given an economic interpretation: The marginal value of relaxing the constraint (in units of the objective function).
  - The more precise meaning will depend on the problem.
- In the above problem, the constraint is $a_2 \geq -\overline{D}$. This is relaxed when we increase the debt limit by one.
- So, in the above problem, $\lambda_2 \geq 0$ is the marginal value of being able to leave behind one more unit of unpaid debt at date 2.
- The complementary slackness says that:
  - If the optimum happens to feature $a_2 > -\overline{D}$, then $\lambda_2 = 0$.
  - Conversely, if $\lambda_2 > 0$, then it needs to be the case that $a_2 = -\overline{D}$.

  (Economic intuition suggests that the second case will apply here.)

Euler equations and transversality conditions

- Let's go back to solving the problem for the case with $\overline{D} = 0$.
- Rewriting the first condition, $\frac{d\mathcal{L}}{da_1} = 0$, we have,

$$-\frac{1}{a_0 - a_1} + \frac{\beta}{a_1 - a_2} = 0. \tag{1}$$

This is an illustration of an **Euler equation**. = interior variations don't improve things

- Rewriting the second condition, $\frac{d\mathcal{L}}{da_2} = 0$, we have,

$$-\frac{\beta}{a_1 - a_2} + \lambda_2 = 0.$$

- This can be combined with complementary slackness to give:

$$\frac{\beta}{a_1 - a_2} a_2 = 0. \tag{2}$$

This is an illustration of **a transversality condition**.

Boundary condition: "don't leave behind valuable stuff."

Solution & comparative statics/dynamics

- Note that we have 2 equations in 2 unknowns, a_1, a_2.
  - This can be solved to obtain,

$$a_2 = 0 \text{ and } a_1 = \frac{\beta}{1+\beta} a_0$$

  - Translating into consumption, we find the optimum

$$c_0 = \frac{1}{1+\beta} a_0 \text{ and } c_1 = \frac{\beta}{1+\beta} a_0.$$

- What is the economic intuition for the solution?
- What happens to the solution as we change the parameter, β?
- We will often be interested in such **comparative statics/dynamics**.

Beyond two periods: Dynamical systems

- If we had $T \geq 2$ periods, as opposed to 2, then Eqs. $(1) - (2)$ would feature T equations in T unknowns.
  - The basic approach is essentially the same (next lecture).
- In the more general case, the solution features a **dynamical system**:
  - In discrete time, a **difference equation** relating a_t, a_{t+1}, a_{t+2}.
  - In continuous time, a **differential equation** with da/dt and d^2a/dt^2.

Outline

1. A Bird's Eye View of the Course

2. Preview: A 2-Period Model
   - Euler equations and transversality conditions
   - Dynamic programming and the Principle of Optimality

3. Part I: Discrete time with certainty – Warm-Up

Alternative approach: Dynamic programming

- Consider the problem once more (after we have done the substitution):

$$\begin{aligned} V^*(0, a_0) &= \max_{a_1, a_2} \log(a_0 - a_1) + \beta \log(a_1 - a_2). \\ \text{and} & \quad a_2 \geq 0 \end{aligned}$$

- What if we optimized as we go along, as opposed to in one shot?
  - This means solving a different problem – the so-called recursive problem, and not the (original) sequential problem.
  - Language: "sequential" problem because you choose the entire sequence

Dynamic programming formulation

- Choose a_2 optimally at date 1, regardless of your past choice:

$$\begin{aligned} V(1, a_1) &= \max_{a_2} \log(a_1 - a_2) \\ \text{and } a_2 &\geq 0. \end{aligned} \tag{3}$$

- Choose a_1 assuming your future choice of a_2 will be optimal:

$$V(0, a_0) = \max_{a_1 > 0} \log(a_0 - a_1) + \beta V(1, a_1). \tag{4}$$

  - Note how I dropped the *: this is strictly speaking a different problem

- This is known as the **recursive or dynamic programming (DP) formulation** of the problem.
  - Can show that if (a_1, a_2) is optimal for the original problem, then a_2 and a_1 are optimal for problems $(3-4)$ (and vice versa).
  - Can also show that $V^*(0, a_0) = V(0, a_0)$ for each $a_0 > 0$.

Principle of Optimality + converse: seq. and rec. formulation are the same

Dynamic programming solution

- So let's follow the principle of optimality and solve the problem backwards:
  - For problem (3), choose $a_2 = 0$ to obtain $V(1, a_1) = \log a_1$.
  - For problem (4), the optimality condition is:

$$\overbrace{-\frac{1}{a_0 - a_1}}^{\text{current cost}} + \overbrace{\beta \frac{dV(1, a_1)}{da_1}}^{\text{future benefit}} = 0.$$

- The same solution as before (check) but with additional interpretation.
  - The marginal value of assets in the future is captured by $\frac{dV(1,a_1)}{da_1}$.
- Note: dynamic programming/the recursive set-up turns the dynamic problem into (many) static ones!

Dynamic programming & recursiveness

- Eq. (4) accounts for the value function in terms of the future value function and the current (optimally determined) flow benefit.
- Mathematically, this makes Eq. (4) a **functional equation (FE)**.
- In this example, not very useful. But if we had infinite horizons, then the FE would have a self-similar (recursive) structure:

$$V(a) = \max_{a^{next}>0} \log\left(a - a^{next}\right) + \beta V\left(a^{next}\right).$$

  - Mathematically, looking for a fixed-point in the space of functions.
  - Hence, the PO provides another line of attack on the problem.
  - Approached with powerful tools: **Contraction mapping theory**.

Benefits of the recursive approach

Why should we use the dynamic programming/recursive approach?

1. **Numerical solution:**
   - Can solve the FE using a computer. May be easier to find function on (compact) domain than finding time path from 0 to ∞.
2. **Comparative statics/dynamics:**
   - Can establish properties of the solution, without actually solving.
3. **Concise and intuitive representation of problem and solution.**
   - This is useful especially when the problem features uncertainty.
   - Don't need to keep track of optimal choice in every possible history.
   - Only keep track of $V(\cdot)$ and behavior in each (payoff-relevant) state.

Taking stock – the roadmap

1. Deterministic discrete-time optimization
   - Sequential problem (variational approach, Euler equations)
   - Recursive problem (principle of optimality, dynamic programming, Bellman equations)
   - Characterizing deterministic dynamics (linear difference equations)
2. Stochastic discrete-time optimization
   - Sequential & recursive problem
3. Deterministic continuous-time optimization
   - Sequential problem (optimal control, maximum principle)
   - Recursive problem (principle of optimality, HJB equation)
   - Characterizing deterministic dynamics (phase diagrams)
4. Consumption saving problem in partial equilibrium
5. General equilibrium with incomplete markets

Outline

1. A Bird's Eye View of the Course

2. Preview: A 2-Period Model
   - Euler equations and transversality conditions
   - Dynamic programming and the Principle of Optimality

3. Part I: Discrete time with certainty – Warm-Up

Consumption-saving in discrete time

- To set the stage, it is useful to set up a canonical optimization problem that we will attempt to solve.
- Consider the infinite-horizon version of the cake-eating problem:

$$\begin{aligned} V^*(0, a_0) \quad &= \quad \max_{\{c_t, a_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t \log c_t, \\ \text{s.t.} \quad & \quad a_{t+1} + c_t = a_t, \\ & \quad c_t \geq 0, \\ & \quad a_{t+1} \geq 0 \text{ for each } t \geq 0 \text{ and } a_0 \text{ given.} \end{aligned}$$

- The key difference is we have many—in fact, infinite—periods.
- We also strengthened $a_2 \geq 0$ to $a_{t+1} \geq 0$ for each t (don't worry about this for now).
- Let's consider a canonical problem that encompasses this...

A canonical problem in discrete time

A canonical optimization problem with infinite periods:

$$\begin{aligned} V^*(0, x_0) &= \sup_{\{x_{t+1}, z_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t \tilde{F}(t, x_t, z_t) \\ \text{s.t.} \quad & z_t \in \tilde{\Gamma}(t, x_t), \\ & x_{t+1} = x_t + \tilde{g}(t, x_t, z_t) \text{ for all } t \geq 0 \text{ and } x_0 \text{ given.} \end{aligned}$$

- Here, x_t, z_t are finite dimensional vectors.
- Here, $x_t \in X$ is the state variable, $z_t \in \tilde{\Gamma}(t, x_t)$ is the control.
  - $\tilde{\Gamma}(t, x_t)$ is a set-valued mapping that captures feasibility.
  - $\tilde{g}(t, x_t, z_t)$ describes the evolution of the state variable.
- $\tilde{F}(t, x_t, z_t)$ is the instantaneous payoff function.
- What are the counterparts of x_t, z_t in the cake-eating special case? What are the counterparts of $\tilde{F}, \tilde{\Gamma}, \tilde{g}$ in the cake-eating case?

Cake-eating problem in canonical notation

- For the cake-eating problem, we have $x_t = a_t$ and $z_t = c_t$.
- The objective function is

$$\tilde{F}(t, a_t, c_t) = \log(c_t).$$

- The state evolution equation, $a_{t+1} = a_t - c_t$ implies

$$\tilde{g}(t, a_t, c_t) = -c_t.$$

- The two constraints, $c_t \geq 0$, $a_{t+1} \geq 0$, are captured by setting,

$$\tilde{\Gamma}(t, a_t) = [0, a_t], \text{ so that } c_t \in \tilde{\Gamma}(t, a_t) = [0, a_t].$$

- The lower bound of the interval captures the constraint $c_t \geq 0$ and the upper bound captures $a_{t+1} \geq 0$.

Substituting out controls

- Recall that, after substituting out consumption, we can also write the cake-eating problem in terms of only the state variable,

$$\begin{aligned} V^*(0, a_0) &= \max_{\{a_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t \log(a_t - a_{t+1}) \\ &\quad \text{s.t. } a_{t+1} \in [0, a_t] \text{ for all } t \geq 0. \end{aligned}$$

- We can typically do this substitution for other problems.
- So we have another canonical problem written in this way...

Canonical problem in terms of states

$$\text{Problem (SP-N):} \qquad V^*(0, x_0) = \max_{\{x_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t F(t, x_t, x_{t+1})$$
$$\text{s.t. } x_{t+1} \in \Gamma(t, x_t) \text{ for all } t \geq 0.$$

- The function, F, describes the payoff in terms of state variables.
- The correspondence, Γ, captures feasibility in terms of state vars.
- This is known as the **sequence problem** since it involves finding a sequence.
  - We call **problem (SP-N)** since it is allowed to be nonstationary.
  - This is the actual problem that we want to solve.

Stationary & recursive canonical problem

- If F and Γ do not depend on time, the problem becomes stationary:

$$V^*(x_0) = \max_{\{x_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t F(x_t, x_{t+1})$$
$$\text{s.t. } x_{t+1} \in \Gamma(x_t) \text{ for all } t \geq 0.$$

  This is **problem (SP)** as defined Chapter 4 of SLP.

  - We will mostly consider stationary environments, so **(SP)** is the main problem that we will want to solve.

- Finally consider the recursive version of the same problem:

$$\text{Find } V(\cdot) \text{ such that } V(x) = \sup_{y \in \Gamma(x)} F(x, y) + \beta V(y).$$

  This is **problem (FE)** (function equation) in Chapter 4 of SLP.

  - **Principle of Optimality & converse**: conditions under which **(SP)** and **(FE)** are the same.

Roadmap for the next couple of lectures

Rest of today: Variational/standard approach.

- Direct attack on problem (SP-N). Applications.
- Useful but does not exploit the recursive structure.

Next week: Dynamic programming approach.

- Establish the "equivalence" of problem (SP) and (FE).
- Indirect attack via problem (FE). Further characterization.