CENTRO DE ESTUDIOS ECONÓMICOS

Maestría en Economía 2024–2026

Macroeconomics 3

Topic 2B : An Introduction to Dynamic Programming

PRESENTED BY: José Daniel Fuentes García

Índice

2B.1 Introduction and Aims

- This course introduces the **fundamentals** of *dynamic programming theory.*
- We will focus exclusively on **deterministic dynamic programming**, assuming *no uncertainty.*
- The material provides a **basic foundation** for solving *macroeconomic models* using deterministic dynamic programming techniques.
- The main objective is to **equip you** with the tools to solve *dynamic optimization problems* using this approach.
- The learning method is primarily **hands-on**, emphasizing *learning by doing.*
- You will explore a **variety of solution strategies** applicable to solving *recursive problems.*
- Keep in mind that **modern macroeconomic dynamic programming problems** often cannot be solved analytically and typically require *numerical methods* and **software tools** like MATLAB—topics usually covered in graduate school.

Suggested Reading

- **The material of this class is based on:**
  Adda and Cooper (2003), Dynamic Economics: Quantitative Methods & Applications, MIT Press, Chapter 2.
- **There are other texts that also briefly cover dynamic programming which you may find useful. For example:**
  Ljungqvist and Sargent (2004), Recursive Macroeconomic Theory, MIT Press (2nd Edition), Chapters 1 and 3.
- **For a PhD level coverage of the material we cover see:**
  Stokey and Lucas (1989), Recursive Methods in Economic Dynamics, Harvard University Press, Chapter 2.1 and Chapter 4.

Direct Method / Sequence Approach

1. Euler equation & other FOCs
2. Lifetime budget constraint + TVC imposed

Dynamic Programming

A. Euler Equation Method – Here you don't need to know the form of the value function, but it has to be differentiable. Same procedure as the Direct Method

B. Solve the Bellman Equation – Here the *value function does not need to be differentiable*, and you find the form of the value function. Either:

a) Value function iteration method
b) Guess & verify method
c) Policy function iteration method

2B.2 An Overview of Dynamic Programming

- At its core, the dynamic programming approach builds on the concept of *indirect utility*, which you may recognize from standard consumer theory.
- This method allows us to compactly represent the value of being in a certain situation or **state**.
- Dynamic programming extends this concept to environments that evolve over time.
- To connect with earlier knowledge, recall how we express indirect utility in static consumer theory.
- In that framework, consumers solve:

$$V(I,p) = \max_c u(c),$$

- subject to the constraint $pc = I$
- where c is a vector of consumption goods, p is a vector of prices, and I is income.

Maximization problem:

$$\max_c u(c) \quad \text{s.t.} \quad pc = I$$

Lagrangian:

$$\mathcal{L}(c,\lambda) = u(c) + \lambda(I - pc)$$

First-order condition:

$$\frac{\partial \mathcal{L}}{\partial c_j} = u_j(c) - \lambda p_j = 0 \Rightarrow \frac{u_j(c)}{p_j} = \lambda$$

Envelope theorem:

$$\frac{\partial V(I,p)}{\partial I} = \lambda \Rightarrow \lambda = V_I(I,p)$$

Substitute:

$$\frac{u_j(c)}{p_j} = V_I(I,p)$$

- The first-order condition is:

$$\frac{u_j(c)}{p_j} = \lambda$$

- for all j, where λ is the multiplier on the budget constraint and $u_j(c)$ is the marginal utility from good j
- Here $V(I, p)$ is an indirect utility function: it gives you the maximized level of utility from the current state (I, p)
- If the consumer is in this state, you can predict that they will attain this level of utility
- You do not need to know what the consumer will do with their income: it is enough to know that they will act optimally
- This is very powerful logic and underlies the idea behind dynamic programming
- To illustrate, what happens if we give the consumer some more income?
- Well, welfare goes up by $V_I(I, p) > 0$. Can I predict what will happen when I give the consumer some more income?
- Not really, since the optimizing consumer is indifferent with respect to how this extra income is spent:

$$\frac{u_j(c)}{p_j} = V_I(I, p)$$

- for all j
- The indirect utility function helps summarize the solution to the consumer's optimization problem and reveals the marginal value of income, even without detailed knowledge of their consumption behavior.
- Consider a basic example with two goods, where the consumer maximizes:

$$\max_{c_1, c_2} [a \log c_1 + (1-a) \log c_2]$$

subject to the budget constraint:

$$p_1 c_1 + p_2 c_2 = I$$

- The first-order conditions for this problem are:

$$p_1 c_1 = aI, \quad p_2 c_2 = (1-a)I$$

- Solving for c_1 and c_2, and plugging them into the utility function, gives us the indirect utility:

$$V(I, p_1, p_2) = a \log\left(\frac{a}{p_1}\right) + (1-a)\log\left(\frac{1-a}{p_2}\right) + \log I$$

- This shows how utility depends on the state variables I, p_1, and p_2.
- *In dynamic programming terminology, this is called the value function*, because it maps the state variables to the maximum attainable value.

Full procedure

Maximization problem:

$$\max_{c_1,c_2} [a \log c_1 + (1-a)\log c_2] \quad \text{s.t.} \quad p_1c_1 + p_2c_2 = I$$

Lagrangian:

$$\mathcal{L}(c_1, c_2, \lambda) = a\log c_1 + (1-a)\log c_2 + \lambda(I - p_1c_1 - p_2c_2)$$

First-order conditions:

$$\frac{\partial \mathcal{L}}{\partial c_1} : \quad \frac{a}{c_1} - \lambda p_1 = 0 \Rightarrow p_1c_1 = aI$$
$$\frac{\partial \mathcal{L}}{\partial c_2} : \quad \frac{1-a}{c_2} - \lambda p_2 = 0 \Rightarrow p_2c_2 = (1-a)I$$

Solve for c_1 **and** c_2:

$$c_1 = \frac{aI}{p_1}, \quad c_2 = \frac{(1-a)I}{p_2}$$

Substitute into utility:

$$V(I, p_1, p_2) = a\log\left(\frac{aI}{p_1}\right) + (1-a)\log\left(\frac{(1-a)I}{p_2}\right)$$

Simplify:

$$V(I, p_1, p_2) = a\log\left(\frac{a}{p_1}\right) + (1-a)\log\left(\frac{1-a}{p_2}\right) + \log I$$

2B.3 Finite-Horizon Dynamic Programming

- We now study a basic dynamic optimization scenario: the classic *cake-eating problem.*
- We begin by analyzing the case with a finite time horizon before moving to the infinite horizon case.
- Suppose you are initially endowed with a cake of size W_1.
- At every time period $t = 1, 2, 3, \ldots, T$, you can consume a portion of the cake, but must save the remaining amount for future periods.
- Let c_t denote consumption in period t, and let $u(c_t)$ represent the instantaneous utility derived from consuming c_t.

- The utility function is assumed to be time-invariant — this reflects stationary preferences.
- Assume that $u(\cdot)$ is strictly increasing and strictly concave. Additionally, it satisfies the Inada condition:
$$\lim_{c \to 0} u'(c) = \infty$$
- The Inada condition ensures that consumption in every period is strictly positive — we always consume at least a small amount of cake.
- Therefore, the agent's lifetime utility over T periods is:
$$\sum_{t=1}^{T} \beta^{(t-1)} u(c_t)$$
- where the discount factor β lies in the interval $0 < \beta < 1$.
- Assume that the cake does *not depreciate* — it neither spoils nor grows over time. This means total cake only changes when the agent consumes it.
- The evolution of the cake across periods is described by the following **transition equation (or law of motion)**:
$$W_{t+1} = W_t - c_t \tag{1}$$
where W_t is the remaining cake (state), and c_t is the amount consumed in period t (control).
- This relation holds for each period $t = 1, 2, 3, \ldots, T$. It shows how today's choice affects tomorrow's available resources.
- Therefore, the decision problem is: how much of the cake should be eaten in each period to maximize total lifetime utility?
- Formally, how can we determine the optimal consumption path $\{c_t\}_{t=1}^{T} = (c_1, c_2, \ldots, c_T)$? That is, the best way to spread the cake over time.
- There are two main approaches to solving this kind of optimization problem:
  1. **The Direct Method (also called the Sequence Approach)**
     This involves solving the problem by substituting constraints into the utility function and directly optimizing.
  2. **Dynamic Programming**
     This uses recursive logic, breaking the problem into smaller stages and solving it backward in time.

Direct Method / Sequence Approach

- One way to solve this kind of constrained optimization is to directly work with all constraints at once — this is called the *sequence approach*, a term popularized by Stokey and Lucas (1989).

- This is the method we've been using throughout the course so far.

- The setup of the optimization problem is:

$$\max_{\{c_t\}_{t=1}^{T},\{W_{t+1}\}_{t=2}^{T+1}} \sum_{t=1}^{T} \beta^{(t-1)} u(c_t), \tag{2}$$

- subject to the transition equation from before:

$$W_{t+1} = W_t - c_t \quad \text{for all } t = 1, 2, \ldots, T$$

- Additionally, we impose non-negativity constraints on both consumption and remaining cake:

$$c_t \geq 0, \quad W_t \geq 0$$

  and the initial cake size W_1 is given.

- Alternatively, we can rewrite the full set of period-by-period constraints into a single **intertemporal budget constraint** by summing:

$$\sum_{t=1}^{T} c_t + \sum_{t=1}^{T} W_{t+1} = \sum_{t=1}^{T} W_t$$

- Which simplifies to:

$$\sum_{t=1}^{T} c_t + W_{T+1} = W_1 \tag{3}$$

- This final expression shows the total resources available (initial cake) equal the total consumed plus the cake leftover at the end.

- In this setup, the non-negativity constraints are straightforward:

$$c_t \geq 0 \quad \text{for } t = 1, 2, \ldots, T, \quad \text{and} \quad W_{T+1} \geq 0$$

- From equation (3), we know that the sum of consumption over all periods plus the final leftover cake must equal the initial amount:

$$\sum_{t=1}^{T} c_t + W_{T+1} = W_1$$

- Let λ be the multiplier on this constraint. The first-order conditions are:

$$\beta^{t-1}u'(c_t) = \lambda \quad \text{for } t = 1, 2, \dots, T$$

$$\lambda = \phi$$

where ϕ is the multiplier on the constraint $W_{T+1} \geq 0$.

- The non-negativity constraint $c_t \geq 0$ is not active here because we assume that the utility function satisfies the Inada condition — i.e., marginal utility goes to infinity as consumption approaches zero:

$$\lim_{c \to 0} u'(c) = \infty$$

- So in practice, the agent always consumes something in each period.

- For any two periods, the FOCs imply the Euler condition. For example:
    - In period 1: $\beta^0 u'(c_1) = u'(c_1) = \lambda$
    - In period 2: $\beta^1 u'(c_2) = \beta u'(c_2) = \lambda$

- Eliminating λ gives:

$$u'(c_t) = \beta u'(c_{t+1})$$

- This equation characterizes the optimal intertemporal allocation — it balances marginal utility across periods, discounted by β.

- Final result (Euler equation):

$$\boxed{u'(c_t) = \beta u'(c_{t+1})} \tag{4}$$

- The Euler equation is a **necessary condition** for any t: if it's violated, the agent can improve utility by adjusting c_t and c_{t+1}.

- To recall its meaning: suppose we have a candidate solution for this problem given by $\{c_t^*\}_{t=1}^{T}, \{W_t^*\}_{t=2}^{T+1}$.

- The Euler equation tells us that the marginal utility cost of reducing consumption in period t equals the discounted marginal utility gain from consuming that saved amount in period $t+1$, via factor β.

- *But what if the deviation spans more than one period?* That is, what if the agent reduces consumption today and delays re-spending for more than one period?

- Example: suppose consumption is reduced by an amount x in period t, saved for two periods, and then spent in $t+2$. Can this increase utility?

- This type of deviation isn't directly ruled out by the one-period Euler equation (4).

- However, we can chain together two Euler conditions:

$$u'(c_t) = \beta u'(c_{t+1})$$

$$u'(c_{t+1}) = \beta u'(c_{t+2})$$

- Combining them yields:

$$u'(c_t) = \beta^2 u'(c_{t+2})$$

- This shows that even across two periods, marginal utility is optimally balanced, and no such multi-period deviation will raise utility — the candidate solution remains valid.

- Since the problem is finite, the fact that the Euler equation holds between all adjacent periods implies that any local deviation from a candidate solution won't improve utility — as long as the Euler equation is satisfied.

- But this condition is only *necessary*, not sufficient. A solution can satisfy the Euler equation but still be suboptimal.

- The true optimal solution satisfies the Euler equation in every period *up to the point where the agent has fully consumed the cake.*

- Therefore, in an optimal plan, the non-negativity constraint on W_{T+1} must bind.

- Without this constraint, the agent would choose $W_{T+1} = -\infty$, which implies they consume more than the total cake — effectively dying with debt (or owing cake!).

- So, for optimality, this constraint must be binding to ensure that no cake is left unused at the end.

- The problem is fully characterized by:
  - The initial condition W_1 (the starting cake size), and
  - The terminal condition $W_{T+1} = 0$ (no cake left).

- Together with the $T-1$ Euler equations and the intertemporal budget constraint (3), we determine the optimal consumption path.

- Let the value of this optimal solution be denoted:

$$V_T(W_1)$$

  where T is the number of periods and W_1 is the initial cake size.

- $V_T(W_1)$ represents the maximum utility attainable in a T-period problem, starting with a cake of size W_1.

- From this point forward, we refer to this as the **value function**, directly analogous to the household indirect utility functions discussed earlier.

- The value function is defined as:

$$V_T(W_1) = \max_{c_t} \sum_{t=1}^{T} \beta^{t-1} u(c_t) + \lambda \left[W_1 - \sum_{t=1}^{T} c_t - W_{T+1} \right] + \phi W_{T+1}$$

- This formulation includes the intertemporal resource constraint and the terminal condition.
- As with indirect utility, a marginal increase in W_1 leads to a change in lifetime utility equal to the marginal utility of consumption — regardless of when that cake is consumed.
- Mathematically:

$$\frac{dV(W_1)}{dW_1} = V_T'(W_1) = \lambda = \beta^{t-1} u'(c_t)$$

- This holds for any $t = 1, 2, \ldots, T$
- That is, *the timing of consuming the extra cake doesn't affect utility*, as long as the consumer is optimizing — utility adjusts through the Euler condition.

Summary of the direct / sequential approach

- A unique solution exists under the following conditions:
  1. The utility function is strictly increasing: $u'(\cdot) > 0$
  2. The utility function is strictly concave: $u''(\cdot) < 0$
  3. The constraint set is bounded (i.e., it rules out infinite values), ensuring a well-defined and finite choice set
- Additionally, the **Inada condition** ensures a unique *interior* solution by avoiding corner cases:

$$\lim_{c \to 0} u'(c) = \infty$$

- The resulting unique optimal consumption path $\{c_t^*\}_{t=1}^T = (c_1^*, c_2^*, \ldots, c_T^*)$ is characterized by:
  1. The Euler equation
  2. The intertemporal budget constraint
  3. The boundary conditions:

$$W_{T+1} = 0 \quad \text{and} \quad W_1 > 0$$

Dynamic Programming Approach

- This method uses a **recursive formulation** — solving a dynamic problem by breaking it into smaller subproblems.
- When the agent faces the same decision structure in every period, the T-period problem can be simplified to a 2-period recursive problem.
- This approach relies on the *value function* derived from solving a shorter-horizon version of the problem.
- Given W_0, we consider the problem:

$$V_T(W_0) = \max_{c_0} u(c_0) + \beta V_T(W_1) \tag{5}$$

where

$$W_1 = W_0 - c_0$$

- Here, consumption in period 0 (c_0) determines the remaining cake W_1 that will be available from period 1 onward.
- Rather than selecting a full sequence of consumption decisions, the agent only needs to choose c_0, and the rest of the behavior is embedded recursively in the value function $V_T(W_1)$.
- Once c_0 is chosen (and hence W_1 is known), the entire value of the problem is given by $V_T(W_1)$.
- In this way, the value function completely encapsulates optimal decision-making from period 1 forward.
- For the dynamic programming problem, the timing of future consumption after the initial period is irrelevant.
- What matters is that the agent behaves optimally starting from period 1, so utility is captured by the value function $V_T(W_1)$.
- This principle is called the **principle of optimality** — the agent doesn't need to know all future values of c_t to decide on c_0; she only needs to know that future choices will be made optimally.
- Given this, a rational decision on c_0 can be made using just $V_T(W_1)$, assuming optimal behavior from period 1 onward.
- The first-order condition, assuming differentiability of $V_T(W_1)$, is obtained by taking the derivative with respect to the control variable c_0:

$$V_T(W_0) = u(c_0) + \beta V_T(W_1) = u(c_0) + \beta V_T(W_0 - c_0)$$

$$\Rightarrow \quad u'(c_0) = \beta V_T'(W_1)$$

- Since $W_1 = W_0 - c_0$, the chain rule implies:

$$V_T'(W_1) = V_T'(W_0 - c_0)$$

- The marginal benefit of consuming less in period 0 is reflected in the derivative of the value function with respect to W_1.
- As discussed earlier in the sequence (direct) approach, we already established:

$$V_T'(W_1) = \beta^{t-1}u'(c_t) = u'(c_1) \tag{6}$$

- Therefore, we conclude:

$$u'(c_0) = \beta u'(c_1)$$

- This is the familiar Euler equation.
- Once $V_T'(W_1)$ is known, the entire optimal consumption path can be derived using equation (6).
- Of course, the simplicity here is somewhat deceptive — this step relied on knowing $V_T(W_1)$ from solving the direct approach first.
- In practical application, with a finite horizon, we solve this recursively by working backward from period T.

Full procedure

$$V_T(W_0) = u(c_0) + \beta V_T(W_1)$$

$$W_1 = W_0 - c_0$$

$$\frac{dV_T(W_0)}{dc_0} = u'(c_0) - \beta V_T'(W_1) = 0$$

$$\Rightarrow u'(c_0) = \beta V_T'(W_1)$$

$$\text{From the direct approach:} \quad V_T'(W_1) = u'(c_1)$$

$$\Rightarrow u'(c_0) = \beta u'(c_1)$$

EXAMPLE

- Consider the utility function $u(c) = \log(c)$, and assume the agent lives only for $T = 2$ periods.

- We solve this cake-eating problem first using the **direct approach**.
- Lifetime utility:

$$U = \log(c_1) + \beta \log(c_2)$$

- Budget constraint (with $W_3 = 0$):

$$c_1 + c_2 = W_1$$

- First-order conditions:

$$\frac{1}{c_1} = \lambda, \quad \frac{\beta}{c_2} = \lambda, \quad c_1 + c_2 = W_1 \tag{7}$$

- Euler equation:

$$\frac{1}{c_1} = \frac{\beta}{c_2} \tag{8}$$

- Solving (7) and (8):

$$c_1 = \frac{W_1}{1+\beta}, \quad c_2 = \frac{\beta W_1}{1+\beta}$$

- Value function:

$$V_T(W_1) = \log\left(\frac{W_1}{1+\beta}\right) + \beta \log\left(\frac{\beta W_1}{1+\beta}\right)$$

- Now solve using **dynamic programming**. At $T = 2$, the terminal condition is:

$$W_3 = 0 = W_2 - c_2 \Rightarrow V_2(W_2) = \log(W_2)$$

- At $T = 1$, the value function becomes:

$$V_1(W_1) = \max_{W_2} \log(W_1 - W_2) + \beta \log(W_2)$$

- First-order condition:

$$\frac{1}{W_1 - W_2} = \frac{\beta}{W_2}$$

- Rearranging:

$$W_2 = \frac{\beta W_1}{1+\beta} \Rightarrow c_1 = W_1 - W_2 = \frac{W_1}{1+\beta}$$

- Final value function:

$$V_1(W_1) = \log\left(\frac{W_1}{1+\beta}\right) + \beta \log\left(\frac{\beta W_1}{1+\beta}\right)$$

- The value function $V_1(W_1)$ represents the **maximum utility** the agent can attain starting with cake size W_1.

- It captures the **optimal consumption allocation** between periods 1 and 2, given logarithmic preferences and discount factor β.
- Its analytical form shows how utility depends on **initial resources** and the agent's **degree of impatience**.

Full procedure

$$\max_{c_1,c_2} \log(c_1) + \beta \log(c_2) \quad \text{s.t.} \quad c_1 + c_2 = W_1$$

Lagrangian:

$$\mathcal{L} = \log(c_1) + \beta \log(c_2) - \lambda(c_1 + c_2 - W_1)$$

F.O.C.:

$$\frac{1}{c_1} = \lambda, \quad \frac{\beta}{c_2} = \lambda \Rightarrow \frac{1}{c_1} = \frac{\beta}{c_2}$$

Budget:

$$c_1 + c_2 = W_1$$

Solve:

$$c_1 = \frac{W_1}{1+\beta}, \quad c_2 = \frac{\beta W_1}{1+\beta}$$

Value function:

$$V_T(W_1) = \log(c_1) + \beta \log(c_2) = \log\left(\frac{W_1}{1+\beta}\right) + \beta \log\left(\frac{\beta W_1}{1+\beta}\right)$$

Dynamic programming:

$$V_2(W_2) = \log(W_2), \quad W_3 = 0$$

$$V_1(W_1) = \max_{W_2} \log(W_1 - W_2) + \beta \log(W_2)$$

F.O.C.:

$$\frac{1}{W_1 - W_2} = \frac{\beta}{W_2} \Rightarrow W_2 = \frac{\beta W_1}{1+\beta}$$

$$c_1 = W_1 - W_2 = \frac{W_1}{1+\beta}$$

Value function:

$$V_1(W_1) = \log\left(\frac{W_1}{1+\beta}\right) + \beta \log\left(\frac{\beta W_1}{1+\beta}\right)$$

2B.4 Infinite-Horizon Dynamic Programming

- In macroeconomics, we often analyze problems over an **infinite horizon**, where $T = \infty$.
- For the cake-eating problem, the infinite-horizon version involves solving:

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

- subject to the resource constraint:

$$W_{t+1} = W_t - c_t$$

- with the conditions:

$$c_t \geq 0 \quad \text{and} \quad W_1 \text{ is given.}$$

- To ensure the objective function remains bounded (i.e., utility does not go to infinity), we typically require:

$$0 < \beta < 1$$

  which imposes discounting over time and ensures convergence of the utility sum.

- To formulate the dynamic programming problem, we write:

$$V(W_t) = \max_{c_t, W_{t+1}} u(c_t) + \beta V(W_{t+1}) \tag{9}$$

- This equation reflects the fact that when $T = \infty$, the agent faces the same optimization problem each period — only the cake size W_t changes over time.
- The function $V(W_t)$ represents the value of the infinite-horizon problem starting with cake of size W_t.
- In each period, the agent chooses c_t, and the cake evolves according to:

$$W_{t+1} = W_t - c_t$$

- The value of entering the next period with cake W_{t+1} is given by $V(W_{t+1})$, discounted by $\beta \in (0, 1)$.
- For convenience, we could rewrite (9) more abstractly as:

$$V(W) = \max_{c, W'} u(c) + \beta V(W')$$

- In this stationary infinite-horizon setup, we drop time subscripts on $V(\cdot)$ or W, since the structure is the same in every period. The prime (W') simply denotes the next period's state.

- In dynamic programming problems, we need to distinguish between **state variables** and **control variables**.
- In this problem, the state variable is the size of the cake W_t, given at the start of any period.
- **A state variable** fully captures all past relevant information needed for the current optimization. (In more complex setups, we may need to distinguish between controlled and exogenous state variables.)
- We don't care how we arrived at the current cake size W_t; what matters is how much cake is available and how we will allocate it going forward.
- **A control variable** is the decision variable — here, it's the consumption c_t and consequently the leftover cake W_{t+1}.
- The relationship between today's state and control and tomorrow's state is:

$$W_{t+1} = W_t - c_t,$$

  which is called the **transition equation** or **law of motion**.
- *Intuition:* We are learning to formally split what the agent controls (consumption) from what evolves (cake), allowing us to track and optimize decisions across time.
- There are multiple equivalent ways to represent the cake-eating problem.

1. Starting from equation (9), we can eliminate the transition equation and write:

$$V(W_t) = \max_{c_t \in [0, W_t]} \; u(c_t) + \beta V(W_t - c_t) \tag{10}$$

- In this version, the only control variable is current consumption c_t.
- *Intuition:* This form emphasizes the direct trade-off between consuming today and leaving less for the future.

2. Alternatively, we can reformulate the problem by making tomorrow's state the choice variable:

$$V(W_t) = \max_{W_{t+1} \in [0, W_t]} \; u(W_t - W_{t+1}) + \beta V(W_{t+1}) \tag{11}$$

- In this case, the control variable is the amount of cake left for the next period W_{t+1}.
- *Intuition:* This form highlights how the agent directly chooses how much cake to save, which often simplifies algebra and makes recursive analysis more straightforward.
- Either specification (10) or (11) produces the same results, but (11) is often easier to work with, so we will focus on it.

3. For more complicated problems, it is often easier to solve by using a **Lagrangian formulation** instead of substituting directly into the transition equation:

$$V(W_t) = \max_{c_t, W_{t+1}} \Big\{ u(c_t) + \beta V(W_{t+1}) \Big\} + \lambda_t \Big[W_t - c_t - W_{t+1} \Big],$$

- Here, both c_t and W_{t+1} are treated as control variables.
- The expression in (11) is called a **functional equation** or a **Bellman equation**.
- The unknown in the Bellman equation is the value function itself — the objective is to find a function $V(W)$ that satisfies the condition for all W.
- Unlike the finite-horizon setup, in the infinite horizon there is no terminal period, so we rely on the recursive structure of the value function.
- In general, proving the existence of such a function $V(W)$ is nontrivial (it usually requires the advanced material of Stokey and Lucas (1989)).
- For this course, we do not attempt to prove existence.
- Instead, we assume that such a solution exists, allowing us to explore its properties.
- **Note:** Since $V(W)$ appears on both sides of the Bellman equation, it acts as a fixed-point problem. This fixed-point property provides a means of solving the equation.
- Thanks to stationarity, we can make meaningful arguments about the existence of a value function that satisfies the functional equation.
- *Intuition:* The Bellman equation is like a consistency check — it requires that today's best choice, combined with tomorrow's best choices, reproduces the same value function everywhere. It ensures the recursive logic holds.

Envelope Theorem

- We can rewrite the dynamic programming problem as:

$$V(W_t) = \max_{W_{t+1} \in [0, W_t]} u(W_t - W_{t+1}) + \beta V(W_{t+1}) \tag{12}$$

- The first-order condition for (12) is:

$$u'(W_t - W_{t+1}) = \beta V'(W_{t+1})$$

- Equivalently, using $c_t = W_t - W_{t+1}$:

$$u'(c_t) = \beta V'(W_{t+1})$$

- The difficulty arises with the term $V'(W_{t+1})$: it involves the derivative of the value function, which is unknown.

- Thus, a key objective is to find a way to eliminate $V'(W_{t+1})$ from the first-order condition.
- *Intuition:* The envelope theorem allows us to connect the marginal value of resources today directly to consumption, avoiding the need to explicitly differentiate the unknown value function.

Full procedure of FOC (12)

$$\max_{W_{t+1}\in[0,W_t]} \quad u(W_t - W_{t+1}) + \beta V(W_{t+1}) \qquad \text{(Eq. 12)}$$

$$\frac{d}{dW_{t+1}}\Big[u(W_t - W_{t+1}) + \beta V(W_{t+1})\Big] = 0$$

$$-u'(W_t - W_{t+1}) \; + \; \beta V'(W_{t+1}) \; = \; 0$$

$$\boxed{u'(W_t - W_{t+1}) \; = \; \beta V'(W_{t+1})}$$

- Suppose a solution exists such that the state variable evolves according to:

$$W_{t+1} = g(W_t).$$

- The Bellman equation (12) can then be written as:

$$V(W_t) = u(W_t - g(W_t)) + \beta V(g(W_t)).$$

- Totally differentiating with respect to W_t (where everything is treated as a function of W_t):

$$V'(W_t) = u'(W_t - g(W_t)) - u'(W_t - g(W_t))g'(W_t) + \beta V'(g(W_t))g'(W_t)$$

- Rearranging:

$$V'(W_t) = u'(W_t - W_{t+1}) - \Big[u'(W_t - W_{t+1}) - \beta V'(W_{t+1})\Big]g'(W_t)$$

- From the first-order condition, the term in brackets is zero, so we have:

$$V'(W_t) = u'(W_t - W_{t+1}).$$

- Moving forward one period gives:

$$V'(W_{t+1}) = u'(W_{t+1} - W_{t+2}).$$

- *Intuition:* The envelope theorem allows us to replace the derivative of the unknown value function with the marginal utility of current consumption, which is directly observable.

- From before, the envelope condition is:

$$V'(W_{t+1}) = u'(W_{t+1} - W_{t+2}) \tag{13}$$

- Using (13), the first-order condition can be written as:

$$u'(c_t) = \beta V'(W_{t+1}) = \beta u'(W_{t+1} - W_{t+2}) = \beta u'(c_{t+1})$$
$$\Rightarrow u'(c_t) = \beta u'(c_{t+1}) \tag{14}$$

- Equation (14) is just the standard **Euler equation**.
- In general, the envelope condition is obtained by differentiating the problem with respect to each state variable. For this case:

$$V(W_t) = \max_{W_{t+1}\in[0,W_t]} u(W_t - W_{t+1}) + \beta V(W_{t+1})$$

- Differentiating both sides of the Bellman equation with respect to W_t gives:

$$V'(W_t) = u'(W_t - W_{t+1})$$

  (this is sometimes referred to as the Benveniste-Scheinkman condition).
- Shifting the envelope condition forward by one period gives equation (13), which when substituted back leads directly to the Euler equation (14).
- *Intuition:* The envelope theorem acts as a shortcut — instead of differentiating the full optimization problem, it tells us that the marginal value of resources today equals the marginal utility of consuming them, which then delivers the Euler condition in a very clean way.

Full procedure

$$\text{Bellman:} \quad V(W_t) = \max_{W_{t+1}\in[0,W_t]} u(W_t - W_{t+1}) + \beta V(W_{t+1})$$

$$\text{FOC:} \quad -u'(W_t - W_{t+1}) + \beta V'(W_{t+1}) = 0 \;\Rightarrow\; u'(W_t - W_{t+1}) = \beta V'(W_{t+1})$$

$$\text{Envelope:} \quad V'(W_t) = u'(W_t - W_{t+1})$$

$$\text{Shift:} \quad V'(W_{t+1}) = u'(W_{t+1} - W_{t+2})$$

$$\text{Substitute:} \quad u'(W_t - W_{t+1}) = \beta\, u'(W_{t+1} - W_{t+2})$$

$$\text{Euler:} \quad u'(c_t) = \beta\, u'(c_{t+1}), \qquad c_t = W_t - W_{t+1},\; c_{t+1} = W_{t+1} - W_{t+2}$$

- The solution to the cake-eating problem satisfies the Euler equation (the necessary condition) for all W.
- The link between consumption C_t and the next period's cake W_{t+1} (from the different formulations) to the size of the cake (state W_t) is expressed through the policy functions:

$$C_t = \mu(W_t)$$

$$W_{t+1} = \gamma(W_t) \equiv W_t - \mu(W_t)$$

- These policy functions are **time-consistent**, meaning the optimal plan today remains optimal in the future — there is no incentive to deviate.
- In applied research, policy functions are crucial because they provide a direct mapping from states (resources available) to actions (consumption choices).
- The main objective is to solve for the value function and, from it, derive the associated policy functions.
- Often, closed-form solutions are not available. In such cases, we rely on **numerical methods** to approximate the value and policy functions.
- *Intuition:* The value function tells us how good it is to have a certain amount of cake, while the policy function tells us exactly how much to eat — turning theory into actionable rules.

2B.5 Solution Techniques

- There are several different methods available for solving infinite-horizon dynamic programming problems.
- The aim of these techniques is to compute either the **value function**, the **policy functions**, or both.
- The four solution techniques we will consider are:
  1. Value function iteration
  2. Guess and verify method
  3. Policy function iteration
  4. Euler equation method
- The first three methods all focus on solving the functional or Bellman equation directly. The Euler equation method is different: it bypasses the Bellman equation by working directly with the necessary conditions.
- *Intuition:* Each of these techniques provides a different "toolbox" to tackle the recursive problem. Some are better for theory (guess and verify), others for computation (value or policy iteration), while the Euler approach works by exploiting first-order conditions.

Value function iteration

- The first method constructs a sequence of value functions and associated policy functions.
- The idea is to iterate on the Bellman equation, starting from an arbitrary initial guess (e.g., $V_0 = 0$), and continue until the sequence V_j converges.
- Formally, the iterative process is given by:

$$V_{j+1}(W_t) = \max_{c_t, W_{t+1}} \; u(W_t, c_t) + \beta V_j(W_{t+1}),$$

- subject to the transition equation:

$$W_{t+1} = g(W_t, c_t)$$

- Here, W_t is the state variable, and we can equivalently express the problem in terms of the control variable W_{t+1} or c_t.
- *Intuition:* Value function iteration works like "guessing and updating" — we start with a crude approximation of the value function and improve it step by step until it stabilizes, at which point the implied policy is optimal.

  1. Start with an arbitrary guess for the value function:

  $$V_0(W)$$

  2. Proceed recursively by solving:

  $$V_1(W_t) = \max_{W_{t+1}} \; u(W_t - W_{t+1}) + \beta V_0(W_{t+1})$$

     - Since we now know V_0 (by construction), this maximization problem can be solved.
     - Find the first-order condition and solve for W_{t+1}.
     - Call this solution W^*. Substituting back gives:

  $$V_1(W_t) = u(W_t - W^*) + \beta V_0(W^*)$$

  3. Next step:

  $$V_2(W_t) = \max_{W_{t+1}} \; u(W_t - W_{t+1}) + \beta V_1(W_{t+1}),$$

     since now V_1 is known.

  4. Continue this process to generate a sequence of value functions and policy functions. Over time, this sequence converges to the true value function and its corresponding policy.

- *Intuition:* This method is like "training" the value function: we start with a rough guess, refine it step by step using the Bellman equation, and eventually converge to the true optimal solution.

Guess and Verify Method

- The second method involves **guessing and verifying** a solution to the Bellman equation.
- The idea is to make an *informed guess* about the functional form of the value function.
- For simple problems, the value function often shares the same general functional form as the period utility function.
- Example: If the period utility is logarithmic, we can often expect the value function to also take a logarithmic form.
- This method depends on the uniqueness of the solution to the Bellman equation. However, because it relies on a good guess, it is usually only applicable in relatively simple problems.
- *Intuition:* Guess and verify is like solving a riddle — if you guess the right "shape" of the solution, the math confirms it. But for complex problems, guessing becomes unrealistic, so this method is rarely used in practice.

  1. Set up the Bellman equation:
  $$V(W_t) = \max_{c_t, W_{t+1}} \; u(W_t, c_t) + \beta V(W_{t+1})$$
  2. Make a functional-form guess for the value function, e.g.:
  $$V(W) = XW \quad \text{where } X \text{ is an unknown coefficient.}$$
  3. Substitute the guess into the Bellman equation:
  $$V(W_t) = \max_{c_t, W_{t+1}} \; u(W_t, c_t) + \beta X W_{t+1}.$$
  4. Solve the maximization problem by finding the first-order condition and solving for W_{t+1} and c_t.
  5. Substitute the solution back into the Bellman equation. Check if the guess is consistent, i.e.,
  $$XW_t = XW_{t+1}.$$
  6. If a consistent X can be found, then the initial guess has been verified. Otherwise, modify the guess and repeat the process.
  7. By refining successive guesses (approximations), we can converge to the correct solution for the value and policy functions, much like in value function iteration.

- *Intuition:* Guess and verify works like "pattern recognition" — we assume the solution has a certain form, plug it in, and see if the math confirms it. If it fits, we've solved the problem; if not, we adjust and try again.

Policy function iteration

- This method is also known as **Howard's improvement algorithm**.
- The idea is to **guess a policy function** and then use it to help find the value function.
- Convergence is usually much faster than in the value function iteration method.
- For further details, see Ljungqvist and Sargent (2004), page 88.
- Each of the three main methods (value function iteration, guess and verify, policy function iteration) has its own advantages.
- However, each method is easier said than done. In practice, it is often impossible to compute even one Bellman iteration analytically.
- This is why computational (numerical) methods are so important in solving applied dynamic programming problems.
- In the next section, we will solve the cake-eating problem analytically under logarithmic preferences, using both value function iteration and guess-and-verify.
- Homework: Consider other economic problems that can be solved analytically.
    - **Two-period consumption-savings model:** Solve analytically for optimal C_1, C_2 given utility $u(C) = \log(C)$ and intertemporal budget constraint.
    - **Labor-leisure choice problem:** Derive optimal split between consumption and leisure under utility $u(C, L) = \log(C) + \theta \log(L)$ with time and budget constraints.
- *Intuition:* Policy function iteration speeds things up by focusing directly on the "rules of behavior" (the policy functions) rather than repeatedly recalculating the value function from scratch.

Euler equation approximation method

- This method is a common alternative to the solution techniques previously discussed.
- The key idea: compute the policy function **directly** without solving for the value function.
- Advantage: we do not need the explicit form of the value function (only that it is differentiable).
- The approach relies on optimality conditions — such as the Euler equation and the transversality condition (TVC) — along with the resource constraint.
- Since these conditions are typically nonlinear, we linearize them around the steady state. By solving this linear system, we obtain approximate policy functions.
- This produces an **approximate solution** using log-linearization.

- Note: this approach also applies under the direct method, which partly explains its popularity.

- Consider the one-dimensional functional equation:
$$V(x) = \max_{y \in G(x)} \{U(x, y) + \beta V(y)\},$$
where x is the state variable and y is the control variable.
- From earlier, the first-order condition is given by the Euler equation.
- However, the Euler condition alone is not sufficient. We also require the **transversality condition (TVC)**:
$$\lim_{t \to \infty} \beta^t U_x(x_t, x_{t+1}) x_t = 0$$
- Interpretation: the product of the marginal utility from the state x_t and the state level x_t must not grow asymptotically faster than $1/\beta$.
- Stokey–Lucas (section 4.5) show that the Euler equation + TVC are sufficient conditions for characterizing solutions to dynamic programming problems.
- These conditions hold under standard assumptions in neoclassical models: concave, differentiable utility and production functions; convex, nonempty choice sets.
- *Intuition:* The Euler equation ensures local optimality between periods, while the TVC rules out “Ponzi schemes” (endless borrowing) by guaranteeing sustainability in the long run.

2B.6 Solving the Cake-Eating Example

- **Consider the following infinite-horizon cake-eating problem:**
$$\max_{\{c_t\}_{t=1}^{\infty}, \{W_{t+1}\}_{t=1}^{\infty}} \sum_{t=0}^{\infty} \beta^t \log(C_t),$$
- **subject to** $W_{t+1} = (1 + r)W_t - C_t$ and a given value for W_1
- **Note:** The cake that is saved (i.e., not consumed) grows at a constant rate of $1 + r$
- **We will solve this problem using:**
  1. The value function iteration method
  2. The guess and verify method
  3. The Euler equation method
- **Intuition:** The agent allocates consumption over time to maximize utility from eating cake while accounting for the growth of uneaten cake. This trade-off gives rise to a dynamic optimization problem.

The value function iteration method

Full procedure

Step 0: Initial guess for value function

$$V_0(W) = \log(W)$$

Step 1: First iteration — Bellman equation

$$V_1(W_t) = \max_{W_{t+1}} \left\{ \log\left[(1+r)W_t - W_{t+1}\right] + \beta \log(W_{t+1}) \right\}$$

FOC:

$$\frac{1}{(1+r)W_t - W_{t+1}} = \frac{\beta}{W_{t+1}} \Rightarrow W_{t+1} = \frac{\beta(1+r)W_t}{1+\beta}$$

Then:

$$C_t = (1+r)W_t - W_{t+1} = \frac{(1+r)W_t}{1+\beta}$$

Substitute into $V_1(W_t)$:

$$\begin{aligned} V_1(W_t) &= \log\left[(1+r)W_t - W_{t+1}\right] + \beta \log(W_{t+1}) \\ &= \log\left[\frac{(1+r)W_t}{1+\beta}\right] + \beta \log\left[\frac{\beta(1+r)W_t}{1+\beta}\right] \\ &= (1+\beta)\log(W_t) + (1+\beta)\log\left(\frac{1+r}{1+\beta}\right) + \beta\log(\beta) \\ &= E_1 + (1+\beta)\log(W_t) \end{aligned}$$

Step 2: Second iteration

$$V_2(W_t) = \max_{W_{t+1}} \left\{ \log\left[(1+r)W_t - W_{t+1}\right] + \beta V_1(W_{t+1}) \right\}$$

$$V_1(W_{t+1}) = E_1 + (1+\beta)\log(W_{t+1})$$

Substitute:

$$V_2(W_t) = \max_{W_{t+1}} \left\{ \log\left[(1+r)W_t - W_{t+1}\right] + \beta E_1 + \beta(1+\beta)\log(W_{t+1}) \right\}$$

Drop constant:

$$\Rightarrow \max_{W_{t+1}} \left\{ \log\left[(1+r)W_t - W_{t+1}\right] + \beta(1+\beta)\log(W_{t+1}) \right\}$$

FOC:

$$\frac{1}{(1+r)W_t - W_{t+1}} = \frac{\beta(1+\beta)}{W_{t+1}} \Rightarrow W_{t+1} = \frac{\beta(1+\beta)(1+r)W_t}{1+\beta+\beta^2}$$

$$C_t = \frac{(1+r)W_t}{1+\beta+\beta^2}$$

Substitute into $V_2(W_t)$:

$$\begin{aligned}V_2(W_t) &= \log\left[\frac{(1+r)W_t}{1+\beta+\beta^2}\right] + \beta E_1 + \beta(1+\beta)\log\left[\frac{\beta(1+\beta)(1+r)W_t}{1+\beta+\beta^2}\right]\\ &= (1+\beta+\beta^2)\log(W_t) + (\text{constants})\\ &= E_2 + (1+\beta+\beta^2)\log(W_t)\end{aligned}$$

General pattern after j **steps:**

$$V_j(W_t) = E_j + \left(\sum_{s=0}^{j}\beta^s\right)\log(W_t) \quad \text{and} \quad W_{t+1} = \left(\frac{\sum_{s=1}^{j}\beta^s}{\sum_{s=0}^{j}\beta^s}\right)(1+r)W_t$$

As $j\to\infty$:

$$\sum_{s=0}^{\infty}\beta^s = \frac{1}{1-\beta}, \quad \sum_{s=1}^{\infty}\beta^s = \frac{\beta}{1-\beta} \Rightarrow W_{t+1} = \beta(1+r)W_t, \quad C_t = (1-\beta)(1+r)W_t$$

What does these policy functions tell us? The optimal policy is to save a constant fraction of the cake (regardless of the size of the cake) and eat the remaining fraction

The guess and verify method

Full procedure

Step 1: Bellman equation

$$V(W_t) = \max_{W_{t+1}}\{\log\left[(1+r)W_t - W_{t+1}\right] + \beta V(W_{t+1})\}$$

Step 2: Guess a functional form

$$V(W) = E + F\log(W), \quad \text{where } E, F \text{ to be determined}$$

Step 3: Substitute guess into Bellman equation

$$V(W_t) = \max_{W_{t+1}}\{\log\left[(1+r)W_t - W_{t+1}\right] + \beta\left[E + F\log(W_{t+1})\right]\}$$

Step 4: First-order condition

$$\frac{1}{(1+r)W_t - W_{t+1}} = \frac{\beta F}{W_{t+1}} \Rightarrow W_{t+1} = \frac{\beta F(1+r)}{1+\beta F}W_t$$

Step 5: Substitute W_{t+1} **into value function**

$$\begin{aligned}V(W_t) = \log&\left[(1+r)W_t - \frac{\beta F(1+r)}{1+\beta F}W_t\right]\\ &+ \beta E + \beta F\log\left(\frac{\beta F(1+r)}{1+\beta F}W_t\right)\\ &= \log\left[\frac{(1+r)W_t}{1+\beta F}\right] + \beta E + \beta F\log\left[\frac{\beta F(1+r)}{1+\beta F}W_t\right]\\ &= \log\left(\frac{1+r}{1+\beta F}\right) + \log(W_t) + \beta E + \beta F\left[\log\left(\frac{\beta F(1+r)}{1+\beta F}\right) + \log(W_t)\right]\end{aligned}$$

Step 6: Combine log terms

$$V(W_t) = \left[\log\left(\frac{1+r}{1+\beta F}\right) + \beta E + \beta F \log\left(\frac{\beta F(1+r)}{1+\beta F}\right)\right] + [1+\beta F]\log(W_t)$$

Compare with guess: $V(W_t) = E + F\log(W_t)$
Match coefficients:

$$F = 1 + \beta F \Rightarrow F(1-\beta) = 1 \Rightarrow F = \frac{1}{1-\beta}$$

$$E = \log\left(\frac{1+r}{1+\beta F}\right) + \beta E + \beta F \log\left(\frac{\beta F(1+r)}{1+\beta F}\right)$$

Solve for $E(1-\beta)$:

$$\begin{aligned} E(1-\beta) &= \log\left(\frac{1+r}{1+\beta F}\right) + \beta F \log\left(\frac{\beta F(1+r)}{1+\beta F}\right) \\ &= \log[(1+r)(1-\beta)] + \frac{\beta}{1-\beta}\log[\beta(1+r)] \\ &= \log(1+r) + \log(1-\beta) + \frac{\beta}{1-\beta}\left[\log(\beta) + \log(1+r)\right] \end{aligned}$$

Final expression for E:

$$E = \frac{1}{1-\beta}\left[\log(1-\beta) + \frac{\beta}{1-\beta}\log(\beta) + \left(1 + \frac{\beta}{1-\beta}\right)\log(1+r)\right]$$

Value function:

$$V(W_t) = E + \frac{1}{1-\beta}\log(W_t)$$

Policy function (from FOC):

$$W_{t+1} = \frac{\beta F(1+r)}{1+\beta F} W_t = \beta(1+r)W_t$$

Consumption:

$$C_t = (1+r)W_t - W_{t+1} = (1+r)(1-\beta)W_t$$

Envelope condition (verification):

$$V'(W_t) = \frac{1+r}{(1+r)W_t - W_{t+1}} = \frac{1+r}{(1+r)(1-\beta)W_t} = \frac{1}{(1-\beta)W_t} = \frac{F}{W_t} \Rightarrow F = \frac{1}{1-\beta} \quad \text{(consistent)}$$

The Euler equation method

Full procedure

Step 1: Bellman equation

$$V(W_t) = \max_{W_{t+1}} \{\log [(1+r)W_t - W_{t+1}] + \beta V(W_{t+1})\}$$

Step 2: Define consumption

$$C_t = (1+r)W_t - W_{t+1}$$

Step 3: First-order condition

$$\frac{1}{C_t} = \beta V'(W_{t+1})$$

Step 4: Envelope condition

$$V'(W_t) = \frac{1+r}{C_t}$$

Step 5: Euler equation

$$\frac{1}{C_t} = \beta \cdot \frac{1+r}{C_{t+1}} \Rightarrow C_{t+1} = \beta(1+r)C_t$$

Step 6: Forward iteration

$$C_{t+j} = [\beta(1+r)]^j \, C_t$$

Step 7: Use transition law

$$W_t = \frac{W_{t+1}}{1+r} + \frac{C_t}{1+r} \Rightarrow W_{t+1} = \frac{W_{t+2}}{1+r} + \frac{C_{t+1}}{1+r}$$

$$\Rightarrow W_t = \left(\frac{1}{1+r}\right)^2 W_{t+2} + \left(\frac{1}{1+r}\right)^2 C_{t+1} + \frac{C_t}{1+r}$$

Step 8: Forward-substitute T times

$$W_t = \left(\frac{1}{1+r}\right)^T W_{t+T} + \frac{1}{1+r}\sum_{j=0}^{T-1} \left(\frac{1}{1+r}\right)^j C_{t+j}$$

Step 9: Impose transversality condition (TVC)

$$\lim_{T\to\infty} \left(\frac{1}{1+r}\right)^T W_{t+T} = 0$$

Then:

$$W_t = \frac{1}{1+r}\sum_{j=0}^{\infty} \left(\frac{1}{1+r}\right)^j C_{t+j} = \sum_{j=0}^{\infty} \left(\frac{1}{1+r}\right)^{j+1} C_{t+j}$$

Step 10: Substitute Euler path of consumption

$$C_{t+j} = [\beta(1+r)]^j C_t$$

$$(1+r)W_t = \sum_{j=0}^{\infty} \left(\frac{1}{1+r}\right)^j [\beta(1+r)]^j C_t = \sum_{j=0}^{\infty} [\beta]^j C_t = \frac{1}{1-\beta} C_t$$

Step 11: Solve for optimal consumption

$$C_t = (1-\beta)(1+r)W_t$$

Step 12: Use transition equation to find capital

$$W_{t+1} = (1+r)W_t - C_t = (1+r)W_t - (1-\beta)(1+r)W_t = \beta(1+r)W_t$$

The Euler equation balances the marginal benefit of consuming today with the discounted benefit of saving for future consumption. It shows that optimal consumption grows at a **constant rate over time.** As a result, the agent always saves a fixed fraction of wealth and consumes the rest.

2B.7 A Little Bit of Stokey and Lucas (1989)

- A formal abstract treatment of dynamic programming is beyond the scope of this class.
- Stokey and Lucas (1989) provide a rigorous mathematical treatment of deterministic and stochastic dynamic programming.
- Here I want to simply talk about some of their key results and put them into context of what we have done (without formally covering the mathematical proof of each theorem).
- This may help those students who wish to pursue further graduate studies where the Stokey-Lucas text is standard reading in all dynamic programming or PhD macroeconomic courses.
- **Intuition:** This section introduces key insights from a foundational text in dynamic programming to help bridge classroom concepts with more advanced, formal graduate-level theory.
- As in Stokey and Lucas (1989), the Bellman equation with discounting can be expressed as:
$$V(s) = \max_{s' \in \Gamma(s)} \sigma(s, s') + \beta V(s'). \tag{15}$$
- Time subscripts are removed to emphasize the stationarity of the problem. Future variables are denoted with a prime.
- Here, $\sigma(\cdot)$ is the payoff function for the current period, s is the state vector, s' is the control vector, and the discount factor is $0 < \beta < 1$.

- The policy function that defines the optimal choice of the control (i.e., the future state) given the current state is:
$$s' = \phi(s). \tag{16}$$

- We are interested in solving for $\phi(s)$, which requires solving the Bellman equation (15). Thus, we must determine the value function $V(s)$ that satisfies (15).

- It's crucial to recognize that the value function is defined as the solution to the functional equation (15).

- Depending on the assumptions made about the payoff function and the transition function, we can determine conditions under which a solution to the Bellman equation exists.

- **Intuition:** This slide abstracts the dynamic programming problem into its most general mathematical form. Solving the Bellman equation yields both the value function and optimal decision rules, foundational in recursive economic models.

Theorem 1

- **Theorem 1:** Assume that $\sigma(s, s')$ is real-valued, continuous, and bounded, $0 < \beta < 1$, and that the constraint set $\Gamma(s)$ is nonempty, compact-valued, and continuous. Then there exists a unique value function $V(s)$ that solves (15):
$$V(s) = \max_{s' \in \Gamma(s)} \sigma(s, s') + \beta V(s').$$

- The proof of this result is given in Stokey-Lucas, Theorem 4.6 (page 79).

- The key tools for proving Theorem 1 are:
  1. Contraction Mapping Theorem (SL, Theorem 3.2, page 50)
  2. Blackwell's Sufficiency Conditions (SL, Theorem 3.3, page 54)

- Theorem 1 provides the theoretical foundation for the value function iteration method.

- The idea behind Theorem 1:
  - Begin by guessing an initial value function $W(\cdot)$.
  - Apply an operator T to generate a new value function:
$$T(W)(s) = \max_{s' \in \Gamma(s)} \sigma(s, s') + \beta W(s').$$
  - This operator maps a guess W into a new function $T(W)$.

- **Intuition:** Under the right conditions, repeatedly applying the Bellman operator to any initial guess will converge to the unique value function. This is the mathematical justification behind value function iteration.

- Clearly, any $V(s) = T(V)(s)$ is a solution to (15). So, we can reduce the problem to finding fixed points of the operator $T(W)$.
- If the transformation T is a contraction, we can apply the contraction mapping theorem, which implies:
  1. There is a unique fixed point.
  2. This fixed point can be reached through iterative application of T, starting from any initial guess.
- Using the contraction mapping theorem guarantees the existence (and uniqueness) of the value function solution.
- What ensures T is a contraction? We invoke **Blackwell's sufficiency conditions**, which require:
  - **Monotonicity**
  - **Discounting**
- **Intuition:** This slide explains how we can rigorously justify convergence of value function iteration using fixed point theory, particularly when the Bellman operator satisfies basic mathematical properties.

Theorem 2

- **Theorem 2:** If, in addition to the conditions in Theorem 1, $\sigma(s, s')$ is concave and $\Gamma(s)$ is convex, then the unique solution to (15) is strictly concave, and the associated policy function is a continuous, single-valued function.
- The proof of this result is given in Stokey-Lucas, Theorem 4.8 (page 81).
- We have often used the Euler equation to characterize aspects of the optimal solution.
- To do so rigorously, we require:
  (a) The strict concavity of the value function (from Theorem 2);
  (b) The first-order condition of (15);
  (c) That the value function is differentiable (conditions for this are given in Stokey-Lucas, Theorem 4.11, page 85).
- **Intuition:** This theorem ensures that not only does a solution exist, but it is well-behaved — the value function is nicely shaped, and the policy function reacts smoothly to changes in the state.