Dynamic Programming part 1

Investment, Finance, Asset Prices ECON5068

Thomas Walsh

Adam Smith Business School

Lecture Overview

- Dynamic Programming
- Wealth and Consumption Choice – A Cake Eating Problem
- Essential Reading:
  - Adda and Cooper Dynamic Economics: Quantitative Methods and Applications - Chapter 1. (See Moodle PDF)
  - Gregory, Chow Dynamic Economics: Optimization by the Lagrange Method - Chapter 2

Dynamic Programming

Dynamic Programming - Introduction

- In the last lecture, we used **Lagrange multipliers** to solve the **optimisation** problem of the firm.
- **Dynamic Programming** is an alternate method that can be used to solve optimisation problems.
- Developed in the 1940s by **Richard Bellman** at RAND Corporation
- Solves multistage decision-making problems by decomposing into smaller subproblems
- The approach is different yet gives an **identical solution**
- The name was chosen to avoid words like "**research, planning**" but still related to decision making, hence the credibility of "**Dynamic**" from Physics with the somewhat uninformative "**Programming**"
- something like **Chained Decomposition Solution Method** might make more sense

Dynamic Programming - Introduction

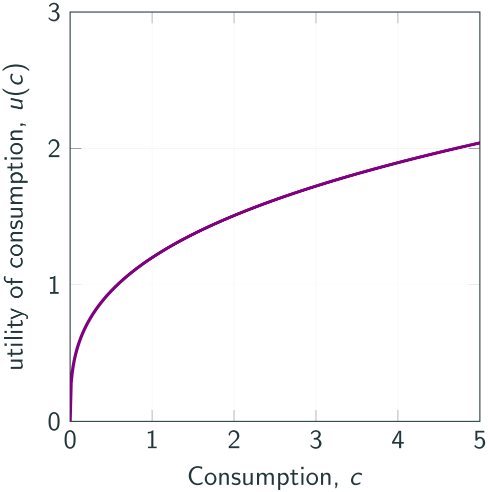
- **Dynamic Programming** is popular since it is **easy to implement numerically** with a computer.
- Widely used across Econ, Finance, Compuster Science, Operations, Engineering, Game Theory, Machine Learning ... basically any quantitative field
- Most **modern macro(-finance) models** written in **recursive** (self-similar) form, essential tool
- We will learn dynamic programming using an **example: Eating a Cake**

Eating a Cake

Dynamic Programming with Cake-Eating Example

- Suppose you have a cake of size W_1, **your wealth**. You have T **periods to consume** this cake.
- Every period $t = 1, 2...T$ you **consume some of the cake** and **save the rest**. The initial size of the cake at $t = 1$ is W_1.
- Assume that the cake **cannot melt (depreciate) or grow**.
- Let c_t represent the consumption of cake at time t and $u(c_t)$ the flow of utility (satisfaction) from this consumption.
- Assume $u(.)$ is real-valued, **continuous, differentiable and concave** and consumption should always be non-negative.
- Examples: $c^{0.5}, ln(c), \frac{c^{1-\sigma}}{1-\sigma}$. Might need to be careful with negative u in context

Example of Utility Function



- increasing and concave (the next piece is not as good as the last)

Preferences

- The **life-time utility** from consuming the cake is given by the **discounted sum** of all current and future utility of consumption:

$$u(c_1) + \beta u(c_2) + \beta^2 u(c_3) + ... + \beta^{T-1} u(c_T) \tag{1}$$

- That is,

$$\sum_{t=1}^{T} \beta^{t-1} u(c_t) \tag{2}$$

where $0 \leq \beta \leq 1$ is the **discount factor**, a measure of (im)patience.

Law of Motion of Cake

- The **evolution of cake** size (a.k.a law of motion) every period is given by::

$$W_{t+1} = W_t - c_t \tag{3}$$

  Problem: How would you find the optimal path of consumption $\{c_t\}_{t=1}^{T}$

- In other words, what is the level of **consumption every period** that **maximizes your lifetime utility** in equation (2) above.

- We are looking for a consumption **plan** for all periods jointly: $\{c_t\}_{t=1}^{T}$.

Cake-Eating Example - Sequential Lagrangian Approach

- One approach is to use the method of Lagrange multipliers.

- This is then a constrained optimization problem where:

$$\max_{\{c_t, W_{t+1}\}_{t=1}^{T}} \left[\sum_{t=1}^{T} \beta^{t-1} u(c_t) \right]$$

- subject to the constraint:

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

for all $t = 1, 2, \ldots T$.

Cake-Eating Example - Sequential Lagrangian Approach

The Lagrangian function can be written as:

$$\mathcal{L} = \sum_{t=1}^{T} \beta^{t-1} \left[u(c_t) - \lambda_t (W_{t+1} - W_t + c_t) \right]$$

Note: This is a dynamic optimization problem, we have an objective function and a constraint at every period t. All future values need to be discounted.

$$\begin{aligned} \mathcal{L} = \big[& u(c_1) + \beta u(c_2) + ... + \beta^{t-1} u(c_t) + ... \\ & - \lambda_1 (W_2 - W_1 + c_1) - \beta \lambda_2 (W_3 - W_2 + c_2) ... \\ & - \beta^{t-1} \lambda_t (W_{t+1} - W_t + c_t) - \beta^t \lambda_{t+1} (W_{t+2} - W_{t+1} + c_{t+1}) ... \end{aligned} \tag{4}$$

Remember! Like the Tobin model, we have to check for $t+1$-variables in two places

Cake-Eating Example - Sequential Lagrangian Approach

The necessary condition for maximizing this lagrangian function is given by the three FOCs:

$$\frac{\partial L}{\partial c_t} = 0 \Rightarrow u'(c_t) = \lambda_t$$

$$\frac{\partial L}{\partial W_{t+1}} = 0 \Rightarrow \lambda_t = \beta \lambda_{t+1}$$

$$\frac{\partial L}{\partial \lambda_t} = 0 \Rightarrow W_{t+1} = W_t - c_t$$

Cake-Eating Example - Euler Equation Intuition

- From eqs (1), (2), we get the **Euler equation**, intertemporal optimality condition:

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

- LHS represents the **marginal loss in utility** when you sacrifice a **small unit of consumption** and the RHS is the **discounted marginal gain in utility** from this extra unit of consumption next period.
- If the Euler equation holds, then it is **impossible to increase utility** by moving consumption across adjacent periods given a candidate solution $\{\tilde{c}_t\}_1^T = \{c_t^*\}_1^T$.
- **No Abitrage condition:** $+\beta u_{c,t+1} dc - u_{c,t} dc = 0$ or $-\beta u_{c,t+1} dc + u_{c,t} dc = 0$ depending on which way transfer consumption (small dc)

Euler Equation links periods $(t, t+1)$

- From eqs (1), (2), we get the **Euler equation**, **intertemporal optimality** condition, for $t = 1, ..., T-1$:

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

- Example: if $u(c) = ln(c)$ implies a (negative) growth path:

$$\frac{1}{c_t} = \beta \frac{1}{c_{t+1}} \quad \Rightarrow \quad \frac{c_{t+1} - c_t}{c_t} = (\text{\% Growth in } c) = -(1-\beta)$$

- Euler Equation makes a chain of pairs:

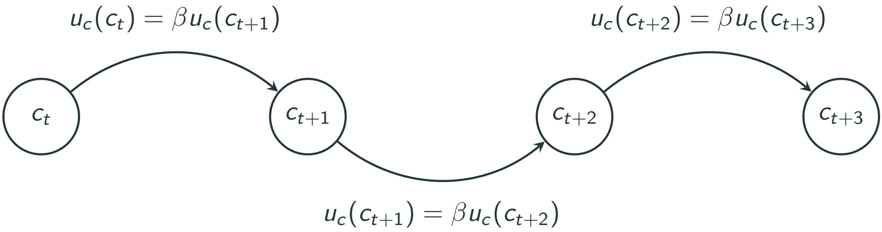
$$(c_1, c_2), (c_2, c_3), (c_3, c_4), ..., (c_{99}, c_{100}) \tag{5}$$

$$(c_1, \beta c_1), (\beta c_1, \beta^2 c_1), (\beta^2 c_1, \beta^3 c_1), ..., (\beta^{98} c_1, \beta^{99} c_1) \tag{6}$$

$$c_t = \beta^{t-1} c_1 \tag{7}$$

But we don't yet know c_1. Once we solve for that, we get the full chain

Euler Equation links periods $(t, t+1)$



Cake-Eating Example - Sequential Lagrangian Approach

Since this is a **finite time horizon** problem, we need to have a **terminal condition**.

For **maximum utility**, there should not be any cake left over at the end of the last period (no waste). That is,

$$W_{T+1} = 0 \tag{END}$$

This terminal condition naturally implies that the sum of consumption across all periods should equal the total size of the cake (resource constraint, RC):

$$\sum_{t=1}^{T} c_t = W_1 \ (e.g. = 100) \tag{RC}$$

Using the value of W_1 (RC) and eq.s (EE) and (END), we can find the optimal path of consumption $\{c_t^*\}_{t=1}^{T}$ that maximizes utility.

For log-utility we can use pen and paper

We can plug the Euler Equation bridges into consumption, and use RC:

$$\sum_{t=1}^{T} c_t = W_1 \Rightarrow \sum_{t=1}^{T} \beta^{t-1} c_1 = W_1$$

We can arrange the sum:

$$c_1(1 + \beta + \beta^2 + ... + \beta^{T-1}) = W_1$$

This is a geometric sum, we know from the toolkit how to solve this:

$$c_1 \frac{(1-\beta^T)}{1-\beta} = W_1$$

Solving for c as a function of parameters for patience and total periods:

$$c_1 = \frac{(1-\beta)}{1-\beta^T} W_1$$

And this **nests** the well-known **infinite horizon solution** ($T \to \infty$)

$$c_t = (1-\beta)W_t \quad \forall t$$

Consume (e.g.) 5 percent of remaining cake (like every period is like the start)

Time for Some Drawing!

Consumption and Wealth Sequences

Blue line: impatient; Purple: patient

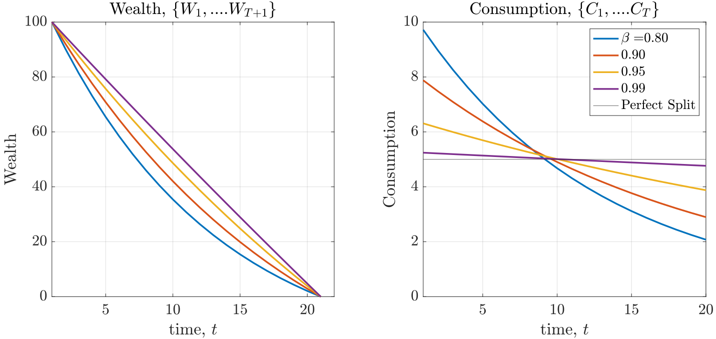


Figure 1: $\{W_{t+1}, c_t\}_{t=1}^{T=20}$, $W_1 = 100$, $u(c) = \frac{c^{1-\sigma}}{1-\sigma}$

$$W_{t+2} = (1 + \beta^{1/\sigma})W_{t+1} - (\beta^{1/\sigma})W_t; \text{ given } W_{T+1} = 0; W_1 = 100$$

Cake-Eating Example - Value Function

- The solution to this T-period cake eating problem is found by substituting the optimal path of consumption in the lifetime utility function.

- We will denote this maximum as $V^T(W_1)$:

$$V^T(W_1) = \max \left[\sum_{t=1}^{T} \beta^{t-1} u(c_t) \right] = \sum_{t=1}^{T} \beta^{t-1} u(c_t^*)$$

- $V(W_1)$ is called as a **value function** and here it represents the **maximum T period utility of consumption** given an initial level of cake size W_1.

Cake-Eating Example - Dynamic Programming Approach

- Suppose we change this cake eating problem by **adding a period 0** and giving an initial cake size of W_0.
- We can again solve this by formulating **a new Lagrangian** for the $T+1$ period problem.
- However, **a better way** would be to somehow make use of the T period solution that we found, $V^T(W_1)$ to create $V^{T+1}(W_0)$
- Dynamic Programming (DP) provides means for doing this.
- **DP essentially converts a general T period problem into a 2 period one.**

Cake-Eating Example - Dynamic Programming Approach

- DP breaks down the optimal path into two parts, what is **optimal today** and the **optimal continuation path**.
- Given W_0, the optimization problem can be written as:

$$V^{T+1}(W_0) = \max_{\{c_t, W_{t+1}\}_{t=0}^{T}} \left[\sum_{t=0}^{T} \beta^t u(c_t) \right] \tag{8}$$

$$= \max_{\{c_t, W_{t+1}\}_{t=0}^{T}} \left[u(c_0) + \sum_{t=1}^{T} \beta^t u(c_t) \right]$$

$$= \max_{\{c_t, W_{t+1}\}_{t=0}^{T}} \left[u(c_0) + \beta \sum_{t=1}^{T} \beta^{t-1} u(c_t) \right]$$

$$= \max_{c_0, W_1} \left[u(c_0) + \beta \max_{\{c_t, W_{t+1}\}_{t=1}^{T}} \left[\sum_{t=1}^{T} \beta^{t-1} u(c_t) \right] \right]$$

$$V^{T+1}(W_0) = \max_{c_0, W_1} \left[u(c_0) + \beta V^{T}(W_1) \right] \tag{9}$$

note: calendar time isn't important per se, how much time left matters!

Cake-Eating Example - Dynamic Programming Approach

- Subject to the constraint

$$W_1 = W_0 - c_0$$

- Note V^T here denotes value function for the T-**periods-left model** not value function at time T!!! Best to think of this as V_t^T, V_{t+1}^{T-1} for some time t.
- In terms of **time-t/calendar-time** notation, the general **Bellman equation** is:

$$V_t(W_t) = \max_{c_t, W_{t+1}} \left\{ u(c_t) + \beta V_{t+1}(W_{t+1}) \right\}$$

  where $t = 0, 1, \ldots T$.
- This is a **functional equation** - the unknown is now a function V.
  - depends on cake left W_0
  - and number of periods left $T + 1$

Cake-Eating Example - Dynamic Programming Approach

- **So instead of choosing the entire path of c_t, we are just choosing c_0.**
- The rest of the path is optimally determined by the **value function**, $V^T(W_1)$.
- Once c_0 and hence W_1 is determined, the value function summarizes the rest of the problem
- This is the **principle of optimality** due to **Richard Bellman**: we can represent the full dynamic problem as a sequence of recursive 2 period problems:
- **Optimal Today + Optimal Continuation Path** (we know we will be optimising!)

Cake-Eating Example - Dynamic Programming Approach

- The **Bellman equation** for the cake eating problem is then written as

$$V_t(W_0) = \max_{c_t, W_{t+1}} [u(c_t) + \beta V_{t+1}(W_1)]$$

where $t = 0, 1, \dots T$. Here V_t is the value function at any time t and V_{t+1} is the value function for the next period $t + 1$.

- The **solution** to this problem is given by the **decision rules (functions)** for consumption and next period cake size: $c_t(W_t)$ and $W_{t+1}(W_t)$.
- To obtain these decision rules, we need to find the **unknown value function** $V_t(W_t)$.
- Since this is a finite horizon problem, we can achieve this task easily. **Start with the last period** T where $V_{T+1} = 0$ and work backwards to obtain all the other value functions and decision rules.

Cake-Eating Example - Dynamic Programming Approach

Substituting for W_{t+1} from the constraint, we can write eq. (7) as:

$$V_t(W_t) = \max_{c_t} \left[u(c_t) + \beta V_{t+1}(W_t - c_t)\right]$$

The **first order condition** of this value function problem [EC] is given by:

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

Denote the solution to the problem, optimal consumption by $c_t^* = h_t(W_t)$.

Then the value function is

$$V_t(W_t) = \left[u(h_t(W_t)) + \beta V_{t+1}(W_t - h_t(W_t))\right]$$

Taking the derivative w.r.t W_t, we get the **Envelope condition**[1]

$$V'_t(W_t) = \left[u'(h_t(W_t))h'_t(W_t) + \beta V'_{t+1}(\cdot)[1 - h'_t(W_t)]\right] \tag{10}$$

$$= u'(c_t) \tag{11}$$

[1] borrow the FOC for the second term sub

Cake-Eating Example - Dynamic Programming Approach

- **Value** is defined by W_t **cake size today**, and **number of periods** left T, not by when we start the process (Wednesday, Thursday, Friday)...

$$V(a) = V_t(a) = V_{t+1}(a) \qquad \textit{for some number } a$$

- Taking one period forward, with **stationarity** of the value function:

$$V'_{t+1}(W_{t+1}) = u'(c_{t+1})$$

- The FOC along with the above envelope condition together imply the Euler equation,

$$\boxed{u'(c_t) = \beta u'(c_{t+1}) \quad \text{for } t = 0, 1, 2, \ldots T - 1}$$

Recursive **Dynamic Programming** Solution = **Sequential Lagrangian** Solution

Infinite Horizon

Cake-Eating Example - Infinite Horizon

- **Suppose we allow the horizon to go to infinity**.
- As before, one can consider solving the infinite horizon sequence problem given by:

$$\max_{\{c_t, W_{t+1}\}_0^\infty} \sum_{t=0}^{\infty} \beta^t u(c_t)$$

along with the transition equation of

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

for $t = 0, 1, 2, \ldots \infty$ and some given $W_0 > 0$.

Infinite Horizon - Dynamic Programming

- Since the time **horizon is infinite**, the future **from today** and the future **from tomorrow** is of the same length (which is infinity).
- Value function is not a function of the time period, but only of the cake size.
- The value function for the infinite horizon case is

$$V(W_t) = \max_{\{c_t, W_{t+1}\}_0^\infty} \sum_{t=0}^{\infty} \beta^t u(c_t)$$

Infinite Horizon - Dynamic Programming

We can form the **Bellman equation** by breaking down this infinite sequence into a **recursive** two-period problem:

$$V(W_t) = \max_{\{c_t, W_{t+1}\}_0^\infty} \sum_{t=0}^{\infty} \beta^t u(c_t) \tag{12}$$

$$= \max_{\{c_t, W_{t+1}\}_0^\infty} \left[u(c_0) + \sum_{t=1}^{\infty} \beta^t u(c_t) \right] \tag{13}$$

$$= \max_{c_0, W_1} \left[u(c_0) + \max_{\{c_t, W_{t+1}\}_1^\infty} \sum_{t=1}^{\infty} \beta^t u(c_t) \right] \tag{14}$$

$$= \max_{c_0, W_1} \left[u(c_0) + \beta \max_{\{c_t, W_{t+1}\}_1^\infty} \sum_{t=0}^{\infty} \beta^t u(c_{t+1}) \right] \tag{15}$$

$$V(W_t) = \max_{c_0, W_1} [u(c_0) + \beta V(W_{t+1})] \tag{16}$$

e.g. $V(100) = u(10) + \beta V(100 - 10)$

Infinite Horizon - Dynamic Programming

- So the **infinite horizon dynamic programming problem** is

$$V(W) = \max_{c,W'} \left\{ u(c) + \beta V(W') \right\} \quad \text{for all } W \tag{17}$$

$$s.t. \quad W' = W - c \tag{18}$$

- Variables with **prime** denote **future values**[2].

- $V(W)$ **is the value** of the infinite horizon cake eating problem or the **maximal utility from this consumption**.

- $W' = W - c$ is the **state transition equation** or equivalently the evolution of cake size.

[2]not to be confused with derivatives, that is W denotes W_t and W' denotes W_{t+1}

Infinite Horizon - Remarks

- In general, we use **primes** to denote **future values** when we are looking for a stationary solution to an infinite horizon problem.

- The value function here is **stationary**, that is:

$$V_t(W) = V_{t+k}(W) = V(W) \quad \text{for any } k > 0$$

- **Stationarity** means time-invariant, that is the value function or policy functions are optimal and do not change with time.

- Remember these functions denote **a path or a rule**, so **stationarity** here means that this **path is constant** (not the actual variable).

Infinite Horizon - Remarks

- The two **policy functions** maps the **state variables** to controls (choices).
- In this problem, the two policy functions are:

$$W'(W) \text{ and } c(W)$$

  next period cake size and consumption.

- **State = Sufficient** knowing W is sufficient to summarize all the data we need for our problem. W is therefore, the **state variable**
- If I know $V(W)$: tell me W and I will tell you how much to consume and to save

Infinite Horizon - State and Control Variables

- What are the state and control (choice) variables?
- The **state variable** is the size of the cake (W) that is given at the start of any period.
- The cake size completely summarizes all information from the past that is needed for the forward looking optimization problem.
- The **control variable** is the variable that is being **chosen**. In this case, it is the level of consumption in the current period, c and next period cake size W'.
- The **transition (or the constraint)** desribes the dependence of the state tomorrow on the state today and the control today:

$$W' = W - c$$

Infinite Horizon - State and Control Variables

- Alternatively, we can write the DP, in (10), as:

$$V(W) = \max_{W'} \left\{ u(W - W') + \beta V(W') \right\}$$

  where we have substituted the constraint so that we have to choose only tomorrow's cake size.

- **Either specification will yield the same result.** Fewer choice variables are easier to work with.

- This expression is a **functional equation** and is often called a **Bellman equation after Richard Bellman**, the originator of dynamic programming.

- Note that the **unknown in the Bellman equation is the value function itself**: the idea is to find a function $V(W)$ that satisfies this condition for all W.

Items for Review

- Sequential Lagrangian
- Shadow Price
- Consumption/Saving with no production, depreciation
- Sequential solution with Euler Equation
- Finite Horizon
- Recursive Approach
- Bellman Equation
- Continuation Value
- Infinite Horizon
- State and Choice/Control Variables

Appendix: For all the rest: Shooting Algorithm

1. **Initial Condition:** Start with W_1, e.g. 100.
2. **Update:** Use Euler Equation in terms of cake, **second-order difference eqn**:

$$u_c(W_t - W_{t+1}) = \beta u_c(W_{t+1} - W_{t+2})$$

3. Rearrange, and guess W_2:

$$W_{t+2} = W_{t+1} - u_c^{-1}((1/\beta)u_c(W_t - W_{t+1}))$$

4. Start: We have W_1, guess W_2, this implies W_3. Then we can roll forward to get $W_4, ..., W_{T+1}$. This is the first shot. Aim for zero.
5. **Terminal condition** Adjust guess W_2, keep shooting until $W_{T+1} \approx 0$.
6. **Optimal Consumption** path: $C_t = W_{t-1} - W_t$

- Fast numerical methods in **matlab, julia** etc to solve (Bisection!)
- One can also do a **reverse shot**: We know $W_{T+1} = 0$, guess W_T to imply $W_{T-1}, ..., W_1$, and aim for starting $W_1 = 100$.

Recursive model solution(s) 1: constraint substituted

Derivative w.r.t W':

$$V(W) = \max_{W'} \left\{ u(W - W') + \beta V(W') \right\} \tag{19}$$

$$[W' :] \quad - u_c(W - W') + \beta \left(\frac{\partial V(W')}{\partial W'} \right) = 0 \tag{FOC}$$

We can use the **envelope condition** and roll forward one period for the derivative

$$[EC :] \quad V_W(W) = u_c(W - W') = u_c(c) \tag{20}$$

$$\Rightarrow V_{W'}(W') = u_c(W' - W'') = u_c(c') \tag{21}$$

Combined:

$$\boxed{u_c(c) = \beta u_c(c')} \tag{22}$$

Recursive model solution(s) 2: constraint explicit

$$V(W) = \max_{c,W'} \left\{ u(c) + \beta V(W') \right\} \quad s.t. \quad c + W' = W \tag{23}$$

We can still build a recursive Lagrangian with one (1!) constraint

$$\mathcal{L} = u(c) + \beta V(W') - \lambda(c + W' - W)$$

FOCs wrt (c, W'):

$$[c:] \quad u_c(c) - \lambda = 0 \tag{24}$$

$$[W':] \quad \beta \left(\frac{\partial V(W')}{\partial W'} \right) - \lambda = 0 \tag{25}$$

Envelope Condition again (we can ignore indirect effects)

$$[EC:] \quad \frac{\partial V(W)}{\partial W} = \frac{\partial \mathcal{L}}{\partial W} = \lambda; \quad \frac{\partial V(W')}{\partial W'} = \lambda' \tag{26}$$

Combined:

$$\boxed{u_c(c) = \beta u_c(c')} \tag{27}$$