Macroeconomics III

Dynamic Programming

Diego Rodrigues

SciencesPo
diego.desousarodrigues@sciencespo.fr

Fall 2022

Infinite Horizon Optimization

1. Time is discrete and the horizon is infinite: $t \in \{0, 1, ...\}$.
2. Single infinitely lived household.
3. Single good representing capital and consumption.
4. Production takes one period.
5. The amount y_t of good produced in period t depends on the stock of capital k_t in this period.

The Model

- We assume that $y_t = F(k_t)$ where $F : \mathbb{R}^+ \to \mathbb{R}^+$ is continuous on $\mathbb{R}^+$, continuously differentiable on $(0, \infty)$, strictly concave, with

$$F(0) = 0, \quad \forall k > 0, \quad F'(k) > 0, \quad \text{and} \quad \lim_{k \to \infty} F'(k) = 0$$

The Model

- We assume that $y_t = F(k_t)$ where $F : \mathbb{R}^+ \to \mathbb{R}^+$ is continuous on $\mathbb{R}^+$, continuously differentiable on $(0, \infty)$, strictly concave, with

$$F(0) = 0, \quad \forall k > 0, \quad F'(k) > 0, \quad \text{and} \quad \lim_{k \to \infty} F'(k) = 0$$

- In each period t, the output y_t combined with the depreciated capital stock $(1 - \delta)k_t$ can be divided between consumption c_t and new capital stock k_{t+1}.

The Model

- We assume that $y_t = F(k_t)$ where $F : \mathbb{R}^+ \to \mathbb{R}^+$ is continuous on $\mathbb{R}^+$, continuously differentiable on $(0, \infty)$, strictly concave, with

$$F(0) = 0, \quad \forall k > 0, \quad F'(k) > 0, \quad \text{and} \quad \lim_{k \to \infty} F'(k) = 0$$

- In each period t, the output y_t combined with the depreciated capital stock $(1 - \delta)k_t$ can be divided between consumption c_t and new capital stock k_{t+1}.
- Implicitly, there is a **one-to-one technology transforming capital in good**.

The Model

- The household's initial stock of capital is $k_0 > 0$.

The Model

- The household's initial stock of capital is $k_0 > 0$.
- His utility from an infinite consumption stream $c = (c_t)_{t \geq 0}$ is

$$U(c) := \sum_{t=0}^{\infty} \beta^t u(c_t),$$

where $\beta \in (0,1)$ and $u : \mathbb{R}^+ \to \mathbb{R}^+$ is bounded, continuous on $\mathbb{R}^+$, continuously differentiable on $(0, \infty)$, strictly increasing, and strictly concave.

The Set of Alternatives

- We let $f : \mathbb{R}_+ \to \mathbb{R}_+$ be defined by:

$$f(k) := F(k) + (1 - \delta)k$$

The Set of Alternatives

- We let $f : \mathbb{R}_+ \to \mathbb{R}_+$ be defined by:

$$f(k) := F(k) + (1 - \delta)k$$

- The household maximizes his discounted utility $U(c)$ among all consumption streams c satisfying the resource feasibility restrictions

$$c_0 + k_1 \leqslant f(k_0)$$

and for every $t \geqslant 1$,

$$c_t + k_{t+1} \leqslant f(k_t)$$

where $c_t \geqslant 0$ and $k_{t+1} \geqslant 0$.

The Set of Alternatives

- We let $f : \mathbb{R}_+ \to \mathbb{R}_+$ be defined by:

$$f(k) := F(k) + (1 - \delta)k$$

- The household maximizes his discounted utility $U(c)$ among all consumption streams c satisfying the resource feasibility restrictions

$$c_0 + k_1 \leqslant f(k_0)$$

and for every $t \geqslant 1$,

$$c_t + k_{t+1} \leqslant f(k_t)$$

where $c_t \geqslant 0$ and $k_{t+1} \geqslant 0$.

- The initial capital stock $k_0 > 0$ is given.
- A key feature in this problem is that it involves **choosing an infinite sequence of consumption or capital stock, one for each** t.

The Set of Alternatives

1. Since the utility function is strictly increasing, we can restrict attention to consumption streams $c = (c_t)\, t \geqslant 0$ satisfying

$$c_t = f(k_t) - k_{t+1}$$

The Set of Alternatives

1. Since the utility function is strictly increasing, we can restrict attention to consumption streams $c = (c_t)\, t \geqslant 0$ satisfying

$$c_t = f(k_t) - k_{t+1}$$

2. Therefore, the household's problem is

$$\max \sum_{t=0}^{\infty} \beta^t u\left(f(k_t) - k_{t+1}\right), \tag{SP}$$

among all capital stock streams $(k_{t+1})_t \geqslant 0$ satisfying $\forall t \geqslant 0, \quad 0 \leqslant k_{t+1} \leqslant f(k_t)$, where k_0 is given.

Solving SP

- We want to solve (SP) by identifying the optimal capital stock stream k^* and the corresponding maximum utility

$$V^*(k_0) = \sum_{t=0}^{\infty} \beta^t u\left(f\left(k_t^*\right) - k_{t+1}^*\right)$$

Solving SP

- We want to solve (SP) by identifying the optimal capital stock stream k^* and the corresponding maximum utility

$$V^*(k_0) = \sum_{t=0}^{\infty} \beta^t u\left(f(k_t^*) - k_{t+1}^*\right)$$

- The problem we need to solve is the following one

$$\underbrace{V(k_0) = \max_{\{k_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t u\left(f(k_t) - k_{t+1}\right)}$$

Solving SP

- We want to solve (SP) by identifying the optimal capital stock stream k^* and the corresponding maximum utility

$$V^*(k_0) = \sum_{t=0}^{\infty} \beta^t u\left(f\left(k_t^*\right) - k_{t+1}^*\right)$$

- The problem we need to solve is the following one

$$\underbrace{V(k_0) = \max_{\{k_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t u\left(f\left(k_t\right) - k_{t+1}\right)}_{V(k_0)=\max_{\{k_{t+1}\}_{t=0}^{\infty}}[u(f(k_0)-k_1)+\beta u(f(k_1)-k_2)+\beta^2 u(f(k_2)-k_3)...]}$$

Solving SP

- Observe

$$V(k_0) = \max_{\{k_{t+1}\}_{t=0}^{\infty}} \left[u(f(k_0) - k_1) + \beta u(f(k_1) - k_2) + \beta^2 u(f(k_2) - k_3) \ldots \right]$$

- Therefore,

$$V(k_0) = \max_{\{0 \leq k_{t+1} \leq f(k_t)\}_{t=0}^{\infty}} \left\{ u(f(k_0) - k_1) + \sum_{t=1}^{\infty} \beta^t u(f(k_t) - k_{t+1}) \right\}$$

Ideas Behind the Dynamic Programming

- Since k_t for $t \geq 2$ does not appear in $u(f(k_0 - k_1))$, we can rewrite the problem as

$$V(k_0) = \max_{k_1} \left\{ u(f(k_0) - k_1) + \beta \max_{\{k_{t+1}\}_{t=1}^{\infty}} \left[\sum_{t=1}^{\infty} \beta^{t-1} u(f(k_t) - k_{t+1}) \right] \right\}$$

Ideas Behind the Dynamic Programming

- Since k_t for $t \geq 2$ does not appear in $u(f(k_0 - k_1))$, we can rewrite the problem as

$$V(k_0) = \max_{k_1} \left\{ u(f(k_0) - k_1) + \beta \max_{\{k_{t+1}\}_{t=1}^{\infty}} \left[\sum_{t=1}^{\infty} \beta^{t-1} u(f(k_t) - k_{t+1}) \right] \right\}$$

- Observe:

$$\max_{\{k_{t+1}\}_{t=1}^{\infty}} \left[\sum_{t=1}^{\infty} \beta^{t-1} u(f(k_t) - k_{t+1}) \right] =$$

$$\max_{\{k_{t+1}\}_{t=1}^{\infty}} \left[u(f(k_1) - k_2) + \beta u(f(k_2) - k_3) + \ldots \right] = V(k_1)$$

Ideas Behind the Dynamic Programming

- Since k_t for $t \geq 2$ does not appear in $u(f(k_0 - k_1))$, we can rewrite the problem as

$$V(k_0) = \max_{k_1} \left\{ u(f(k_0) - k_1) + \beta \max_{\{k_{t+1}\}_{t=1}^{\infty}} \left[\sum_{t=1}^{\infty} \beta^{t-1} u(f(k_t) - k_{t+1}) \right] \right\}$$

- Observe:

$$\max_{\{k_{t+1}\}_{t=1}^{\infty}} \left[\sum_{t=1}^{\infty} \beta^{t-1} u(f(k_t) - k_{t+1}) \right] =$$

$$\max_{\{k_{t+1}\}_{t=1}^{\infty}} \left[u(f(k_1) - k_2) + \beta u(f(k_2) - k_3) + \ldots \right] = V(k_1)$$

- Therefore,

$$\boldsymbol{V(k_0) = \max_{0 \leq k_1 \leq f(k_0)} \{u(f(k_0) - k_1) + \beta V(k_1)\}}$$

Ideas Behind the Dynamic Programming

- But the dates $t = 0$ and $t = 1$ are arbitrary,so it is common to simply denote the capital stock today by k and the capital stock in the next period by $k^{'}$. Then

$$\boldsymbol{V(k) = \max_{0 \leq k' \leq f(k)} \left\{ u\left(f(k) - k'\right) + \beta V\left(k'\right) \right\},} \qquad \text{(BE)}$$

  where (') denotes for future variables.
- The main idea of dynamic programming was to transform an **inter-temporal program** into a **two-period program : today and the future**.
- If function $V(k)$ were known, we could use (BE) to define policy function $k^{'}$ that attains the maximum in (BE). Given the capital stock today, agents decide about capital stock tomorrow.

Ideas Behind the Dynamic Programming

- The main idea was to transform an inter-temporal program into a two period program: **today and the future.**
- Notice that with the Lagrangian approach we had introduced a set of Lagrange Multipliers .
- With the Bellman Equation we introduced a new function, the function V, which is **inter-temporal welfare**.

Recursive structure

- Notice

$$V(k) = \max_{0 \leq k' \leq f(k)} \left\{ u\left(f(k) - k'\right) + \beta V\left(k'\right) \right\}, \qquad \text{(BE)}$$

Recursive structure

- Notice

$$\boldsymbol{V(k) = \max_{0 \leq k' \leq f(k)} \left\{ u\left(f(k) - k'\right) + \beta V\left(k'\right) \right\},} \qquad \text{(BE)}$$

- This determines tomorrow's capital stock as a function of today's capital stock. Specifically,

$$k' = g(k) \in \arg \max_{0 \leq x \leq f(k)} \{u[f(k) - x] + \beta V(x)\}$$

Recursive structure

- Notice

$$\boldsymbol{V(k) = \max_{0 \leq k' \leq f(k)} \left\{ u\left(f(k) - k'\right) + \beta V\left(k'\right) \right\},} \qquad \text{(BE)}$$

- This determines tomorrow's capital stock as a function of today's capital stock. Specifically,

$$k' = g(k) \in \arg \max_{0 \leq x \leq f(k)} \{u[f(k) - x] + \beta V(x)\}$$

- This function g is know as a **policy function**.

Bellman Equation in Practice

- Suppose further that V is **concave** and **differentiable**. Then the **objective function is the sum of two concave functions and hence is also concave**. So the **optimal policy is characterized by the first order condition**:

$$u'(c) = \beta V\left(k'\right)$$

Bellman Equation in Practice

- Suppose further that V is **concave** and **differentiable**. Then the **objective function is the sum of two concave functions and hence is also concave**. So the **optimal policy is characterized by the first order condition**:

$$u'(c) = \beta V\left(k'\right)$$

- If V were known, we could solve for the policy function by finding the appropriate $k' = g(k)$ for each k such that;

$$u'[f(k) - g(k)] = \beta V'(g(k))$$

Bellman Equation in Practice

- With the policy function g, we can then construct the entire sequence of capital stocks $k_{t+1} = g(k_t)$ starting with the given initial condition k_0. Then we can back out consumption with $c_t = f(k_t) - g(k_t)$.

Bellman Equation in Practice

- With the policy function g, we can then construct the entire sequence of capital stocks $k_{t+1} = g(k_t)$ starting with the given initial condition k_0. Then we can back out consumption with $c_t = f(k_t) - g(k_t)$.
- Note that **today's capital stock is all that we need to know in order to choose optimal consumption**. In this regard, today's capital stock is the only **state variable** in this problem.

Bellman Equation in Practice

- As already discussed, an optimal policy g(k) is characterized by the FOC:

$$u'[f(k) - g(k)] = \beta V'(g(k))$$

Bellman Equation in Practice

- As already discussed, an optimal policy g(k) is characterized by the FOC:

$$u'[f(k) - g(k)] = \beta V'(g(k))$$

- We can make progress by using the **envelope theorem**.

$$V'(k) = u'[f(k) - g(k)]f'(k)$$

The above is called the envelope condition.

Bellman Equation in Practice

- As already discussed, an optimal policy g(k) is characterized by the FOC:

$$u'[f(k) - g(k)] = \beta V'(g(k))$$

- We can make progress by using the **envelope theorem**.

$$V'(k) = u'[f(k) - g(k)]f'(k)$$

The above is called the envelope condition.

- This can be written as:

$$V'(g(k)) = u'[f(g(k)) - g(g(k))]f'(g(k))$$

Bellman Equation in Practice

- Plugging this into the FOC gives:

$$u'[f(k) - g(k)] = \beta u'[f(g(k)) - g(g(k))]f'(g(k))$$

Bellman Equation in Practice

- Plugging this into the FOC gives:

$$u'[f(k) - g(k)] = \beta u'[f(g(k)) - g(g(k))]f'(g(k))$$

- The above is just the standard consumption Euler Equation, which is

$$u'\left(c_t\right) = \beta u'\left(c_{t+1}\right) f'\left(k_{t+1}\right)$$

Bellman Equation in Practice

- Plugging this into the FOC gives:

$$u'[f(k) - g(k)] = \beta u'[f(g(k)) - g(g(k))]f'(g(k))$$

- The above is just the standard consumption Euler Equation, which is

$$u'\left(c_t\right) = \beta u'\left(c_{t+1}\right) f'\left(k_{t+1}\right)$$

- We can solve for the steady state of the model without knowing either the value function or the policy function. In the steady state $k' = k$ so that $k^* = g(k^*)$. Therefore, we have:

$$1 = \beta f'\left(k^*\right),$$

which we can do without knowing either V or g.

Bellman Equation in Practice

1. **Take the F.O.C.**

$$u'(c) = \beta V'\left(k'\right)$$

Bellman Equation in Practice

1. **Take the F.O.C.**

$$u'(c) = \beta V'\left(k'\right)$$

2. **Consider the Envelop condition**

$$V'(k) = \alpha k^{\alpha-1} u'(c)$$

Bellman Equation in Practice

1. **Take the F.O.C.**

$$u'(c) = \beta V'\left(k'\right)$$

2. **Consider the Envelop condition**

$$V'(k) = \alpha k^{\alpha-1} u'(c)$$

3. **Consider the Envelop condition the next period**

$$V'\left(k'\right) = \alpha k'^{\alpha-1} u'\left(c'\right)$$

Bellman Equation in Practice

1. **Take the F.O.C.**

$$u'(c) = \beta V'\left(k'\right)$$

2. **Consider the Envelop condition**

$$V'(k) = \alpha k^{\alpha-1} u'(c)$$

3. **Consider the Envelop condition the next period**

$$V'\left(k'\right) = \alpha k'^{\alpha-1} u'\left(c'\right)$$

4. **Substitute for V' in the envelop condition, using the F.O.C.**

$$u'(c) = \beta\alpha k'^{\alpha-1} u'\left(c'\right)$$

and find the Euler Equation. Sequential representation:

$$u'\left(c_t\right) = \beta\alpha k_{t+1}^{\alpha-1} u'\left(c_{t+1}\right).$$

It is always involves these four steps:

Example 1: Saving choice problem

- **Sequential problem:**

$$\begin{array}{ll} \max_{\{c_t, a_{t+1}\}_{t=0}^{\infty}} & \sum_{t=0}^{\infty} \beta^t u(c_t) \\ a_{t+1} + c_t & = a_t (1 + r_t) + w_t \end{array}$$

Example 1: Saving choice problem

- **Sequential problem:**

$$\begin{array}{ll} \max_{\{c_t, a_{t+1}\}_{t=0}^{\infty}} & \sum_{t=0}^{\infty} \beta^t u\left(c_t\right) \\ a_{t+1} + c_t & = a_t\left(1 + r_t\right) + w_t \end{array}$$

- **Recursive problem:**

$$\begin{aligned} V(a) = & u(c) + \beta V\left(a'\right) \\ \text{s.t. } & a' + c = a(1 + r) + w \end{aligned}$$

Example 1: Saving choice problem

1. **Take the F.O.C.**
2. **Consider the Envelop condition**
3. **Consider the Envelop condition the next period**
4. **Substitute for V' in the envelop condition, using the F.O.C.**

Example 2: Internal habit formation

- Consider the problem: $u\left(c_t - hc_{t-1}\right)$.

Example 2: Internal habit formation

- Consider the problem: $u\left(c_t - hc_{t-1}\right)$.
- Let's write the Bellman equation recursively. Beginning-of-period wealth is a and past consumption is denoted c_{-1}. The Bellman equation is

$$V\left(a, c_{-1}\right) = \max_{a'} u\left(c - hc_{t-1}\right) + \beta V\left(a', c\right)$$
$$\text{s.t } a' + c = a(1+r) + w$$

Example 2: Internal Habit Formation

1. **Take the F.O.C.**
2. **Consider the Envelop condition**
3. **Consider the Envelop condition the next period**
4. **Substitute for V' in the envelop condition, using the F.O.C.**

Example 3: Stochastic Bellman Equations

$$V(a) = \max_{a'} u(c) + \beta \mathbb{E} V\left(a'\right)$$
$$a' + c = (1 + r)a + w$$

Calculations are the same as before with an additional $\mathbb{E}$ operator. Find:

$$u'(c) = \beta \mathbb{E}(1 + r)u'\left(c'\right)$$

Ways of Solving for the Value Function

We will discuss two ways of solving dynamic programming problems. The two ways aim to solve the functional equation for V:

1. **Guess and verify:** this method involves a guess of V and verification of the Bellman equation. Inevitably, it assumes the uniqueness of the solution to the equation.

Ways of Solving for the Value Function

We will discuss two ways of solving dynamic programming problems. The two ways aim to solve the functional equation for V:

1. **Guess and verify:** this method involves a guess of V and verification of the Bellman equation. Inevitably, it assumes the uniqueness of the solution to the equation.
2. **Value function iteration**

Guess and Verify

Consider an economy in which the representative consumer lives forever. There is a good in each period that can be consumed or saved as capital as well as labor. The consumer's utility function is:

$$\sum_{t=0}^{\infty} \beta^t \log c_t,$$

where $0 < \beta < 1$. The consumer is endowed with 1 unit of labor in each period and with k_0 units of capital in period 0. Feasible allocations satisfy:

$$c_t + k_{t+1} \leq \theta k_t^{\alpha} l_t^{1-\alpha},$$

where $\theta > 1$ and $0 < \alpha < 1$.

1. Write the appropriate **Bellman Equation.**
2. Guess that the solution takes the form $\boldsymbol{V(k) = a_0 + a_1 \log k}$. By using this guess solve the dynamic programming.

Value Function Iteration

1. Recall the Bellman equation:

$$V(k) = \max_{0 \leq k' \leq f(k)} \left\{ u\left[f(k) - k' \right] + \beta V\left(k' \right) \right\}.$$

Value Function Iteration

1. Recall the Bellman equation:

$$V(k) = \max_{0 \leq k' \leq f(k)} \left\{ u\left[f(k) - k' \right] + \beta V\left(k' \right) \right\}.$$

2. Notice the RHS of this equation defines an operator that maps a given function into a new one.

Value Function Iteration

1. Recall the Bellman equation:

$$V(k) = \max_{0 \leq k' \leq f(k)} \left\{ u\left[f(k) - k'\right] + \beta V\left(k'\right) \right\}.$$

2. Notice the RHS of this equation defines an operator that maps a given function into a new one.
3. This operator is often denoted T and we write TV for the new function. That is,

$$TV(k) \equiv \max_{0 \leq k' \leq f(k)} \left\{ u\left[f(k) - k'\right] + \beta V\left(k'\right) \right\}$$

Value Function Iteration

1. Recall the Bellman equation:

$$V(k) = \max_{0 \leq k' \leq f(k)} \left\{ u\left[f(k) - k'\right] + \beta V\left(k'\right) \right\}.$$

2. Notice the RHS of this equation defines an operator that maps a given function into a new one.
3. This operator is often denoted T and we write TV for the new function. That is,

$$TV(k) \equiv \max_{0 \leq k' \leq f(k)} \left\{ u\left[f(k) - k'\right] + \beta V\left(k'\right) \right\}$$

4. Then the Bellman equation implies $V = TV$. The value function V is in fact a fixed point of the operator equation.

Procedure to Value Function Iteration

1. Guess some initial value function, V_0.
2. Use T to construct a new value function $V_1 = TV_0$.

$$V_1(k) = TV_0(k) = \max_{0 \leq k' \leq f(k)} \left\{ u\left[f(k) - k'\right] + \beta V_0\left(k'\right) \right\}$$

If V_1 is sufficiently close to V_0 according to some metric, you have got the fixed point and stop here. Otherwise,

3. Keep iterating the value functions $V_{j+1} = TV_j = T^j V_0$. Under suitable regularity conditions, $T^j V_0$ converges to a unique V as $j \to \infty$. The limit is the fixed point and is independent of the initial guess V_0.

Conditions for the Fixed Point

- This iterative procedure works especially well when the operator T is a **contraction mapping.**
- A **contraction mapping** T has the property that for two functions V and W the distance between TV and TW is less than the distance between V and W . That is, using a contracting operator brings functions closer and closer.

Conditions for the Fixed Point

- This iterative procedure works especially well when the operator T is a **contraction mapping.**
- A **contraction mapping** T has the property that for two functions V and W the distance between TV and TW is less than the distance between V and W . That is, using a contracting operator brings functions closer and closer.
- **Sufficient conditions for the operator T to be a contraction mapping are:**
  1. the operator is **monotone**, so if two functions V , W satisfy $V \leq W$, then $TV \leq TW$; and
  2. the operator exhibits **discounting**, so if $a > 0$ is a constant and V is a function then $T(V + a) \leq TV + \beta a$ for some constant $\beta \in (0, 1)$
- **Is the Bellman equation for the growth model as contraction?**

Practical Dynamic Programming:

- Consider our growth model with $u(c) = \log c$ and $f(k) = Ak^{\alpha}$ and define the operator:

$$V_j(k) = \max_{k'} \left\{ \log \left(Ak^{\alpha} - k' \right) + \beta V_{j-1} \left(k' \right) \right] \Bigg\}$$

Practical Dynamic Programming:

- Consider our growth model with $u(c) = \log c$ and $f(k) = Ak^{\alpha}$ and define the operator:

$$V_j(k) = \max_{k'} \left\{ \log \left(Ak^{\alpha} - k' \right) + \beta V_{j-1} \left(k' \right) \right] \Bigg\}$$

- We can find the solution to this problem **analytically**:
  1. Start with $V_0(k) = 0$. Solve one period problem. $V_1(k) = \max_{k'} \ln \left(Ak^{\alpha} - k' \right)$. Solution,$k' = 0$ and $c = Ak^{\alpha}$. So, $V_1(k) = \ln A + \alpha \ln k$

Practical Dynamic Programming:

- Consider our growth model with $u(c) = \log c$ and $f(k) = Ak^{\alpha}$ and define the operator:

$$V_j(k) = \max_{k'} \left\{ \log \left(Ak^{\alpha} - k' \right) + \beta V_{j-1} \left(k' \right) \right] \Bigg\}$$

- We can find the solution to this problem **analytically**:
  1. Start with $V_0(k) = 0$. Solve one period problem. $V_1(k) = \max_{k'} \ln \left(Ak^{\alpha} - k' \right)$. Solution,$k' = 0$ and $c = Ak^{\alpha}$. So, $V_1(k) = \ln A + \alpha \ln k$
  2. $V_2(k) = \max_{k'} \left\{ \ln \left(Ak^{\alpha} - k' \right) + \beta \left(\ln A + \alpha \ln k' \right) \right\}$. Solution: $k' = \frac{\beta\alpha}{1+\beta\alpha} Ak^{\alpha}$, and $V_2(k) = (1+\beta) \ln A + \beta\alpha \ln \beta\alpha A - (1+\beta\alpha) \ln(1+\beta\alpha) + \alpha(1+\beta\alpha) \ln k$
  3. By keeping iterating until the limit we reach $k' = \alpha\beta Ak^{\alpha}$

Practical Dynamic Programming

The problem is not always that we can found an analytical solution, so we need a computer.

How to write a computer code:

- Discretize the state space: $k_1 < k_2 < \ldots < k_n; V_j(k) = \max_{k'}$

$$\begin{bmatrix} \log\left(Ak_1^{\alpha} - k_1'\right) + \beta V_{j-1}\left(k_1'\right) & \log\left(Ak_1^{\alpha} - k_2'\right) + \beta V_{j-1}\left(k_2'\right) & \ldots & \log\left(Ak_1^{\alpha} - k_n'\right) + \beta V_{j-1}\left(k_n'\right) \\ \log\left(Ak_2^{\alpha} - k_1'\right) + \beta V_{j-1}\left(k_1'\right) & \log\left(Ak_2^{\alpha} - k_2'\right) + \beta V_{j-1}\left(k_2'\right) & \ldots & \log\left(Ak_2^{\alpha} - k_n'\right) + \beta V_{j-1}\left(k_n'\right) \\ \log\left(Ak_n^{\alpha} - k_1'\right) + \beta V_{j-1}\left(k_1'\right) & \log\left(Ak_n^{\alpha} - k_2'\right) + \beta V_{j-1}\left(k_2'\right) & \ldots & \log\left(Ak_n^{\alpha} - k_n'\right) + \beta V_{j-1}\left(k_n'\right) \end{bmatrix}$$

Practical Dynamic Programming

The problem is not always that we can found an analytical solution, so we need a computer.

How to write a computer code:

- Discretize the state space: $k_1 < k_2 < \ldots < k_n; V_j(k) = \max_{k'}$

$$\begin{bmatrix} \log\left(Ak_1^\alpha - k_1'\right) + \beta V_{j-1}\left(k_1'\right) & \log\left(Ak_1^\alpha - k_2'\right) + \beta V_{j-1}\left(k_2'\right) & \ldots & \log\left(Ak_1^\alpha - k_n'\right) + \beta V_{j-1}\left(k_n'\right) \\ \log\left(Ak_2^\alpha - k_1'\right) + \beta V_{j-1}\left(k_1'\right) & \log\left(Ak_2^\alpha - k_2'\right) + \beta V_{j-1}\left(k_2'\right) & \ldots & \log\left(Ak_2^\alpha - k_n'\right) + \beta V_{j-1}\left(k_n'\right) \\ \log\left(Ak_n^\alpha - k_1'\right) + \beta V_{j-1}\left(k_1'\right) & \log\left(Ak_n^\alpha - k_2'\right) + \beta V_{j-1}\left(k_2'\right) & \ldots & \log\left(Ak_n^\alpha - k_n'\right) + \beta V_{j-1}\left(k_n'\right) \end{bmatrix}$$

- Start with $V_0(k) = 0$.
- Find $V_1(k)$.Check if $|V_1(k) - V_0(k)| < \varepsilon$;
- If not ,find $V_2(k)$. Keep iterating until $|V_n(k) - V_{n-1}(k)| < \varepsilon$.

How to solve the iterations process using computer

The goal is to translate the **Dynamic Programming into a computer code**:

1. Set parameters α, β and A and compute the steady state $\bar{k} = [\alpha\beta A]^{\frac{1}{1-\alpha}}$.
2. Grid the vector $k : k \in [k_1 < k_2 < \ldots < k_m]$ with $k_1 > 0$.

How to solve the iterations process using computer

The goal is to translate the **Dynamic Programming into a computer code**:

1. Set parameters α, β and A and compute the steady state $\bar{k} = [\alpha\beta A]^{\frac{1}{1-\alpha}}$.
2. Grid the vector $k : k \in [k_1 < k_2 < \ldots < k_m]$ with $k_1 > 0$.
3. Given the grid for k,compute the grid for $c = Ak^{\alpha} - k$. Matrix, with dimensions mXm:

$$\begin{bmatrix} c_{11} & \cdots & c_{1j} & \cdots & c_{1m} \\ \vdots & & & & \vdots \\ c_{i1} & & c_{ij} & & c_{im} \\ \vdots & & & & \vdots \\ c_{m1} & \cdots & c_{mj} & \cdots & c_{mm} \end{bmatrix}$$

The rows represent k and the columns k', such that:

$$c_{ij} = Ak_i^{\alpha} - k_j^{'}$$

Solving with iterations

To build the previous matrix we can do a loop using any computer language:

Given the consumption matrix c, create the utility matrix, U, of dimensions mXm.

Value Function Iteration

1. Initialize the value function, V, by defining a vector of zeros, with dimensions $mX1$.
2. Compute the first period value function and call it TV as follows:

$$\begin{bmatrix} TV(k_1) \\ TV(k_2) \\ \vdots \\ TV(k_m) \end{bmatrix} = \left(\max_{k'} \left\{ \begin{bmatrix} U_{11} & U_{12} & & U_{1m} \\ U_{21} & U_{22} & & \\ & & \ddots & \\ U_{m1} & U_{m2} & & U_{mm} \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1] \end{bmatrix} \right\} \right)$$

Given a k_i (i.e. k) we want to find k_j (i.e. k') that maximizes the expression in the curly brackets. I.e. we choose the column number j, that maximizes the expression for row i. In the code this will look like: V = zeros(m, 1);
o = ones(1, m);
TV = (max(U + beta * V * o));

Value Function Iteration

3. Create a **loop** to iterate on the value function:
   - Set V to be the TV computed from the previous step.
   - Compute a matrix that contains all possible values for $\log c + \beta V$, then of all the elements of the matrix choose the max and call it TV (like before).
   - Then compare your old value function V with the new value function TV for the stopping rule.
   - Stop the loop using a stopping criterion and call the final V^*.

while looping you may wanna to show the number of iterations until convergence

How to find the Policy Function?

1. Previously we found the value for k' that solves

$$\boldsymbol{TV} = \max\left(U + \beta V^*\left(k'\right)\right)$$

2. For each column we can save the index where we found the maximum.
3. To find the policy function, use the index and the grid of capital to find the value for k'.

Value Function Iteration

TO MAKE THINGS CLEAR LET'S GIVE A LOOK IN A CODE THAT SOLVES THE VALUE FUNCTION ITERATION.

Now observe the following facts:

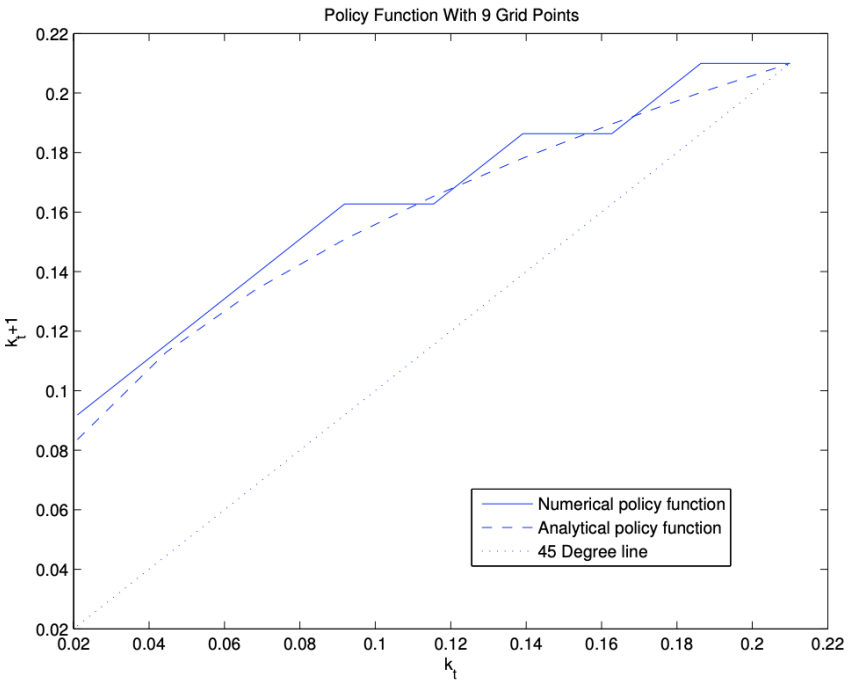

Figure 1: 9 Grid Points

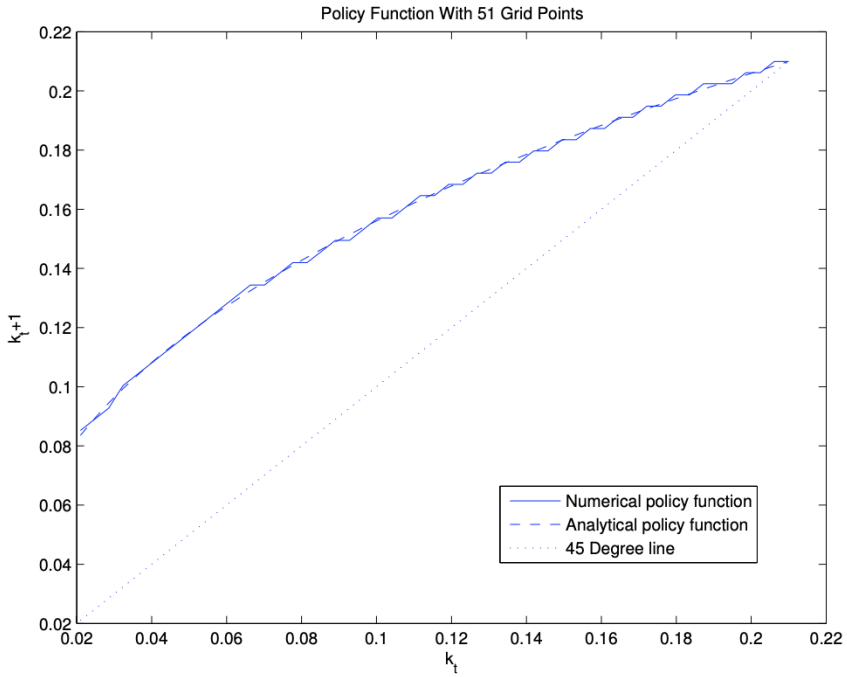

Figure 2: 51 Grid Points

Stochastic Dynamic Programming

Suppose consumer maximizes

$$V\left(k_0, z_0\right) = \max_{\left\{c_t, k_{t+1}\right\}_{t=0}^{\infty}} = E_0\left[\sum_{t=0}^{\infty} \beta^t u\left(c_t\right)\right], 0 < \beta < 1$$

s.t.

$$k_{t+1} + c_t \leq z_t f\left(k_t\right)$$

$$z_{t+1} = \rho z_t + \epsilon_{t+1} \quad , \rho \in (0,1), \epsilon \sim F\left(0, \sigma_\epsilon^2\right)$$

$$k_0 > 0 \quad \text{given and ,}$$

$$c_t \geq 0$$

- Note: u(.) and f(.) satisfy standard properties.
- At period t agents know the realization of the shock z_t.

Optimal Growth Problem with Uncertainty

- We can show that:

$$V\left(k_0, z_0\right) = \max_{0 \leq k_1 \leq z_0 f(k_0)} \left\{u\left(z_0 f\left(k_0\right) - k_1\right) + \beta E_0\left[V\left(k_1, z_1\right)\right]\right\}$$

Optimal Growth Problem with Uncertainty

- We can show that:

$$V\left(k_0, z_0\right) = \max_{0 \leq k_1 \leq z_0 f(k_0)} \left\{u\left(z_0 f\left(k_0\right) - k_1\right) + \beta E_0\left[V\left(k_1, z_1\right)\right]\right\}$$

- Since the problem repeats itself in every period we have:

$$V(k, z) = \max_{0 \leq k' \leq z f(k)} \left\{u\left(z f(k) - k'\right) + \beta E\left[V\left(k', z'\right)\right]\right\}$$

Optimal Growth Problem with Uncertainty

- We can show that:

$$V\left(k_0, z_0\right) = \max_{0 \leq k_1 \leq z_0 f(k_0)} \left\{u\left(z_0 f\left(k_0\right) - k_1\right) + \beta E_0\left[V\left(k_1, z_1\right)\right]\right\}$$

- Since the problem repeats itself in every period we have:

$$V(k, z) = \max_{0 \leq k' \leq z f(k)} \left\{u\left(z f(k) - k'\right) + \beta E\left[V\left(k', z'\right)\right]\right\}$$

- Notice that z is not our choice variable, since it is exogenous. So, we again have the **stochastic version of the Euler Equation**:

$$u'\left(c_t\right) = \beta E\left[u'\left(c_{t+1}\right)\left(1 + z_{t+1} f'\left(k_{t+1}\right) - \delta\right)\right]$$

Conclusions

- Bellman equations : powerful tool for dynamic economies.
- Structure like two-period economies.
- In complicated environment, the issue is to find the correct state space.
- The interpretation is then simple. It can be computed in more complex environment.