Dynamic Optimization

Jesús Bueren

EUI

Dynamic Optimization

- In this chapter we are going to characterize solutions to dynamic optimization problems
- In order to solve them, we are going to introduce discrete dynamic programming.
- Along our way, we are going to revise some mathematical concepts covered by Villanacci.
- References: *The PhD Macro Book* (Ch 4), Acemoglu (Ch 6), and SLP (Ch 4).

Motivating the Recursive Formulation

A Cake Eating Problem

- We will go over a very simple dynamic optimization problem.
- Suppose that you are presented with a cake of size W_1.
- At each point in time $t = 1, 2, \ldots, T$, you can eat some of the cake but must save the rest.
- Let c_t be your consumption at time t and $u(c_t)$ represent the flow of utility.
- u twice differentiable, strictly increasing, strictly concave, $\lim_{c \to 0} u'(c) = \infty$.
- Discount factor: $0 < \beta < 1$

The Sequential Formulation

A Cake Eating Problem

- The agent is solving:

$$\max_{\{c_t, W_{t+1}\}_{t=0}^{T}} \sum_{t=0}^{T} \beta^t u(c_t)$$
$$\text{s.t. } c_t + W_{t+1} = W_t \ \forall t$$
$$W_{T+1} \geq 0$$

- The Lagrangian associated to this problem is given by:

$$\mathcal{L} = \sum_{t=0}^{T} \beta^t u(c_t) + \sum_{t=0}^{T} \lambda_t (W_t - c_t - W_{t+1}) + \phi W_{T+1}$$

The Sequential Formulation

A Cake Eating Problem

- FOCs:

$$
\begin{aligned}
\beta^t u_c(c_t) &= \lambda_t \\
\lambda_t &= \lambda_{t+1} \\
\lambda_T &= \phi \\
\phi \geq 0 \text{ with } \phi W_{T+1} &= 0 \Rightarrow \beta^T u_c(c_t) W_{T+1} = 0
\end{aligned}
$$

$$
\begin{aligned}
& u'(c_t) = \beta u'(c_{t+1}) \; \forall t \in [0, T-1] \\
& W_{T+1} = 0
\end{aligned}
$$

- With the set of T intertemporal equations (euler equations), an initial condition and a terminal condition

The Recursive Formulation

A Cake Eating Problem

- In order to solve finite-horizon dynamic programming problems, we are going to proceed by backwards induction.
- For $t = T$, given the properties of u and the constraint, the optimal solution is given by:

$$c_T = W_T$$
$$u(c_T) = u(W_T)$$

The Recursive Formulation

A Cake Eating Problem

- We define the **value function** at time T for the problem at time T as:

$$V_T(W_T) = \max_{c_T} u(c_T)$$
$$c_T + W_{T+1} = W_T$$

- The optimal cake-saving decision is thus:

$$g_T(W_T) = 0$$

and the value function is given by:

$$V_T(W_T) = u(W_T)$$

The Recursive Formulation

A Cake Eating Problem

- Now let's go to $t = T - 1$ given that we have solved the problem for $t = T$ and define V_{T-1}.

$$V_{T-1}(W_{T-1}) = \max_{c_{T-1}, c_T, W_T, W_{T+1}} u(c_{T-1}) + \beta u(c_T)$$
$$s.t. \;\; c_{T-1} + W_T = W_{T-1}$$
$$c_T + W_{T+1} = W_T$$

- Given that we already we know what is optimal to do in the next period, we can simplify the problem at $T - 1$ as:

$$V_{T-1}(W_{T-1}) = \max_{c_{T-1}, W_T} u(c_{T-1}) + \beta V_T(W_T)$$
$$s.t. \;\; c_{T-1} + W_T = W_{T-1}$$

The Recursive Formulation

A Cake Eating Problem

- Le's write the optimality conditions as:

$$u'(c_{T-1}) = \beta V_T'(W_T)$$
$$u'(c_{T-1}) = \beta u_T'(W_T)$$

- The solution coincides with the sequential formulation in the last period.

- We are in good track but what about previous periods?

The Recursive Formulation

A Cake Eating Problem

- Since it's going to be useful let's first derive the value of $V'_{T-1}(W_{T-1})$ given the optimal cake saving decision $g_{T-1}(W_{T-1})$ obtained from the previous FOC.

$$
\begin{aligned}
V_{T-1}(W_{T-1}) =& u(W_{T-1} - g_{T-1}(W_{T-1})) + \beta V_T(g_{T-1}(W_{T-1})) \\
\frac{\partial V_{T-1}(W_{T-1})}{\partial W_{T-1}} =& u_c(c_{T-1}) - u_c(c_{T-1})\frac{\partial g_{T-1}(W_{T-1})}{\partial W_{T-1}} + \\
& \beta \frac{\partial g_{T-1}(W_{T-1})}{\partial W_{T-1}} \frac{V_T(W_T)}{\partial W_T} \\
\frac{\partial V_{T-1}(W_{T-1})}{\partial W_{T-1}} =& u_c(c_{T-1}) + \frac{\partial g_{T-1}(W_{T-1})}{\partial W_{T-1}} \left(\beta \frac{V_T(W_T)}{\partial W_T} - u_c(c_{T-1})\right) \\
\frac{\partial V_{T-1}(W_{T-1})}{\partial W_{T-1}} =& u_c(c_{T-1})
\end{aligned}
$$

The Recursive Formulation

A Cake Eating Problem

- At $T-2$ the problem can be written as:

$$V_{T-2}(W_{T-2}) = \max_{c_{T-2}, W_{T-1}} u(c_{T-2}) + \beta V_{T-1}(W_{T-1})$$
$$\text{s.t. } c_{T-2} + W_{T-1} = W_{T-2}$$

- With FOCs:

$$u_c(c_{T-2}) = \beta \frac{\partial V_{T-1}(W_{T-1})}{\partial W_{T-1}} = \beta u_c(c_{T-1})$$

Practical Dynamic Programming

Finite Horizon

- Define a discretized grid of cake: $W \in \{W^1, \ldots, W^{nkk}\}$.
- Define $V_T(W_T)$ for each W_T^i in the cake grid: $V_T(W_T^i) = u(W_T^i)$ and $g_T(W_T^i) = 0 \; \forall \; i \in \{1, \ldots, nkk\}$
- Go to the previous period. We want to find $g_{T-1}(W_{T-1})$
- Grid search: For each W_{T-1}^i, $i \in \{1, \ldots, nkk\}$, the agent has i possible cake saving decisions W_T^j where $j \in \{1, \ldots, i\}$.
- Compute the value for each j:

$$V_{T-1}(W_{T-1}^i, W_{T-1}^j) = u(W_{T-1}^i - W_T^j) + \beta V(W_j)$$

  and select the W_{T-1}^j which achieves the highest utility: j^*, set $g_{T-1}(W_{T-1}^i) = W_T^{j^*}$ and $V_{T-1}(W_{T-1}^i) = V_{T-1}(W_{T-1}^i, W_{T-1}^{j^*})$
- Move to period $T - 2$

Infinite Horizon

A Cake Eating Problem

- Suppose for the cake-eating problem, we allow the horizon to go to infinity.
- The main advantage of an infinite horizon is that the agent problem becomes stationary: the maximization problem at date t is exactly the same as in period $t+1$
- Unlike in finite horizon case, we don't have a terminal condition in the cake eating problem we will thus need to impose a transversality condition:

$$\lim_{t\to\infty} \beta^t u_c(c_t) W_{t+1} = 0$$

if discounted marginal utility is positive, the amount of cake needs to go to zero to rule out over-accumulation

Infinite Horizon

A Cake Eating Problem

- One can consider solving the infinite horizon sequence given by:

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

$$\text{s.t. } c_t + W_{t+1} = W_t + y \ \forall \ t$$

$$\lim_{t\to\infty} \beta^t u_c(c_t) W_{t+1} = 0$$

- Written in recursive form:

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

$$\text{s.t. } c_t + W_{t+1} = W_t + y$$

$$\lim_{t\to\infty} \beta^t V(W_t) = 0 \tag{2}$$

The transversality condition (2) is frequently avoided because assuming V being bounded, its is satisfied for $\beta < 1$.

Infinite Horizon

A Cake Eating Problem

- Equation (1) is referred as the Bellman equation.
- It is a functional equation: the unknown represents as function.
- By FOCs:

$$u_c(c_t) = \beta \frac{\partial V(W_{t+1})}{\partial W_{t+1}} \tag{3}$$

- Let's define $g(W)$ the optimal savings function associated with equation (1):

$$\begin{aligned} g(W_t) =& arg \max_{W_{t+1}} u(W_t + y - W_{t+1}) + \beta V(W_{t+1}) \\ V(W_t) =& u(W_t + y - g(W_t)) + \beta V(g(W_t)) \end{aligned}$$

Infinite Horizon

A Cake Eating Problem

- Provided that g is differentiable we can now compute:

$$\frac{\partial V(W_t)}{\partial W_t} = u_c(c_t) + \frac{\partial g(W_t)}{\partial W_t}\left(\beta \frac{\partial V(W_{t+1})}{\partial W_{t+1}} - u_c(c_t)\right)$$
$$\frac{\partial V(W_t)}{\partial W_t} = u_c(c_t) \Rightarrow \frac{\partial V(W_{t+1})}{\partial W_{t+1}} = u_c(c_{t+1})$$

- Then we can write equation (3) as:

$$u_c(c_t) = \beta u_c(c_{t+1})$$

Infinite Horizon

A Cake Eating Problem

- Under what conditions V exists? Is it unique?
- How to find V in the infinite horizon case?
- Is g a function or a correspondence? Is it differentiable?

The Dynamic Programming Approach

- Buiding on the intuition gained from the cake eating problem, we now consider a more formal treatment of the dynamic programming approach to answer the previous questions.
- We begin with the nonstochastic case and then add uncertainty to the formulation.

The Dynamic Programming Approach

- Consider the infinite horizon optimization problem of an agent with payoff function $\tilde{\sigma}(s_t, c_t)$.
- state vector: s_t; control vector: c_t.
- Transition equation: $s_{t+1} = \tilde{\tau}(s_t, c_t)$.
- The state summarizes all the information from the past that is needed to make a forward-looking decision.
- $s \in \mathcal{S}$ and $c \in \mathcal{C}(s)$.
- Let β be the discount factor and assume $0 < \beta < 1$.

The Dynamic Programming Approach

- The sequential problem can be written as:

$$\max_{\{c_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t \tilde{\sigma}(s_t, c_t)$$
$$s.t.\ s_{t+1} = \tilde{\tau}(s_t, c_t)$$
$$c_t \in \tilde{\mathcal{C}}(s_t)$$

- We can rewrite the problem as Henriette and Matteo prefer by imposing the law of motion of the state:

$$V^*(s_0) = \max_{\{s_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t \sigma(s_t, s_{t+1})$$
$$s.t.\ s_{t+1} \in \mathcal{C}(s_t),$$

Where V^* denotes the highest possible value the the objective function can reach

- The basic idea of dynamic programming is to turn the sequential problem into a functional equation:

$$V(s) = \max_{s' \in \mathcal{C}(s)} \sigma(s, s') + \beta V(s) \tag{4}$$

- Instead of choosing a sequence $\{s_t\}_{t=0}^{\infty}$, we choose a policy, which determines the control s' as a function of the state s.

- Given that V appears both in both sides of the equation 4 and thus it is defined recursively.

- Equation 4 is also referred as the Bellman equation after Richard Bellman, who was the first to introduce the dynamic programming formulation.

- A solution to the functional equation is thus a fixed point.

Math Review

Brouwer's Fixed Point Theorem

- Let $\mathcal{F}$ be a nonempty compact (closed and bounded) convex set.
- Let T be a continuous function that maps each point $x \in \mathcal{F}$ to itself.
- Then T has a fixed point $x^* \in \mathcal{F}$ such that $T(x^*) = x^*$
- More questions:
  - To which set does V belong to?
  - Does the operator defined in the functional equation map each element of that set to itself?
  - Is the fixed point unique?

Math Review

What is a Contraction Mapping?

- Let $(\mathcal{M}, d)$ be a metric space where $\mathcal{M}$ is a set and d is a metric.

  A metric space is a set and a function such that for all $x, y, z \in S$:

  1. $d(x, y) \geq 0$, with equality iff $x = y$
  2. $d(x, y) = d(y, x)$
  3. $d(x, y) \leq d(x, z) + d(z, y)$

- Let $T : \mathcal{M} \rightarrow \mathcal{M}$ be an function mapping $\mathcal{M}$ into itself.

- If there exists a $\beta \in (0, 1)$ such that,

$$d(Tz_1, Tz_2) \leq \beta d(z_1, z_2) \ \ \forall \ z_1, z_2 \in S$$

  then T is a **contraction mapping** with modulus β.

- In other words, a contraction mapping brings elements of the space $\mathcal{M}$ uniformly closer to one another.

Math Review

Contraction Mapping Theorem

Let $(\mathcal{M}, d)$ be a complete metric space and suppose $T : \mathcal{M} \to \mathcal{M}$ is a contraction mapping.

A metric space is complete if every Cauchy sequence is a convergent sequence.

- A sequence $\{x_n\}_{n=0}^{\infty}$ is a Cauchy sequence if for all $\epsilon > 0$ there exists an $N \in \mathbb{N}$ such that for all $l, n > N$, $d(x_l, x_n) < \epsilon$
- A sequence $\{x_n\}_{n=0}^{\infty}$ is a convergent sequence to $\underline{x_0 \in \mathcal{M}}$ if for all $\epsilon > 0$, there exist here exists an $N \in \mathbb{N}$ such that for any $n > N$, $d(x_n, x_0) < \epsilon$

- Then, T has a **unique** fixed point $\hat{z}$ and for any $z_0 \in \mathcal{M}$, and any $n \in \mathbb{N}$ we have $d(T^n z_0, \hat{z}) \leq \beta^n d(z_0, \hat{z})$.
- That is there exists a unique $\hat{z} \in \mathcal{M}$ such that

$$T\hat{z} = \hat{z}$$

and regardless of the starting guess z_0, the sequence $\{T^n z_0\}_{n=0}^{\infty}$ converges to $\hat{z}$.

Match Review

Blackwell's Sufficient Conditions for a Contraction

- Let $s \in \mathcal{S}$ and $(\mathcal{M}, d)$ be the metric space where $\mathcal{M}$ is the set of bounded function equipped with the sup norm.
- Let $T : \mathcal{M}(s) \to \mathcal{M}(s)$ satisfying:
  1. Monotonicity: If $W(s) \geq Q(s)$, for all $s \in \mathcal{S}$, then $TW(s) \geq TQ(s)$.
  2. Discounting: for any constant k there exists $\tilde{\beta} \in [0, 1)$ such that $T(W + k)(s) \leq T(W)(s) + \beta k$.
- Then T is a contraction.

Recursive Formulation

- In order to apply the Blackwell sufficient conditions, we need V to belong to the set of bounded functions.
- For this to be true, we need some assumption on the primitive objects.

Recursive Formulation

- $\sigma(s_t, s_{t+1})$ needs to be bounded so that it does not yield infinite returns: we cannot compare two choices of s_{t+1} that deliver infinite value.

- With $\beta \in (0, 1)$ and bounded σ, the V will be bounded for the problems that we will see in this course.

- Problems might arise in models of growth: you would need growth in the return function to be "smaller" than the rate of discounting such that discounted returns are bounded.

- This assumption will allow us to define the set of V: the set of continuous bounded functions.

- Equipped with the supremum norm forms a complete metric space.

Recursive Formulation

- If σ is continuous and $\mathcal{C}$ is convex, nonempty and compact (closed and bounded).

  $\Rightarrow$ Unique value function satisfying the functional equation and therefore it is possible to find $V(x)$ by an iterative process

  1. Select any initial value $V_0(s)$ $\forall x \in \mathcal{S}$.
  2. Define a sequence of functions:

  $$V_n(x) = \max_{s' \in \mathcal{C}(s)} \sigma(s, s') + \beta V_{n-1}(s)$$

  3. The sequence $\{V_0, V_1, \ldots, V_n\}_{n=0}^{\infty}$ converges to V

Recursive Formulation

- Even if V is unique it could be that the policy associated could be a correspondence unless we put further restrictions of σ and $\mathcal{C}$:

  1. $\sigma(s, s')$: strictly concave, continuous, and differentiable.
  2. $\mathcal{C}(s)$: convex

  $\Rightarrow$ We have a continuous and differentiable policy function

- The Enveloppe theorem holds:

$$\frac{\partial v(s)}{\partial s} = \frac{\partial \sigma(s, s')}{\partial s}$$

Is T a Contraction?

Blackwell's Sufficient Conditions: Monotonicity

- Let $Q(s) \leq W(s)\ \forall s \in \mathcal{S}$.
- Let $\phi_Q(s)$ be the policy function obtained from:

$$\phi_Q(s) = arg \max_{s'\Gamma(s)} \sigma(s, s') + \beta Q(s')$$

- Then,

$$\begin{aligned} TQ(s) &= \sigma(s, \phi_Q(s)) + \beta Q(\phi_Q(s)) \leq \sigma(s, \phi_Q(s)) + \beta W(\phi_Q(s)) \\ &= \max_{s'\Gamma(s)} \sigma(s, s') + \beta W(s') = TW(s) \end{aligned}$$

Is T a Contraction?

Blackwell's Sufficient Conditions: Discounting

- This property is easy to verify in the dynamic programming problem:

$$
\begin{aligned}
T(W+k)(s) &= \max_{s' \in \Gamma(s)} \sigma(s, s') + \beta(W(s') + k) \\
&= TW(s) + \beta k
\end{aligned}
$$

The Neoclassical Growth Model

- In 1928 Frank Ramsey, a young mathematician, posed the problem:

  "*How much of its income should a nation save?*"

  and developed a dynamic model to answer this question.

- Economic agent (a social planner) producing output from labor and capital who must decide how to split production between consumption and capital accumulation.

The Neoclassical Growth Model

The Planner's Problem

- Time is discrete.
- Production is given by $y_t = f(k_t)$ where k_t is capital. f satisfies inada conditions.
- The planner's problem is given by:

$$\max_{\{c_t\}_{t=0}^{\infty}\{k_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t u(c_t)$$
$$s.t. \qquad c_t + k_{t+1} \leq f(k_t) + (1-\delta)k_t \ \forall t$$

The Neoclassical Growth Model

The Planner's Problem

- Now let's write the planner's problem in recursive form:

$$V(k) = \max_{k' \in [0, f(k)+(1-\delta)k]} u\Big(f(k) + (1-\delta)k - k'\Big) + \beta V(k')$$

- The solution is characterized by:

$$u_c(c) = \beta \frac{\partial V(k')}{\partial k'} = \beta \Big(\frac{\partial f(k')}{\partial k'} + 1 - \delta \Big) u_c(c')$$

The Neoclassical Growth Model

The Planner's Problem

- In the one sector growth model we define the operator T to be:

$$TV(k) = \max_{k' \in [0, f(k)+(1-\delta)k]} \{u(f(k) + (1-\delta)k - k') + \beta V(k')\}$$

- We want to argue that this operator has as unique fixed point using the contraction mapping theorem.
- Thus we are going to do it using Blackwell's sufficient conditions.

The Neoclassical Growth Model

The Planner's Problem

- Monotonicity:

$$\text{Let } \phi_Q(k) = arg \max_{k' \in \Gamma(k)} u(f(k) + (1-\delta)k - k' + \beta Q(k')$$

$$\text{if } Q(k) \leq W(k), \text{ for all } k$$

$$\text{then } TQ(k) = u(f(k) + (1-\delta)k - \phi_Q(k)) + \beta V(\phi_Q(k))\}$$

$$\leq u(f(k) + (1-\delta)k - \phi_Q(k)) + \beta W(\phi_Q(k))\} \leq TW(k)$$

- Discounting:

$$\begin{aligned} T(V+a)(k) = & \max_{k' \in \Gamma(k)} \{u(f(k) + (1-\delta)k - k') + \beta(V(k') + a)\} \\ = & TV(k) + \beta a \end{aligned}$$

Solving the problem numerically

Discrete State Methods

- There exists a variety of numerical methods to solve dynamic programming problems like the Ramsey problem (projection, perturbation, parameterized expectation).
- The need of numerical methods arises from the fact that dynamic programming problems generally do not have tractable closed form solutions.
- Because of their simplicity, we are going to focus on discrete-state space methods.

Solving the problem numerically

Discrete State Methods

- In this case, the value function is a finite dimensional object.
- For instance, if the state space is one dimensional and has elements $\mathcal{S} = s_1, s_1, \ldots, s_n$, the value function is just a vector of n elements where each element gives the value attained by the optimal policy if the initial state of the system is $s_n \in \mathcal{S}$.
- Drawback: curse of dimensionality.
  - If the the value function of an m-dimensional problem with n different points in each dimension is an array of n^m different elements and the computation time needed to search this array may be prohibitively high.

Solving the problem numerically

Value Function Iteration

- Given that Blackwell sufficient conditions hold, the can use the following pseudo-code for finding the value function:

  1. Make a guess for V_0 for all values of capital.
  2. Apply the operator T and recover $V_1 = TV_0$
  3. Compute distance between V_0 and V_1.

     3.1 If V_1 and V_0 are close enough, stop.

     3.2 Otherwise set $V_0 = V_1$ and go back to 2.

- Once the algorithm has converged, you can simulate the path for capital of an economy with an initial capital endowment.

Solving the problem numerically

Value Function Iteration

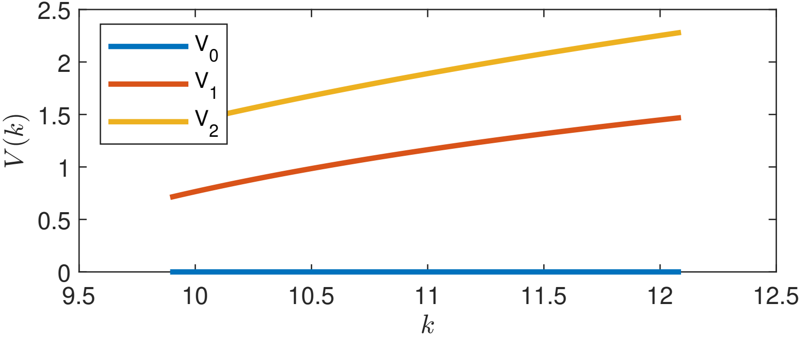
- Define a grid with N points of capital between $[\underline{k}, \bar{k}]$ around the steady state level of capital.
- Define a value of V_0 for all the points in this grid. Let's say $V_0 = 0$ for all k.
- Given this V_0, we can generate a vector for each level of capital k_i which elements are:

$$\begin{bmatrix} u(f(k_i) + (1-\delta)k_i - k_1) + \beta V_0(k_1) \\ u(f(k_i) + (1-\delta)k_i - k_2) + \beta V_0(k_2) \\ \vdots \\ u(f(k_i) + (1-\delta)k_i - k_N) + \beta V_0(k_N) \end{bmatrix}$$

Solving the problem numerically

Value Function Iteration

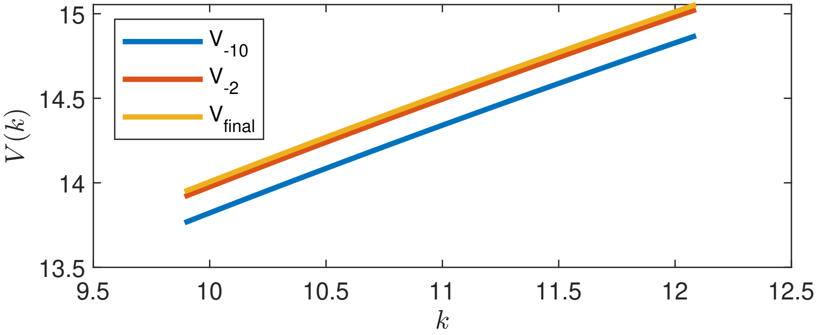
- $TV_0(k)$ can be approximated by the maximum value of the elements of this vector.
- Looping through all values of $i \in [0, N]$ we will recover V_1.
- Given V_1 we can recover V_2.



Solving the problem numerically

Value Function Iteration

- We iterate until V_g and V_{g+1} are sufficiently close



Solving the problem numerically

Value Function Iteration

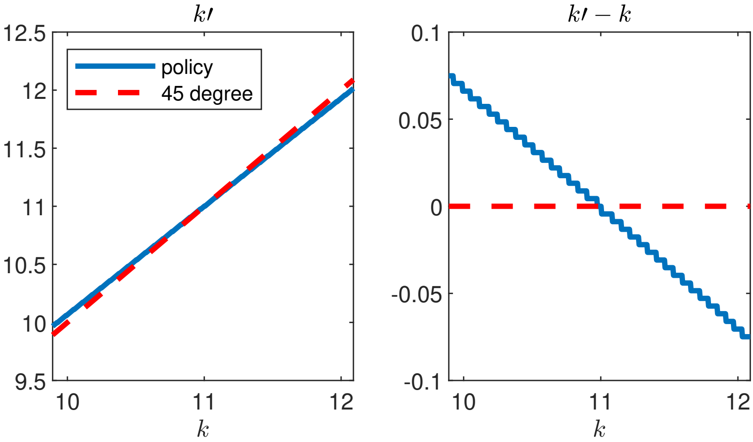
- Now that we have V, we need to recover $\pi(k)$ which is given by:

$$\pi(k) = \arg\max_{k'} \{u(k, k') + \beta V(k')\}$$

Solving the problem numerically

What do we aim for?

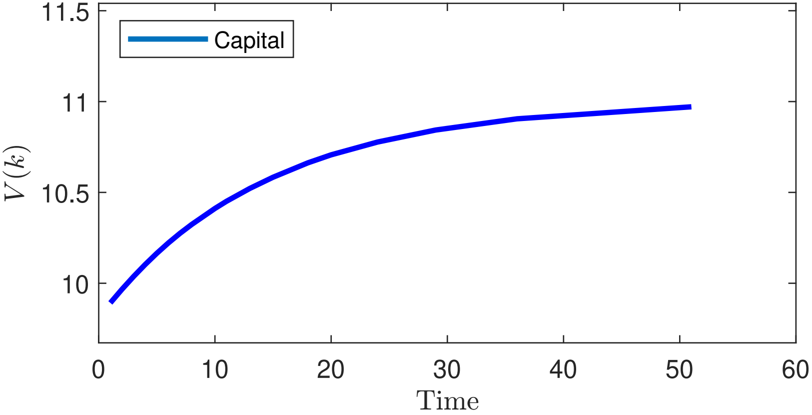
- A policy function:



Solving the problem numerically

Evolution of capital

- Given $\pi(k)$ we can simulate the transition towards the steady state for any $k_0 \in [\underline{k}, \bar{k}]$.



Competitive Equilibrium

Arrow-Debreu Equilibrium

- We will now define three different ways of decentrilizing the non-stochastic one-sector growth model.
- A representative household who owns the capital and labor, which she rents it to firms in exchange of an interest rate r_t and wage w_t in units of consumption good a time t per unit of capital rented and labor used.
- There is a market at time 0 where agents can buy and sell goods of different time periods.
- We assume that all contracts that are agreed at time 0 are honored.
- There is a price p_t for a consumption good at time t relative to consumption goods at $t = 0$ (Normalize: $p_{t_0} = 1$).

Competitive Equilibrium

Arrow-Debreu Equilibrium

- Consumer's problem:

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

$$\text{s.t.} \quad \sum_{t=0}^{\infty} p_t(c_t + k_{t+1}) = \sum_{t=0}^{\infty} p_t((1 + r_t)k_t + w_t)$$

- Firm's problem:

$$\max_{k_t, l_t} p_t(f(k_t, l_t) - (r_t + \delta)k_t - w_t l_t)$$

Competitive Equilibrium

Arrow-Debreu Equilibrium

Definition

- A competitive equilibrium in this economy is a set of sequence of prices $\{p_t, r_t, w_t\}_{t=0}^{\infty}$ and quantities $\{c_t, k_{t+1}\}_{t=0}^{\infty}$ such that:

    1. Given prices, $\{c_t, k_{t+1}\}_{t=0}^{\infty}$ solve the household problem.
    2. Given prices, $\{k_t\}_{t=0}^{\infty}$ solve the firms problem.
    3. Markets clears:

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

Competitive Equilibrium

Sequential Equilibrium

- Suppose now that agents rent capital and labor to firms in return of r_t and w_t.
- Consumer problem:

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

$$\text{s.t. } c_t + k_{t+1} = w_t + (1 + r_t)k_t \ \forall \ t$$

$$\lim_{t \to \infty} \frac{k_{t+1}}{\prod_{s=1}^{t}(1 + r_s)} \geq 0$$

- As the firm's problem is static, is identical as before.

Competitive Equilibrium

Sequential Equilibrium

- A sequential market equilibrium is a sequence of prices $\{r_t, w_t\}_{t=0}^{\infty}$ and quantities $\{c_t, k_t\}_{t=0}^{\infty}$ such that:

  1. $\{c_t, k_{t+1}\}_{t=0}^{\infty}$ solve the household problem.
  2. $\{k_t\}_{t=0}^{\infty}$ solve the firms problem.
  3. Markets clear:

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

Competitive Equilibrium

Recursive Competitive Equilibrium

- Note that when we study dynamic programming approach for solving infinite horizon problems our focus was on policy functions and not on optimal sequences.
- In a recursive competitive equilibrium, the quantities and prices are defined as functions of the state.
- Hence, in a recursive competitive equilibrium both individual decisions (characterized by a value function and a decision rule) and the prices will be functions of the state.

Recursive Competitive Equilibrium

- It is not straightforward to represent the household problem in recursive form because prices are not constant.
  - They depend on the aggregate level of capital:

$$r_t = f_k(K) - \delta$$
$$w_t = f_l(K)$$

- Therefore the future continuation value will depend not only on how many assets are left for the next period but also on these prices.

- The idea is to include aggregate capital as a state variable for the household's problem.

$$V(k, K) = \max_{c,k'}\{u(c) + \beta V(k', K')\}$$
$$\text{s.t. } c + k' = w(K) + (1 + r(K))k$$
$$K' = G(K),$$

where $G(K)$ is the agent perceived law of motion of aggregate capital.

Recursive Competitive Equilibrium

Definition

- A recursive competitive equilibrium is a perceived law of motion $G(K)$, a policy function $g(k, K)$, a lifetime utility level $V(k, K)$, and a price system $r(K), w(K)$ such that

  1. $V(k, K)$ solves the household problem, and $g(k, K)$ is the associated policy function.
  2. Prices are competitively determined by firms FOCs.
  3. Consistency is satisfied:
  $$G(K) = g(K, K)$$
  4. Market clears:
  $$c + G(K) = F(K) + (1 + \delta)K$$

- The third condition states that, whenever the individual consumer is endowed with a level of capital equal to the aggregate level, his own individual behavior will exactly mimic the aggregate behavior.

Recursive Competitive Equilibrium

Algorithm

- We could use the following pseudo-code for solving for the RCE:

  1. Make a guess of $G(K)$

     1.1 Make a guess for V_0 for all values of k and K

     1.2 Apply the operator T and recover $V_1 = TV_0$ given the guess of $G(K)$

     1.3 If V_1 and V_0 are close enough, go to 2. Otherwise set $V_0 = V_1$ and back to 1.1

  2. From 1.3 recover the policy function $g(K, K)$. If $g(K, K)$ and $G(K)$ are close enough, stop. Otherwise set $G(K) = g(K, K)$ and go back to 1.1