# Value Function Iteration

Shang-Chieh Huang

November 3, 2021

# Table of Contents

# Table of Contents

# Concept of Solving Functional Equation

$$v(k) = \max_{k' \in [0,\, g(k)]} u(g(k) - k') + \beta v(k'),$$

- where $g(k) = Ak^{\alpha} + (1-\delta)k$ and $u(c) = \frac{c^{1-\theta}-1}{1-\theta}$.
- Our goal is to solve for the fixed point of contraction mapping $T: V \to V$, where $V$ is functional space.
- To solve for $v(k)$, we basically look for the fixed point of the mapping $T$.
- What is the statement means?

# Concept of Solving Functional Equation

- First, we guess an initial value function $v_0$. From the mapping $T$, we can know:

$$v_1 = Tv_0$$

$$v_2 = Tv_1$$

$$v_3 = Tv_2$$

$$\vdots$$

- Those value functions we get from mapping will converge to a fixed value function, says $v^*$:

$$v_0, v_1, v_2, \dots, v_N \to v^*$$

# Concept of Solving Functional Equation

- How can we know the mapping converge to $v^*$ or not? By checking

$$|v_1 - v_0|$$

$$|v_2 - v_1|$$

$$\vdots$$

$$|v_N - v_{N-1}| < \epsilon,$$

where $\epsilon$ is arbitrary small number.

# Concept of Solving Functional Equation

- Then we can find
  - **The value function**, $v^*(k)$. You can interpret this function as the lifetime utility that household can achieve when he/she is given $k$.
  - **The policy function**, $k' = G(k)$. Policy function is a set of rules describe what $k'$ would household choose as his/her optimal choice when he/she is given $k$.

# Table of Contents

# Setting Parameters

- Same as before

# Discretize the State and Control Variable

- Discretize the domain by construction a vector:

$$k = [0 = k_0, k_1, k_2, \dots, k_{max} = \bar{k} \equiv \left(\frac{\delta}{A}\right)^{\frac{1}{\alpha-1}}]$$

- Matlab code: k = 0 : diff : kbar
- Note that diff is 0.005 in assignment 5

# Discretize the State and Control Variable

- Now we turn back to the mapping of value function

$$v(k) = \max_{k' \in [0,\, g(k)]} u(g(k) - k') + \beta v(k')$$

- We have already constructed vector $k$.

# Discretize the State and Control Variable

- Now we turn back to the mapping of value function

$$v(k) = \max_{k' \in [0,\, g(k)]} u(g(k) - k') + \beta v(k')$$

- What is $k'$ here?
  - $k'$ has same domain as $k$
  - Thus to maximize $u(g(k) - k') + \beta v(k')$, we need to consider all possible $k'$ for each $k$ and find out the maximum.

# Construct the periodic utility function matrix

- How can we achieve it?
- First, focus on $g(k) - k'$ from $u(g(k) - k')$
- We know

$$g(k) - k' = Ak^{\alpha} + (1 - \delta)k - k'$$

# Construct the periodic utility function matrix

$$g(k) - k' = Ak^{\alpha} + (1 - \delta)k - k'$$

- We can easily get $g(k)$ from our vector $k$.
- Matlab code: gk = A*k.^alpha + (1-delta)*k

# Construct the periodic utility function matrix

$$g(k) - k' = Ak^{\alpha} + (1-\delta)k - k'$$

- Next, for each $k$, we want to consider all possible $k'$.

| | | $k'$ | | |
|---|---|---|---|---|
| | | $k_1'$ | $k_2'$ | $k_3'$ |
| | $k_1$ | $g(k_1) - k_1'$ | $g(k_1) - k_2'$ | $g(k_1) - k_3'$ |
| $k$ | $k_2$ | $g(k_2) - k_1'$ | $g(k_2) - k_2'$ | $g(k_2) - k_3'$ |
| | $k_3$ | $g(k_3) - k_1'$ | $g(k_3) - k_2'$ | $g(k_3) - k_3'$ |

- Matlab Code: mC = gk’ – k

# Construct the periodic utility function matrix

- Matlab Code: mC = gk’ – k
- Note that $(gk)' = [g(k_1) \quad g(k_2) \quad g(k_3)]'$ and $k = [k_1 \quad k_2 \quad k_3]$.
- Thus the operation: $gk' - k$ is nonsense mathematically, but Matlab will automatically turn it into

$$\begin{bmatrix} g(k_1) & g(k_1) & g(k_1) \\ g(k_2) & g(k_2) & g(k_2) \\ g(k_3) & g(k_3) & g(k_3) \end{bmatrix} - \begin{bmatrix} k_1 & k_2 & k_3 \\ k_1 & k_2 & k_3 \\ k_1 & k_2 & k_3 \end{bmatrix}$$

# Construct the periodic utility function matrix

- Note that $c = g(k) - k'$ and $c \geq 0$.
- Thus we need to replace the negative value with nan.
- Matlab Code: mC(mC < 0) = nan

# Construct the periodic utility function matrix

- Then we can calculate utility: $u(c) = \frac{c^{1-\theta}-1}{1-\theta}$.
- Matlab Code: mU = (mC.^(1-theta)-1)/(1-theta)

| | | $k'$ | | |
|---|---|---|---|---|
| | | $k'_1$ | $k'_2$ | $k'_3$ |
| | $k_1$ | $u(k_1, k'_1)$ | $u(k_1, k'_2)$ | $u(k_1, k'_3)$ |
| $k$ | $k_2$ | $u(k_2, k'_1)$ | $u(k_2, k'_2)$ | $u(k_2, k'_3)$ |
| | $k_3$ | $u(k_3, k'_1)$ | $u(k_3, k'_2)$ | $u(k_3, k'_3)$ |

# Mapping procedure $T$

- Initial guess on value function $v_0 = [0, 0, 0, \dots, 0\,]$
- Matlab code: *v0* = *zeros*(1, *length*(*k*))
- Now turn back to our mapping, we can rewrite it as

$$v_1 = max\ (mU + \beta v_0)$$

- $mU$ never change during we do the mapping, so that we can calculate it before doing “loop.”

# Mapping procedure $T$

- while loop: we can decide $\epsilon$ so that our loop would be stopped when $|v_N - v_{N-1}| < \epsilon$
- Matlab code:

```
while expression
    statements
end
```

- for loop: we can decide how many times of loop we want to do
- Matlab code: TA session on Sep. 22.

# Mapping procedure $T$

$$v_1 = max\,(mU + \beta v_0)$$

- In the loop, we want to add $v_0$ into $mU$.

| | | $k'$ | | |
|---|---|---|---|---|
| | | $k'_1$ | $k'_2$ | $k'_3$ |
| | $k_1$ | $u(k_1, k'_1) + \beta v_0(k'_1)$ | $u(k_1, k'_2) + \beta v_0(k'_2)$ | $u(k_1, k'_3) + \beta v_0(k'_3)$ |
| $k$ | $k_2$ | $u(k_2, k'_1) + \beta v_0(k'_1)$ | $u(k_2, k'_2) + \beta v_0(k'_2)$ | $u(k_2, k'_3) + \beta v_0(k'_3)$ |
| | $k_3$ | $u(k_3, k'_1) + \beta v_0(k'_1)$ | $u(k_3, k'_2) + \beta v_0(k'_2)$ | $u(k_3, k'_3) + \beta v_0(k'_3)$ |

- Convert $v_0$ into a matrix
- Matlab code: v0_m = ones(length(k),1)*v0

# Mapping procedure $T$

- Then we can have $mU + \beta v_0$
- Matlab code: wis = mU + beta*v0_m

- Next step, we want to find the maximum for each $k$.
- Matlab code: [v1,ind] = max(wis);
- v1 is a column vector recording the maximum of $Tv$ for each $k$ (row).
- ind is a column vector recording the index corresponding to the maximum value of "wis."

# Loop $T$ until value function converges

- Last step of loop, we compare the distance between $v_0$ and $v_1$.
- Matlab code: pcntol = max(abs(v1-v0))

- Stop the loop:
  - while $|v_N - v_{N-1}| < \epsilon$, or
  - for t = 100 (assignment 5)

# Value Function and Policy Function

- After we achieve convergence,
  - **Value function**: v1 in the last mapping.
  - **Policy function**: note that ind indicates for each $k$ which element in vector $k = [0, k_1, k_2, \ldots, k_{max}]$ is the optimal choice for household to choose. So we can construct policy function by choosing the element in vector $k$.
  - Matlab code: PF = k(ind)

# Simulate Time Series

- First, we combine our policy function and vector $k$ as this form:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $k_0$ | $k_1$ | $k_2$ | $k_3$ | $k_4$ | $k_5$ | $\cdots$ | $k_{max}$ |
| $k'(k_0)$ | $k'(k_1)$ | $k'(k_2)$ | $k'(k_3)$ | $k'(k_4)$ | $k'(k_5)$ | $\cdots$ | $k'(k_{max})$ |

- Suppose that we have a old steady state level of capital $k^*_{old}$, then we need to check which element in vector $k$ is most closest to $k^*_{old}$.
- Matlab Code: [min_ts, ind_ts] = min(abs(k - kss_old))
    - "ind_ts" tells us the (ind_ts)th element is most closet to $k^*_{old}$

# Simulate Time Series

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $k_0$ | $k_1$ | $k_2$ | $k_3$ | $k_4$ | $k_5$ | $\cdots$ | $k_{max}$ |
| $k'(k_0)$ | $k'(k_1)$ | $k'(k_2)$ | $k'(k_3)$ | $k'(k_4)$ | $k'(k_5)$ | $\cdots$ | $k'(k_{max})$ |

- Then initialize vectors: tsk to store the simulated time series of capital.
- Store the $k'(k_{ind\_ts})$ from policy function at the next period of the shock.
- Next, check which element in vector $k$ is most closest to $k'(k_{ind\_ts})$ …
- You can repeat this process to construct whole time series.

# Simulate Time Series

- Time series vector of consumption: $c_t = g(k_t) - k_{t+1}$
- Time series vector of output: $y_t = Ak_t^{\alpha}$
- Time series vector of investment: $x_t = y_t - c_t$