Value Function Iteration & Stochastic Growth Model

Emile Alexandre Marin

emarin@ucdavis.edu

October 17, 2022

Dynamic Models: features and solution approaches

- Value function iteration: deterministic dynamic programming;
  - Discretization;
  - Finite element method;
  - Chebyshev polynomials (time permitting);
- Value function iteration: how to deal with uncertainty.

Deterministic Growth Model

The model's characteristics:

1. time is discrete and starts from 0
2. infinite horizon
3. no uncertainty
4. no productivity growth
5. only 1 consumer

The consumer maximizes utility to choose consumption and investment.

Deterministic Growth Model

The model's characteristics:

1. time is discrete and starts from 0
2. infinite horizon
3. no uncertainty
4. no productivity growth
5. only 1 consumer

The consumer maximizes utility to choose consumption and investment.

Two methods to solve the problem:

Deterministic Growth Model

The model's characteristics:

1. time is discrete and starts from 0
2. infinite horizon
3. no uncertainty
4. no productivity growth
5. only 1 consumer

The consumer maximizes utility to choose consumption and investment.

Two methods to solve the problem:

- Optimal control
- Dynamic programming

Optimal control

You need the sequential formulation of the problem:

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

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

$$k_{t+1}, c_t > 0 \text{ and } k_0 \text{ given}$$

$$\beta \in (0,1)$$

Optimal control

You need the sequential formulation of the problem:

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

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

$$k_{t+1}, c_t > 0 \text{ and } k_0 \text{ given}$$

$$\beta \in (0,1)$$

1. Write the Lagrangian of the problem.
2. Take the first order conditions (FOCs) and derive the Euler equation
3. Collect the FOCs and other conditions that must hold in equilibrium (e.g. from imposing market equilibrium)

Optimal control

You need the sequential formulation of the problem:

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

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

$$k_{t+1}, c_t > 0 \text{ and } k_0 \text{ given}$$

$$\beta \in (0,1)$$

1. Write the Lagrangian of the problem.
2. Take the first order conditions (FOCs) and derive the Euler equation
3. Collect the FOCs and other conditions that must hold in equilibrium (e.g. from imposing market equilibrium)

The solution is an infinite sequence $\{c_t, k_{t+1}\}_{t=0}^{\infty}$ *which solves the system of equations.*

Dynamic programming

You need the recursive representation of the problem

$$V(k) = \max_{c,k'} \left\{ u(c) + \beta V\left(k'\right) \right\}$$
$$c + k' = F(k) + (1-\delta)k$$
$$c \geq 0$$
$$k' \geq 0$$

Dynamic programming

You need the recursive representation of the problem

$$
\begin{aligned}
V(k) &= \max_{c,k'} \left\{ u(c) + \beta V\left(k'\right) \right\} \\
c + k' &= F(k) + (1-\delta)k \\
c &\geq 0 \\
k' &\geq 0
\end{aligned}
$$

but lets take a step back first!

From Optimal Control to Dynamic Programming

We can rewrite the sequential problem as:

$$V(k_0) = \sum_{t=0}^{\infty} \beta^t u(\hat{c}_t)$$

$$\hat{c}_t + \hat{k}_{t+1} = F\left(\hat{k}_t\right) + (1-\delta)\hat{k}_t$$

From Optimal Control to Dynamic Programming

We can rewrite the sequential problem as:

$$V(k_0) = \sum_{t=0}^{\infty} \beta^t u(\hat{c}_t)$$
$$\hat{c}_t + \hat{k}_{t+1} = F\left(\hat{k}_t\right) + (1-\delta)\hat{k}_t$$

where $V(k_0)$ is the value **function**, i.e. if we vary k_0..., V will change.

From Optimal Control to Dynamic Programming

We can rewrite the sequential problem as:

$$V(k_0) = \sum_{t=0}^{\infty} \beta^t u(\hat{c}_t)$$

$$\hat{c}_t + \hat{k}_{t+1} = F\left(\hat{k}_t\right) + (1-\delta)\hat{k}_t$$

where $V(k_0)$ is the value **function**, i.e. if we vary k_0..., V will change.

Write out : $V(k_0) = u(\hat{c}_0) + \underbrace{\beta u(\hat{c}_1) + \beta^2 u(\hat{c}_2) + \ldots}_{\overrightarrow{V}_1}$

From Optimal Control to Dynamic Programming

We can rewrite the sequential problem as:

$$V(k_0) = \sum_{t=0}^{\infty} \beta^t u(\hat{c}_t)$$
$$\hat{c}_t + \hat{k}_{t+1} = F\left(\hat{k}_t\right) + (1-\delta)\hat{k}_t$$

where $V(k_0)$ is the value **function**, i.e. if we vary k_0..., V will change.

Write out : $V(k_0) = u(\hat{c}_0) + \underbrace{\beta u(\hat{c}_1) + \beta^2 u(\hat{c}_2) + ...}_{\overrightarrow{V}_1}$

* $\overrightarrow{V}_1$ is a continuation value

Bellman's Principle: the jist

* $\{c_t, k_{t+1}\}_{t=1}^{\infty}$ is the continuation plan

Bellman's Principle: the jist

* $\{c_t, k_{t+1}\}_{t=1}^{\infty}$ is the continuation plan
* if we knew $\overrightarrow{V}_1$, easy problem to find $\hat{c}_0, \hat{k}_1$

Bellman's Principle: the jist

* $\{c_t, k_{t+1}\}_{t=1}^{\infty}$ is the continuation plan
* if we knew $\overrightarrow{V}_1$, easy problem to find $\hat{c}_0, \hat{k}_1$

$\Longrightarrow$ we would know $V(k_0)$

Bellman's Principle: the jist

* $\{c_t, k_{t+1}\}_{t=1}^{\infty}$ is the continuation plan
* if we knew $\overrightarrow{V}_1$, easy problem to find $\hat{c}_0, \hat{k}_1$

$\Longrightarrow$ we would know $V(k_0)$

"An optimal policy [plan] has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision".

Bellman's Principle: the jist

* $\{c_t, k_{t+1}\}_{t=1}^{\infty}$ is the continuation plan
* if we knew $\overrightarrow{V}_1$, easy problem to find $\hat{c}_0, \hat{k}_1$

$\Longrightarrow$ we would know $V(k_0)$

"An optimal policy [plan] has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision".

* $\overrightarrow{V}_1 = V(\hat{k}_{t+1})$ (prove, go through the cases)

From optimal control to recursive

So we can go from:

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

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

$$k_{t+1}, c_t > 0 \text{ and } k_0 \text{ given}$$

$$\beta \in (0,1)$$

From optimal control to recursive

So we can go from:

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

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

$$k_{t+1}, c_t > 0 \text{ and } k_0 \text{ given}$$

$$\beta \in (0,1)$$

to

$$V(k_0) = \max_{c_0, k_1} \{u(c_0) + \beta V(k_1)\}$$

$$s.t. \quad c_0 + k_1 = F(k_0) + (1-\delta)k_0$$

From optimal control to recursive

So we can go from:

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

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

$$k_{t+1}, c_t > 0 \text{ and } k_0 \text{ given}$$

$$\beta \in (0,1)$$

to

$$V(k_0) = \max_{c_0,k_1}\{u(c_0) + \beta V(k_1)\}$$

$$s.t. \quad c_0 + k_1 = F(k_0) + (1-\delta)k_0$$

- and time subscripts redundant, only state k matters

Bellman Operator and Policy Functions

We are interested in $c = h(k)$ and $k' = g(k)$:

$$\begin{aligned} V(k) &= u(h(k) + \beta V(g(k))\} \\ s.t. \quad h(k) + g(k) &= F(k) + (1-\delta)k \end{aligned}$$

- To find policy functions, need $V(k)$
  - functional equation in one unknown function)
- Since everything depends on k, make grid and guess and verify
- Once we have h and g we can simulate model

VFI: Discretization 1/2

1. Set:
   - n_k (number of grid points);
   - $[k_{\min}, k_{\max}]$ (support of k);
   - ε (the tolerance error);
2. Define the grid $\{k_1, k_2, \dots, k_{n_k}\}$
3. Choose the initial guess of the value function: $V^0 = \{V_i^0\}_{i=1}^{n_k}$
4. Find $V^1 = \{V_i^1\}_{i=1}^{n_k}$ following the next steps:

a) For every $j = 1, \dots, n_k$ compute:

$$V_{i,j}^1 = u\left(F\left(k_i\right) + (1-\delta)k_i - k_j\right) + \beta V_j^0$$

b) Choose j which gives the highest $V_{i,j}^1$ and set $V_i^1 = V_{i,j}^1$. Store the optimal decision rule as $j = g_i \in \{1, 2, \dots, n_k\}$.

VFI: Discretization 2/2

5. Do a) and b) for every $i = 1, \dots, n_k$ until you get the entire vector $V^1 = \{V_i^1\}_{i=1}^{n_k}$
6. Compare V^0 and V^1. Compute:

$$d = \max_{i=1,2,\dots,n_k} \left| V_i^0 - V_i^1 \right|$$

   a) if $d > \varepsilon$ go to step 4 and set $V^0 = V^1$;
   b) if $d \leq \varepsilon$ you are done: $V^0 = V^*$ and the optimal decision rule is $g^* = \{g_i\}_{i=1}^{n_k}$

VFI: Discretization 2/2

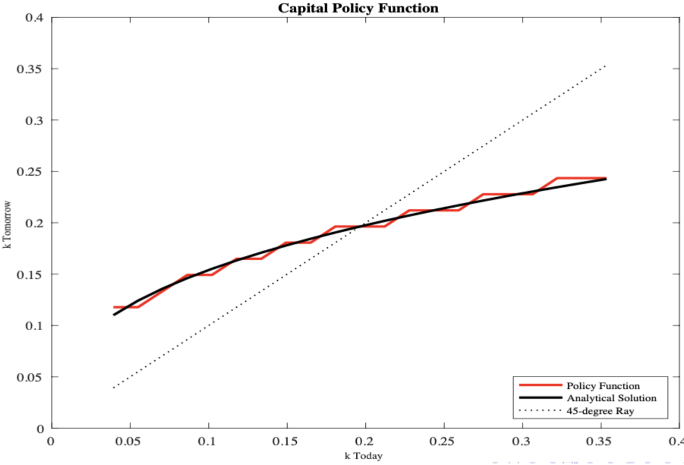
5. Do a) and b) for every $i = 1, \dots, n_k$ until you get the entire vector $V^1 = \{V_i^1\}_{i=1}^{n_k}$
6. Compare V^0 and V^1. Compute:

$$d = \max_{i=1,2,\dots,n_k} \left| V_i^0 - V_i^1 \right|$$

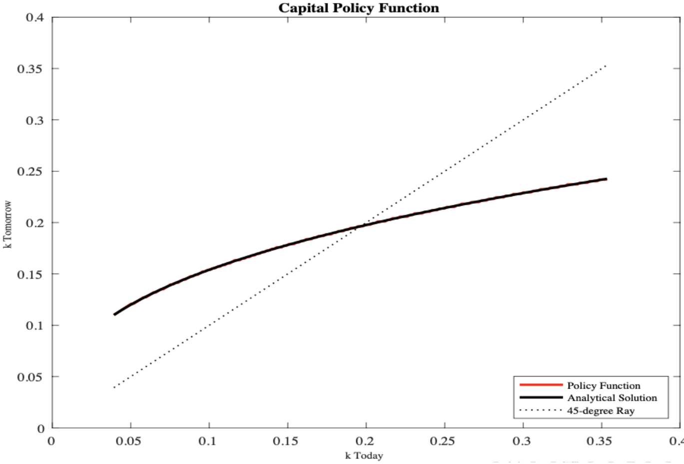
   a) if $d > \varepsilon$ go to step 4 and set $V^0 = V^1$;
   b) if $d \leq \varepsilon$ you are done: $V^0 = V^*$ and the optimal decision rule is $g^* = \{g_i\}_{i=1}^{n_k}$

CAVEATS:
a) Check the bounds of $\{k_1, \dots, k_{n_k}\}$;
b) Redo 1)- 6) with a smaller ε.
c) Increase n_k.

$\varepsilon = 0.0001$
$\mathbf{n}_k = \mathbf{21}$
$k_{\min} = 0.1k_{ss}$ and $k_{\min} = 1.9k_{ss}$



$\varepsilon = 0.0001$
$\mathbf{n}_k = \mathbf{201}$
$k_{\min} = 0.1k_{ss}$ and $k_{\min} = 1.9k_{ss}$



Improvement algorithms (1/3)

Step 4 (maximization) is the computationally expensive part.

Improvement algorithms (1/3)

Step 4 (maximization) is the computationally expensive part.

When problem convave $\implies$ values of objective function decline after optimal!

Improvement algorithms (1/3)

Step 4 (maximization) is the computationally expensive part.

When problem convave $\implies$ values of objective function decline after optimal!

4. Find $V^1 = \{V_i^1\}_{i=1}^{n_k}$ following the next steps. For every $i = 1, \ldots, n_k$:

   a) Set $j = 1$;

   b) Compute $V_{i,j}^1$:

$$V_{i,j}^1 = u\left(F\left(k_i\right) + (1-\delta)k_i - k_j\right) + \beta V_j^0$$

c) Compute $V_{i,j+1}^1$:

$$V_{i,j+1}^1 = u\left(F\left(k_i\right) + (1-\delta)k_i - k_{j+1}\right) + \beta V_{j+1}^0$$

d) Compare $V_{i,j}^1$ and $V_{i,j+1}^1$

   i. If $V_{i,j}^1, < V_{i,j+1}^1$ do $j = j + 1$ and go to step b)
   ii. If $V_{i,j}^1 \geq V_{i,j+1}^1$you are done and $V_i^1 = V_{i,j}^1,\ and g_i = j$.

Improvement algorithms (2/3)

If the optimal control is also monotone:

Improvement algorithms (2/3)

If the optimal control is also monotone:

4. Find $V^1 = \{V_i^1\}_{i=1}^{n_k}$ following the next steps. For every $i = 1, \ldots, n_k$:
   a) Find the lower bound of the optimal choice $j_{\min}$. For $i = 1$, $j_{\min} = 1$. For $i > 1, j_{\min} = g_{i-1}$
   b) For every $j = j_{\min}, j_{\min} + 1, \ldots, n_k$ compute:

$$V_{i,j}^1 = u\left(F\left(k_i\right) + (1-\delta)k_i - k_j\right) + \beta V_j^0$$

c) Choose j which gives the highest $V_{i,j}^1$ and set $V_i^1 = V_{i,j}^1$. Store the optimal decision rule as $j = g_i \in \{1, 2, \ldots, n_k\}$

Improvement algorithms (3/3)

Using information on the policy function (PF), we can improve speed of convergence in V (Howard's improvmt algorithm)

Improvement algorithms (3/3)

Using information on the policy function (PF), we can improve speed of convergence in V (Howard's improvmt algorithm)

Key: Policy function often converges faster than VF

Improvement algorithms (3/3)

Using information on the policy function (PF), we can improve speed of convergence in V (Howard's improvmt algorithm)

Key: Policy function often converges faster than VF

Choose n_h : # times to update the VF using current PF

4. Set $V^{1,1} = V^1$. Then update the value function $V^{1,t}$ by applying the following steps n_h times and obtain V^{1,n_k}. Replace V^1 by V^{1,n_k} and go to step 5

a) For each $i = 1, \ldots, n_h$ we have the optimal decision $g_i \in \{1, 2, \ldots, n_k\}$ associated for each i.

b) Update the value function from $V^{1,t}$ to $V^{1,t+1}$ using the following modified Bellman equation for each $i = 1, \ldots, n_k$:

$$V_i^{1,t+1} = u\left(F\left(k_i\right) + (1-\delta)k_i - k_{g_i}\right) + \beta V_{g_i}^{1,t}$$

Improvement algorithms (3/3)

Using information on the policy function (PF), we can improve speed of convergence in V (Howard's improvmt algorithm)

Key: Policy function often converges faster than VF

Choose n_h : # times to update the VF using current PF

4. Set $V^{1,1} = V^1$. Then update the value function $V^{1,t}$ by applying the following steps n_h times and obtain V^{1,n_k}. Replace V^1 by V^{1,n_k} and go to step 5

a) For each $i = 1, \ldots, n_h$ we have the optimal decision $g_i \in \{1, 2, \ldots, n_k\}$ associated for each i.

b) Update the value function from $V^{1,t}$ to $V^{1,t+1}$ using the following modified Bellman equation for each $i = 1, \ldots, n_k$:

$$V_i^{1,t+1} = u\left(F\left(k_i\right) + (1-\delta)k_i - k_{g_i}\right) + \beta V_{g_i}^{1,t}$$

What if $n_h \to \infty$?

Intuition to speed up VFI

+ VFI is a stable and reliable convergence algorithm
+ Can easily handle occasionally binding constraints (global)
− Slow and suffers from curse of dimensionality

Do not do any operations that are not necessary!

e.g. : Calculating $F(k_i) + (1-\delta)k_i - k_j$ does not depend on V or PF ! Calculate once at the beginning and retrieve values as needed!

VFI: Finite element method

- Likely, your V^{i+1} will not lie on the grid $\implies$ error

SO approximate V with a continuous function

VFI: Finite element method

- Likely, your V^{i+1} will not lie on the grid $\implies$ error

SO approximate V with a continuous function

We split the domain of x into different regions and use different polynomial for each region.

* Piece-wise linear for $x \in [x_i, x_{i+1}]$

$$f(x) \approx f_i + \left(\frac{f_{i+1} - f_i}{x_{i+1} - x_i}\right)(x - x_i)$$

CAVEAT: it is not differentiable in the nodes.

* n^{th}-order spline for $x \in [x_{i-1}, x_i]$

$$f(x) \approx a_i + b_i x + c_i x^2 + d_i x^3 + .. + z_i x^n$$

This is differentiable & shapre preserving

VFI: Finite element method (piece-wise linear)

1. Set:
   - n_k (number of grid points);
   - $[k_{\min}, k_{\max}]$ (support of k);
   - ε (the tolerance error);
2. Define the grid $\{k_1, k_2, \ldots, k_{n_k}\}$
3. Choose the initial guess of the value function at the nodes: $V^0 = \{V_i^0\}_{i=1}^{n_k}$
4. Call $\widetilde{V}(k)$ the approximation of the value function implied by the piece-wise linear interpolation with $\{V_i^0\}_{i=1}^{n_k}$ (in MATLAB interp1 (., ., ., 'linear')
5. $\forall i = 1, 2, \ldots, n_k$ solve the following problem

$$\max_{k' \in [k_{\min}, k_{\max}]} \{u(F(k_i) + (1-\delta)k_i - k')\} + \beta \widetilde{V}^0(k')$$

(CAVEAT: one dimension optimization algorithm needed)

6. Set g_i the argmax of the previous problem.
7. Update the value function, such as:

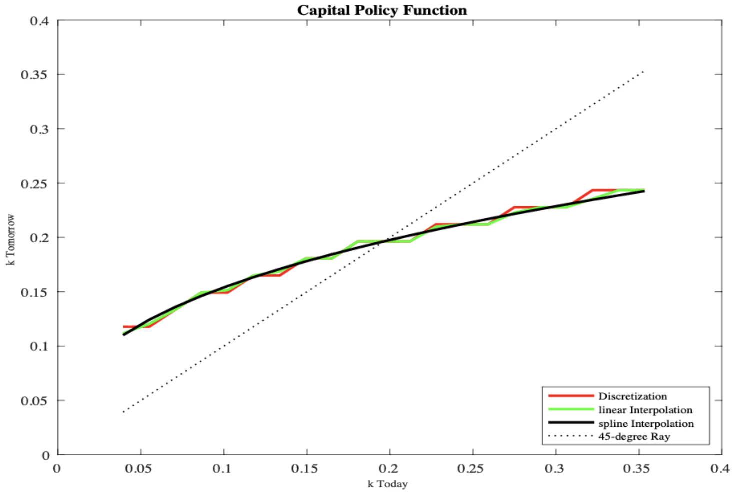
$$V_i^1 = \{u\left(F\left(k_i\right) + (1-\delta)k_i - g_i\right)\} + \beta\widetilde{V}^0\left(g_i\right)$$

9. Compare V^0 and V^1. Compute:

$$d = \max_{i=1,2,\ldots,n_k} \left| V_i^0 - V_i^1 \right|$$

   a) if $d > \varepsilon$ go to step 4 and set $V^0 = V^1$
   b) if $d \leq \varepsilon$ you are done: $V^0 = V^*$ and the optimal decision rule is $g^* = \{g_i\}_{i=1}^{n_k}$

CAVEAT: This approximation is not differentiable in the nodes. A spline is more advisable, but has more conditions to satisfy.


Capital Policy Function
k Tomorrow
k Today
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
Discretization
linear Interpolation
spline Interpolation
45-degree Ray

VFI: Chebyshev

This time we approximate the value function using Chebyshev polynomials of order n.

Chebyshev polynomial of order j defined in $[-1, 1]$:

$$T_j(x) = \cos\left(j \cos^{-1} x\right)$$

The polynomials are:

$$\begin{aligned}
T_0(x) &= 1 \\
T_1(x) &= x \\
T_j(x) &= 2xT_{j-1}(x) - T_{j-2}(x) \quad \text{ for } j > 1
\end{aligned}$$

The approximation of a function $f(x)$ defined in $[-1, 1]$ with Chebyshev polynomials of order n is

$$\widetilde{f}(x) = \sum_{j=0}^{n} \theta_j T_j(x)$$

The problem is to choose $n + 1$ coefficients $\theta = \{\theta_j\}_{j=0}^{n}$ such as the error between $f(x)$ and $\widetilde{f}(x)$ is minimized.

VFI: Chebyshev

1. Choose the number of nodes m :

$$\text{(a) } m = n + 1 \Longrightarrow \text{ Chebyshev collocation}$$
$$\text{(b) } m > n + 1 \Longrightarrow \text{ Chebyshev regression}$$

3. Find the nodes $\{z_i\}_{i=1}^{m}$

$$z_i = -\cos\left(\frac{(2i-1)\pi}{2m}\right)$$

4. Find the coefficients minimizing the sum of the squared errors between $f(x)$ and $\widetilde{f}(x)$ evaluated in $\{z_i\}_{i=1}^{m}$

$$\theta_0 = \frac{1}{m}\sum_{i=1}^{m} f(z_i)$$
$$\theta_j = \frac{2}{m}\sum_{i=1}^{m} f(z_i)\, T_j(z_i)$$

VFI: Chebyshev Cookbook

1. Set:

   - order of the polynomial n; number of collocation points m such that $m \geq n+1$;
   - ε (the tolerance error);
   - $[k_{\min}, k_{\max}]$ (support of k)

3. Compute the Chebyshev roots $\{z_i\}_{i=1}^{m}$

$$z_i = -\cos\left(\frac{(2i-1)\pi}{2m}\right)$$

4. Compute the collocation points implied by $\{z_i\}_{i=1}^{m}$

$$k_i = (z_i+1)\left(\frac{k_{\max}-k_{\min}}{2}\right) + k_{\min}$$

5. We want to approximate the value function $V(k)$ by

$$\widetilde{V}(k) = \sum_{j=0}^{n} \theta_j T_j t(k)$$

VFI: Chebyshev

6. Implicitly guess $\left\{\theta_j^0\right\}_{j=0}^n$:

   (a) Choose a guess of the value function at the collocation points $\{k_i\}_{i=1}^m$ and call them $\{y_i^0\}_{i=1}^m$;

   (b) back up the values of $\left\{\theta_j^0\right\}_{j=0}^n$ using:

$$\theta_0^0 = \frac{1}{m}\sum_{i=1}^m y_i^0$$

$$\theta_j^0 = \frac{2}{m}\sum_{i=1}^m y_i^0 T_j\left(z_i\right)$$

7. Denote the guessed value function as $\widetilde{V}^0(k)$;

8. For each $i = 1, 2, \ldots, m$ solve the following problem

$$\max_{k' \in [k_{\min}, k_{\max}]} \left\{u\left(F\left(k_i\right) + (1-\delta)k_i - k'\right)\right\} + \beta \widetilde{V}^0\left(k'\right)$$

using a one dimension optimization algorithm.

9. Set g_i the argmax of the previous problem
10. Use g_i to update the value function:

$$y_i^1 = \{u\left(F\left(k_i\right) + (1-\delta)k_i - g_i\right)\} + \beta \widetilde{V}^0\left(g_i\right)$$

11. Obtain an updated guess of the coefficients:

$$\theta_0^1 = \frac{1}{m}\sum_{i=1}^{m} y_i^1$$

$$\theta_j^1 = \frac{2}{m}\sum_{i=1}^{m} y_i^1 T_j\left(z_i\right)$$

12. Compare $\left\{\theta_j^0\right\}_{j=0}^{n}$ and $\left\{\theta_j^1\right\}_{j=0}^{n}$:

$$d = \max_{j=1,\ldots,n_k} \left|\theta_j^0 - \theta_j^1\right|$$

(a) if $d > \varepsilon$ set $\theta^0 = \theta^1$ and go to step 6
(b) if $d \leq \varepsilon$ you are done: $V^0 = V^*$ and the optimal decision rule is $g^* = \{g_i\}_{i=1}^{m}$

Moving to Stochastics

Suppose now that the model is the following:

$$
\begin{aligned}
V(k,z) &= \max_{c,k'} \mathbb{E}\left\{u(c) + \beta V\left(k', z'\right)\right\} \\
c + k' &= zF(k) + (1-\delta)k \\
c &\geq 0 \\
k' &\geq 0
\end{aligned}
$$

where z is a stochastic shock.

- discrete valued iid shock (very easy)
- a (discrete valued) Markov chain (easy)
- a (continuous valued) autoregressive process (not so easy - need to discretise/approximate)

Moving to Stochastics

Suppose now that the model is the following:

$$V(k,z) = \max_{c,k'} \mathbb{E}\left\{u(c) + \beta V\left(k', z'\right)\right\}$$
$$c + k' = zF(k) + (1-\delta)k$$
$$c \geq 0$$
$$k' \geq 0$$

where z is a stochastic shock.

- discrete valued iid shock (very easy)
- a (discrete valued) Markov chain (easy)
- a (continuous valued) autoregressive process (not so easy - need to discretise/approximate)

The model now has two state variables k and z:

- The social planner now chooses policy contingent on the realization of the shock

Moving to Stochastics

Suppose now that the model is the following:

$$
\begin{aligned}
V(k,z) &= \max_{c,k'} \mathbb{E}\left\{u(c) + \beta V\left(k', z'\right)\right\} \\
c + k' &= zF(k) + (1-\delta)k \\
c &\geq 0 \\
k' &\geq 0
\end{aligned}
$$

where z is a stochastic shock.

- discrete valued iid shock (very easy)
- a (discrete valued) Markov chain (easy)
- a (continuous valued) autoregressive process (not so easy - need to discretise/approximate)

The model now has two state variables k and z:

- The social planner now chooses policy contingent on the realization of the shock

Unconditional distribution (Transition Matrices)

Consider a state vector $v = [01]'$ and transition matrix
$T = \begin{bmatrix} \pi_{11} & \pi_{1,2} \\ \pi_{21} & \pi_{22} \end{bmatrix}$

Unconditional distribution (Transition Matrices)

Consider a state vector $v = [01]'$ and transition matrix
$T = \begin{bmatrix} \pi_{11} & \pi_{1,2} \\ \pi_{21} & \pi_{22} \end{bmatrix}$

Probability of the state in t years: $v_t = (T')^t v_0 = T' v_{t-1}$

Unconditional distribution (Transition Matrices)

Consider a state vector $v = [01]'$ and transition matrix
$T = \begin{bmatrix} \pi_{11} & \pi_{1,2} \\ \pi_{21} & \pi_{22} \end{bmatrix}$

Probability of the state in t years: $v_t = (T')^t v_0 = T' v_{t-1}$

The unconditional distribution is given by $v = T' v$

Unconditional distribution (Transition Matrices)

Consider a state vector $v = [01]'$ and transition matrix
$T = \begin{bmatrix} \pi_{11} & \pi_{1,2} \\ \pi_{21} & \pi_{22} \end{bmatrix}$

Probability of the state in t years: $v_t = (T')^t v_0 = T' v_{t-1}$
The unconditional distribution is given by $v = T'v$

$$\implies (I - T')v = 0$$

v is the eigenvector associated with a unit eigenvalue of matrix T' and normalised to sum to one

Stochastic Growth Model

With a discrete iid shock:

- Assume

$$z_t = \begin{cases} z_1 & \text{w.p. } \pi_1 \\ z_2 & \text{w.p. } \pi_2 \end{cases}, \text{ for all } t$$

- The Bellman equation is then

$$\begin{aligned} V(k,z) &= \max\left\{U + \beta EV\left(k', z'\right)\right\} \\ &= \max\left\{U + \beta \sum_{i=1}^{2} \pi_i V_i\left(k', z_i'\right)\right\} \end{aligned}$$

- Steps are like for the deterministic case but repeat twice, one for each state z

Stochastic VFI algorithm 1/2

1. Set:
   - n_k (number of grid points);
   - $[k_{\min}, k_{\max}]$ (support of k);
   - ε (the tolerance error);
2. Define the grid $\{k_1, k_2, \dots, k_{n_k}\}$
3. Choose the initial guess of the value functions: $V_1^0 = \left\{V_{1,i}^0\right\}_{i=1}^{n_k}$ and $V_2^0 = \left\{V_{2,i}^0\right\}_{i=1}^{n_k}$
4. Find $V_1^1 = \left\{V_{1,i}^1\right\}_{i=1}^{n_k}$ and $V_2^1 = \left\{V_{2,i}^1\right\}_{i=1}^{n_k}$:
   a. For every $j = 1, \dots, n_k$ compute:

$$
\begin{aligned}
V_{1,i,j_1}^1 =& u\left(z_1 F\left(k_i\right) + (1-\delta)k_i - k_{j_1}\right) \\
&+ \beta\left(\pi_1 V_{1,j_1}^0 + \pi_2 V_{2,j_1}^0\right) \\
V_{2,i,j_2}^1 =& u\left(z_2 F\left(k_i\right) + (1-\delta)k_i - k_{j_2}\right) \\
&+ \beta\left(\pi_1 V_{1,j_2}^0 + \pi_2 V_{2,j_2}^0\right)
\end{aligned}
$$

Stochastic VFI algorithm 1/2

b. Choose j_1 and j_2 which gives the highest V^1_{1,i,j_1} and V^1_{2,i,j_2} and set $V^1_{1,i} = V^1_{1,i,j}$ and $V^1_{2,i} = V^1_{2,i,j_2}$. Store the optimal decision rule as $j_1 = g_{1,i} \in \{1, 2, \ldots, n_k\}$ and $j_2 = g_{2,i} \in \{1, 2, \ldots, n_k\}$

5. Do a) and b) for every $i = 1, \ldots, n_k$ until you get the entire vector

$$V^1_1 = \left\{ V^1_{1,i} \right\}_{i=1}^{n_k} \text{ and } V^1_2 = \left\{ V^1_{2,i} \right\}_{i=1}^{n_k}$$

7. Compare V^0 and V^1. Compute:

$$d_1 = \max_{i=1,2,\ldots,n_k} \left| V^0_{1,i} - V^1_{1,i} \right|$$

$$d_2 = \max_{i=1,2,\ldots,n_k} \left| V^0_{2,i} - V^1_{2,i} \right|$$

a) if $d_1 > \varepsilon$ or $d_2 > \varepsilon$ go to step 4 and set $V^0_1 = V^1_1$ and $V^0_2 = V^1_2$;

b) if $d_1 < \varepsilon$ and $d_2 < \varepsilon$ you are done: $V^0_1 = V^*_1$ and $V^0_2 = V^*_2$ and the optimal decision rules is $g^*_1 = \{g_{1,i}\}_{i=1}^{n_k}$ and

Stochastic VFI (Markov Chain)

- Assume

$$z_t = \begin{cases} z_1 \\ z_2 \end{cases}$$

with transition matrix

$$\Pi = \begin{bmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{bmatrix}$$

where $\pi_{i1} + \pi_{i2} = 1$ (rows sum up to one)

- The steps are the same as with iid shocks - step 4 is now

$$V^1_{1,i,j_1} = u\left(z_1 F\left(k_i\right) + (1-\delta)k_i - k_{j_1}\right) + \beta\left(\pi_{1,1} V^0_{1,j_1} + \pi_{1,2} V^0_{2,j_1}\right)$$
$$V^1_{2,i,j_2} = u\left(z_2 F\left(k_i\right) + (1-\delta)k_i - k_{j_2}\right) + \beta\left(\pi_{2,1} V^0_{1,j_2} + \pi_{2,2} V^0_{2,j_2}\right)$$

Stochastic VFI (Stochastic Process)

Assume now that

$$\log z_t = \rho \log z_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim \text{iid}\left(0, \sigma^2\right)$$

- Problem: The (stochastic) state variable z_t is continuous. Like for capital, we need to discretise the space that z_t takes values from
- Solution: we approximate the Markov process with a Markov chain
- Method: Tauchen (see Heer and Maussner pp 497-499 for the algorithm)
- Then use the approximate Markov chain as above

Readings

- Heer and Maussner,2008, "Value Function Itereration as a Solution Method for the Ramsey Model", CESifo Wp n.2278
- Heer and Maussner, chapters 8-9, relevant sections
- Heer and Maussner, 1.2 and 1.3
- Ljunqvist and Sargent, p. 29 - 36 and p. 39 – 41
- Tauchen, 1986, Finite state markov-chain approximations to univariate and vector autoregressions, Economics Letters
- Wouter den Haan Notes *http://www.wouterdenhaan.com/notes*