

# Advanced Macroeconomics II

## Lecture 4

### Numerical Dynamic Programming

**Isaac Baley**

UPF & Barcelona GSE

January 15, 2016

# Roadmap

---

- ① **Example: Savings problem**
- ② Value function iteration.
- ③ Policy function iteration.
- ④ Example in Matlab: Stochastic Discrete Cake-Eating

# Savings problem (1)

---

- Consider a savings problem.
- Given assets  $a$  and income  $y$ , the agent maximizes her utility by choosing consumption  $c$  and assets next period  $a'$ .

$$V(a, y) = \max_{c, a'} u(c) + \beta \mathbb{E}[V(a', y')]$$

$$a' = g(a, y, c)$$

$y$  is an exogenous stochastic process

- ▶  $u(c)$  and  $g(a, y, c)$  are the functions you know.
- ▶  $V(a, y)$  and  $c(a, y)$  are the function you need to find

## Savings problem (2)

---

- A specific case:

$$V(a, y) = \max_{c, a'} u(c) + \beta \mathbb{E} \{V(a', y')\}$$

- $g = (a, y, c)$  is the budget constraint:

$$a' = R(a + y - c)$$

- Assume a borrowing constraint (cannot eat more than what you have):

$$c \leq a + y$$

- Define  $w \equiv a + y$  and rewrite the problem as:

$$V(w) = \max_{0 \leq c \leq w} u(c) + \beta \mathbb{E}_t \{V(w')\}$$

$$w' = R(w - c) + y'$$

## Before going to computer...

- Choose  $u(c)$ . In macro most widely used is CES:

$$u(c) = \frac{c^{1-\gamma}}{1-\gamma}$$

- Choose stochastic process for  $y$ . We assume  $y$  is *iid* and consider two values:

$$y_t \in \{y^H, y^L\}$$

where the matrix of transition probabilities

	$y_{t+1} = y^L$	$y_{t+1} = y^H$
$y_t = y^L$	$\pi_L$	$\pi_H$
$y_t = y^H$	$\pi_L$	$\pi_H$

- With all the elements together, the Bellman Equation is

$$V(w) = \max_{0 \leq c \leq w} \frac{c^{1-\gamma}}{1-\gamma} + \beta \sum_{i=L,H} \pi_i V[R(w-c) + y^i] \quad (1)$$

# Roadmap

---

- ① Example: Savings problem
- ② **Value function iteration.**
- ③ Policy function iteration.
- ④ Example in Matlab: Stochastic Discrete Cake-Eating

# Value Function Iteration (1)

---

- Steps:

- ① Discretize the state space for  $w$  and for choices  $c$ .
- ② Guess an initial value function  $V^1(w)$
- ③ Find optimal  $c^1(w)$ .
- ④ Derive a new guess of the value function as

$$V^2(w) = u(c^1(w)) + \beta \mathbb{E} \{ V^1(w') \}$$

- ⑤ Define a metric for convergence and repeat points 3 and 4 until converge:  $V^1(\cdot) = V^2(\cdot)$
- Key for method: **Contraction mapping theorem (CMT)**
    - ▶ Guarantees convergence of iterations
    - ▶ Powerful numerical method for finding the solution.

## Step 1: Discretization

---

- Define  $[\bar{w}_L, \bar{w}_H]$  and choose the number of points  $n$ . Therefore:

$$\begin{bmatrix} w^1 = \bar{w}_L \\ w^2 \\ \vdots \\ w^{n-1} \\ w^n = \bar{w}_H \end{bmatrix}$$

- Ideally you want more points where it is needed, i.e. the value function is less linear.
- Analogously, discretise  $c$  in  $M$  points:  $c_i \in \{c_1, \dots, c_M\}$

## Step 2: Initial Guess

---

- A guess of  $V^1$  is a vector of  $n$  values, one for each state.
- In theory any initial guess works:

$$V^1 = \begin{bmatrix} V^1(w^1) \\ V^1(w^2) \\ \dots \\ V^1(w^n) \end{bmatrix}$$

- In practice better to find a guess as close as possible to the final result.
- Educated guess  $\Rightarrow$  Faster convergence.

## Step 3: Find optimal policy

---

- For each state  $w$ , choose  $c_i \in \{c_1, \dots, c_M\}$  that maximises the value function.
- Start with point  $w^1$  :

$$\begin{aligned}V(w^1 | c_1) &= u(c_1) + \beta \sum_{i=L,H} \pi_i V^1 [R(w^1 - c_1) + y^i] \\ &\dots \\ V(w^1 | c_M) &= u(c_M) + \beta \sum_{i=L,H} \pi_i V^1 [R(w^1 - c_M) + y^i]\end{aligned}$$

- Optimal policy is  $c^1$  given by

$$c^1 = \arg \max u(c^1) + \beta \sum_{i=L,H} \pi_i V^1 [R(w^1 - c^1) + y^i]$$

- Note:  $R(w^1 - c) + y^i$  is usually not on the  $\{w^1, \dots, w^n\}$  grid. Need to interpolate!

## Step 4: Update guess

---

- Must repeat this process for each of the  $n$  states  $w^1 \dots w^n$ .
- After  $n * M$  evaluations you get the policy function  $c(w)$ , which is a collection of  $n$  values for  $c$ , one for each state.

$$c^1 = \arg \max u(c^1) + \beta \sum_{i=L,H} \pi_i V^1 [R(w^1 - c^1) + y^i]$$

...

$$c^n = \arg \max u(c^n) + \beta \sum_{i=L,H} \pi_i V^1 [R(w^n - c^n) + y^i]$$

- Then compute the new guess:

$$V^2(w^1) = u(c^1) + \beta \sum_{i=L,H} \pi_i V^1 [R(w^1 - c^1) + y^i]$$

...

$$V^2(w^n) = u(c^n) + \beta \sum_{i=L,H} \pi_i V^1 [R(w^n - c^n) + y^i]$$

## Step 5: Check convergence

---

- Define a metric  $d(V^1, V^2)$  to evaluate convergence.
  - ▶ Sum of square deviations  $< L$
  - ▶ Sum of absolute deviations  $< L$
  - ▶ Maximum deviation  $< L$

where we set a minimum threshold  $L$ .

- If  $d(V^1, V^2) \leq L$ , stop.
- If  $d(V^1, V^2) > L$ , repeat 3 and 4 until convergence.

# Roadmap

---

- ① Example: Savings problem
- ② Value function iteration.
- ③ **Policy function iteration.**
- ④ Example in Matlab: Stochastic Discrete Cake-Eating

## Policy function iteration (1)

---

- Iterate on policy instead of value function.
- Suppose you have an initial guess for the policy  $c(w)$  :

$$c(w) = \begin{bmatrix} c(w^1) \\ c(w^2) \\ \dots \\ c(w^n) \end{bmatrix}$$

- Now you have a system with  $n$  equations and  $n$  unknowns:

$$V(w^1) = u(c(w^1)) + \beta \sum_{i=L,H} \pi_i V [R(w^1 - c(w^1)) + y^i]$$

$$V(w^2) = u(c(w^2)) + \beta \sum_{i=L,H} \pi_i V [R(w^2 - c(w^2)) + y^i]$$

...

$$V(w^n) = u(c(w^n)) + \beta \sum_{i=L,H} \pi_i V [R(w^n - c(w^n)) + y^i]$$

- The  $n$  unknown are  $V(w^1)$ ,  $V(w^2)$ , ...,  $V(w^n)$

# Roadmap

---

- ① Example: Savings problem
- ② Value function iteration.
- ③ Policy function iteration.
- ④ **Example in Matlab: Stochastic Discrete Cake-Eating**

# Stochastic Discrete Cake-Eating: Setup

---

- From Adda & Cooper, p. 46, simpler version here.
- An agent is endowed with a cake of size  $C$ .
- In each period the agent decides to eat the entire cake (and receive utility  $u(C)$ ) or wait. Future is discounted at rate  $\beta$ .
- Even if not eaten, the cake shrinks by a factor  $\rho$  each period.
- Agent experiences *iid* taste shocks  $z$  which take a finite number of values.
- Timing:
  - ① Observe taste shock.
  - ② Decide to eat or postpone before observing next period's shock.
- Even though cake is shrinking, agent might wait for a better realization of the taste shock.

# Stochastic Discrete Cake-Eating: Recursive

---

- The program of the agent can be written as:

$$V(C, z) = \max \{V^E(C, z), V^W(C, z)\}$$

- ▶ value of eating:  $V^E(C, z) = zu(C)$
  - ▶ value of waiting:  $V^W(C, z) = \beta\mathbb{E}[V(\rho C, z')]$
- Policy is a step function (not continuous):

$$d(C, z) \in \{0, 1\}$$

where 1 = *eat* and 0 = *wait*.

- We can define threshold  $z^*(C)$  such that

$$d(C, z) = \begin{cases} 1 & \text{if } z \geq z^*(C) \\ 0 & \text{if } z < z^*(C) \end{cases}$$

# Matlab: Preamble

---

- Name and date.
- Define main elements of the problem.

```
% Author: Isaac Baley
% Advanced Macro II
% Date: Jan 15th, 2016
% Description: Discrete stochastic cake eating problem.
    % Agent chooses to either eat the cake (E) or to wait (W).
    % There are taste shocks which follow a Markov process.
    % The cake does not change size.

% Define values
% E(C,z) = z*u(C)           = value of eating cake
% W(C,z) = beta*Q*V         = value of waiting
% V(C,z) = max {E(C,z) , W(C,z)} = value of having a cake

% policy = eat cake if E>W

clear all
close all
clc
```

# Matlab: Parameters

---

```
%% =====  
%   Initialize parameters  
% =====  
  
% CRRA Utility  $u(c)=c^{(1-\sigma)}/(1-\sigma)$ , if  $\sigma=1$  then log utility  
sigma = 1           ;  
beta  = 0.97       ;  
  
% risk aversion  
% discount factor (impatience)
```

# Matlab: Define state space

---

```
%% =====  
% Define state-space  
% =====  
  
% Cake space  
C = 100 ;  
n_C = length(C) ;  
  
% Taste shocks and transition matrix  
z = [0.75 1.0 1.25] ;  
n_z = length(z) ;  
Q = [ 0.90 0.05 0.05 ;  
      0.05 0.90 0.05 ;  
      0.05 0.05 0.90] ;  
  
% Initialize value functions with matrix of zeros  
%E= zeros(n_z,n_C);  
%W= zeros(n_z,n_C);  
%V= zeros(n_z,n_C);  
  
% Iteration parameters  
it_max = 1000 ;  
it_tol = 0.00001 ;
```

```
% Cake size (but policy function should be independent of C)  
% Only one number since it will not change size  
  
% space for taste shock, 3 shocks = low, med, hi  
% number of taste shocks  
% n_z X n_z transition matrix  
% today's state is row, tomorrow's state is column  
  
% Maximum number of iterations  
% Tolerance for convergence
```

# Matlab: Initialize value functions

---

```
%% =====  
% Initialize value functions  
% =====  
  
% E comes from eating the cake, it is terminal, and so is static  
for i = 1:n_z  
    if sigma==1 % log case  
        E(i,:) = z(i)*log(C);  
    else % general CRRA case  
        E(i,:) = z(i)*C^(1-sigma)/(1-sigma);  
    end  
end  
  
% W needs to be guessed:  
W_0 = beta*Q*E;  
% V also to be guessed:  
V_0 = max(E,W_0);  
  
% Guess you will eat the cake next period  
% Take max to construct initial V based on W guess
```

# Matlab: Value Function Iteration

---

```
%% =====  
% Value Function Iteration  
% =====  
  
it = 1 ;  
dif = 1000 ;  
  
while (it<it_max && dif > it_tol)  
    W_1 = beta*Q*V_0 ;  
    V_1 = max(E,W_1) ;  
    dif = max(abs(V_1-V_0));  
    V_0 = V_1 ;  
    it = it+1 ;  
end  
disp('number of iterations')  
it  
V = V_1;  
W = W_1;
```

% Initialize counter  
% Initialize difference between functions  
  
% update value of waiting given guess for V  
% update value function given update for W  
% check difference between value functions  
% update guess (always update after checking the diff!)

# Matlab: Compute policy and print results

```
%% =====  
% Compute policy  
% =====  
policy = (E>W);  
disp('----- RESULTS -----')  
disp('   shock   policy   V       E       W')  
res=[z' policy V E W];  
disp(res)
```

```
Command Window  
Stochastic Cake Eating problem  
number of iterations for convergence  
it =  
    66  
difference between V_0-V_1  
dif =  
    9.3628e-06  
----- RESULTS -----  
   shock   policy   V       E       W  
    0.7500     0     3.9569   3.4539   3.9569  
    1.0000    1.0000   4.6052   4.6052   4.4914  
    1.2500    1.0000   5.7565   5.7565   5.4407
```

## Matlab: Make some plots

---

```
%% =====  
% Plots  
% =====  
close all  
plot(z,E,'b',z,W,'r-o',z,V,'k--','linewidth',5)  
set(gca,'FontSize',18)  
title('Cake Eating with Taste Shocks')  
xlabel('Taste Shocks')  
ylabel('Values')  
legend('Eating','Waiting','Having a cake','location','northwest')
```

## Matlab: Plot of value function

---

