Macroeconomics I

University of Tokyo

Lecture 7

Dynamic Programming III: Practical DP

LS, Chapter 4

Julen Esteban-Pretel
National Graduate Institute for Policy Studies

Deterministic Problem

- Household's consumption/saving problem (sequential formulation).

(S.P.)
$$\begin{aligned} &\max_{\{c_t, a_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t u(c_t), \quad 0 < \beta < 1, \\ \text{s.t.} \quad & a_{t+1} = (1+r)a_t + w - c_t, \\ & c_t \geq 0, \ a_{t+1} \geq 0. \qquad \text{for } t \geq 0 \end{aligned} \tag{7.1}$$

where c_t is consumption in t,

a_t are assets holdings in t,

r is the 1-period return on the asset,

w is income.

- Assume $u(\ \cdot\ )$ is strictly concave.
- No solution without $a_{t+1} \geq 0$.
- Assume $\beta \leq \frac{1}{1+r}$. Otherwise there may not be a solution for some $u(\ \cdot\ )$.

  (i.e., no solution if $\beta \geq \frac{1}{1+r}$ and $u(c) = c^{\gamma}$).

Bellman Equation

- The Bellman equation for the previous problem is:

$$\text{(D.P.)} \quad \begin{aligned} v(a) &= \max_{a'} \left\{ u[(1+r)a + w - a'] + \beta v(a') \right\}, \\ &\text{s.t. } \; 0 \leq a' \leq (1+r)a + w. \end{aligned} \tag{7.2}$$

  - *State variable*: a (current asset holdings).
  - *Control variable*: a' (next period's asset holdings).
- We will compute the value function by discretizing the state space for a.
- Let $\mathcal{A} = (\bar{a}_1, \ldots, \bar{a}_n)$ be a finite set.
- The value function can be represented as an n-dimensional vector $\underset{(n \times 1)}{v}$. Its *i-th* element is v_i. So $v_i = v(\bar{a}_i)$.

$$v = \begin{bmatrix} v_1 \\ \vdots \\ v_i \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} v(\bar{a}_1) \\ \vdots \\ v(\bar{a}_i) \\ \vdots \\ v(\bar{a}_n) \end{bmatrix}.$$

Bellman Equation with Discretized State Space

- Restating the Bellman equation:

$$v(a) = \max_{a'} \left\{ u[(1+r)a + w - a'] + \beta v(a') \right\}, \tag{7.3}$$

- With $\mathcal{A} = (\bar{a}_1, \ldots, \bar{a}_n)$, the Bellman equation becomes

$$v_i = \max_{h \in \{1,2,\ldots,n\}} \{u[(1+r)\bar{a}_i + w - \bar{a}_h] + \beta v_h\}, \quad i = 1, 2, \ldots, n. \tag{7.4}$$

where i is the index for current asset holdings,
h is the index for next period's asset holdings.

- Let $R_{ih} \equiv u[(1+r)\bar{a}_i + w - \bar{a}_h]$, and $\underset{(n\times 1)}{v}$ be the vector whose *i-th* element is v_i.

$$v_i = \max \left\{ \underset{(1\times n)}{R_{i,\bullet}} + \beta \underset{(1\times n)}{v'} \right\}, \quad i = 1, 2, \ldots, n. \tag{7.5}$$

where "max" is the "largest element", and $R_{i,\bullet} = (R_{i1}, R_{i2}, \ldots, R_{in})$.

Iteration on the Bellman Equation

- For each state the Bellman equation is:

$$v_i = max \; \{(R_{i1}, R_{i2}, \ldots, R_{in}) + \beta(v_1, v_2, \ldots, v_n)\}, \quad i = 1, 2, \ldots, n. \tag{7.6}$$

- In matrix notation

$$\underset{(n\times 1)}{v} = max \left\{ \underset{(n\times n)}{R_{\bullet,\bullet}} + \beta \underset{(n\times 1)}{1_n} \underset{(1\times n)}{v'} \right\} \tag{7.7}$$

- This defines an iteration on $\underset{(n\times 1)}{v}$:

$$v^{(\ell+1)} = max \left\{ R_{\bullet,\bullet} + \beta 1_n v^{(\ell)'} \right\}. \tag{7.8}$$

- Given v, the Bellman equation (7.7) defines the policy function, which is a mapping from current asset holdings (*a*) to next period's holdings (*a*').

Non-negativity Constraints

- Reproducing equation (7.7)

$$\underset{(n\times 1)}{v} = max \left\{ \underset{(n\times n)}{R_{\bullet,\bullet}} + \beta \underset{(n\times 1)}{1_n} \underset{(1\times n)}{v'} \right\} \tag{7.7}$$

- Recall that $R_{ih} \equiv u[(1+r)\bar{a}_i + w - \bar{a}_h]$.
- What about $0 \leq a' \leq (1+r)a + w$?
  - Not including negative numbers in $\mathcal{A}$ takes care of $0 \leq a'$.
  - The non-negativity constraint on consumption (i.e., $(1+r)a + w - a' \geq 0$) can be taken care of by assigning a large negative number to R_{ij} when $(1+r)a + w - a' < 0$.

Matlab Implementation - Preliminaries

$$v(a) = \max_{a'} \left\{ u[(1+r)a + w - a'] + \beta v(a') \right\}$$

- 1.- Discretize asset space: $\mathcal{A} = (\bar{a}_1, ..., \bar{a}_n)$

```
% Discretize the state space
% State variable: Assets in current period (a)
a   = (0:0.05:20)';       % State space for assets
n   = size(a,1);          % n is the number of states for assets
```

- 2.- Choose parameter values

```
% Preference parameters
bet = 0.95;               % Discount factor
sig = 1;                  % Degree of risk aversion in the CRRA utility function
% Technological parameters
r   = 0.04;               % Rate of return on assets
w   = 1;                  % wage rate
% Iteration procedure elements
MxIt= 1000;               % Maximum number of iterations
eps = 0.001;              % Convergence criterium
```

- 3.- Calculate matrix of current returns $\underset{(n\times n)}{R}$, where $R_{ih} \equiv u[(1+r)\bar{a}_i + w - \bar{a}_h]$

```
R=zeros(n,n);           % Initialize the return matrix R (n by n)
% Calculate matrix of consumption (n by n) for assets states (i,h) [current,future]
 C=(1+r)*a*ones(1,n)+w*ones(n,n)-ones(n,1)*a';
% Calculate matrix of current returns for each c (if cons. is neg. util = -999)
 R=UtilFn(C.*(C>=0),sig).*(C>=0)-999*(C<0);
```

Matlab Implementation - Value Function Iteration

$$v(a) = \max_{a'} \left\{ u[(1+r)a + w - a'] + \beta v(a') \right\}, \text{ or } \underset{(n\times 1)}{v} = \max \left\{ \underset{(n\times n)}{R_{\bullet,\bullet}} + \beta \underset{(n\times 1)}{1_n} \underset{(1\times n)}{v'} \right\}$$

- 4.- Iterate on the bellman equation until it converges:

```
V = ones(n,1); Vnew = ones(n,1); Pol = ones(n,1); % Initialize value fns, pol. fn
for l=1:MxIt;                        % Iterate until the max iterations has been reached
```

  - 4.1.- Calculate RHS of the Bellman eq before finding the max: $R + \beta 1_n v'$

```
W=R+bet*ones(n,1)*V'; % Calculates RHS of Bellman eq for i and h.
```

  - 4.2.- Calculate the max (largest element), the LHS of the Bellman eq and the policy fn:

```
[W,index]=max(W');        % Find the max over h (next period asset)
```

  - 4.3.- LHS of the Bellman equation and policy function:

```
Vnew=W';                  % LHS of Bellman eq
Pol=index';               % Calculates the policy fn (choice of h for each i)
```

  - 4.4.- Check for convergence:

```
% Check for convergence
if max(abs(Vnew-V))<eps;
    fprintf(1,'The value function has CONVERGED in %2.0f iterations.\n',l);
    break;                % If convergence if achieved (change is less than eps) stop loop
end;
```

  - 4.5.- If no convergence, update value function and continue until convergence:

```
    V=Vnew;               % Update value function
end;
```

Matlab Implementation - Plot Results

$$v(a) = \max_{a'} \left\{ u[(1+r)a + w - a'] + \beta v(a') \right\}, \text{ or } \underset{(n\times 1)}{v} = \max \left\{ \underset{(n\times n)}{R_{\bullet,\bullet}} + \beta \underset{(n\times 1)}{1_n} \underset{(1\times n)}{v'} \right\}$$

- 5.- Plot the value and policy functions:

```
% value function
subplot(2,1,1)
plot(a,V,'b-')
title('Value function');
xlabel('Current asset level');

% policy function
subplot(2,1,2)
plot(a,a(Pol),'b-')
title('Policy function');
xlabel('Current asset level');
ylabel('Next period asset level');
```

Stochastic Case

- Household's consumption/saving problem (sequential formulation).

$$\text{(S.P.)} \quad \max_{\{c_t, a_{t+1}\}_{t=0}^{\infty}} E_0 \left\{ \sum_{t=0}^{\infty} \beta^t u(c_t) \right\}, \quad 0 < \beta < 1,$$

$$\text{s.t.} \quad a_{t+1} = (1+r)a_t + ws_t - c_t, \qquad \text{for } t \geq 0 \tag{7.9}$$

$$c_t \geq 0, \; a_{t+1} \geq 0.$$

where $\{s_t\}$ follows a Markov process.

s_t can be interpreted as productivity or the employment status.

- Assume that $\beta \leq \dfrac{1}{1+r}$.

Bellman Equation and Discretization of Problem

- The dynamic programming formulation, Bellman equation, of the problem is

(D.P.)
$$v(a,s) = \max_{a'} \left\{ u[(1+r)a + ws - a'] + \beta E[v(a',s')|a,s] \right\}, \quad (7.10)$$
$$\text{s.t.} \;\; 0 \le a' \le (1+r)a + ws.$$

- Assume $\{s_t\}$ is a finite-state Markov chain with transition matrix $\underset{(m\times m)}{P}$ and state space $\mathcal{S} = (\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_m)$.

- $v(a,s)$ can be represented as a collection of functions of a single variable

$$\underset{(n\times m)}{v(a,s)} = \left(\underset{(n\times 1)}{v_1(a)}, \ldots, \underset{(n\times 1)}{v_j(a)}, \ldots, \underset{(n\times 1)}{v_m(a)} \right)$$ where the *i-th* element of v_j is v_{ij}.

- Under these assumptions, we can write the Bellman equation as

$$v_j(a) = \max_{a'} \left\{ u[(1+r)a + w\bar{s}_j - a'] + \beta \sum_{k=1}^{m} P_{jk} v_k(a') \right\}, \quad (7.11)$$
$$\text{s.t.} \;\; 0 \le a' \le (1+r)a + w\bar{s}_j, \;\; \text{for } j = 1, 2, \ldots, m.$$

Bellman with Discretized State Space

- Restating the Bellman equation (7.11):

$$v_j(a) = \max_{a'} \left\{ u[(1+r)a + w\bar{s}_j - a'] + \beta \sum_{k=1}^{m} P_{jk}\, v_k(a') \right\}, \quad j = 1, 2, \ldots, m.$$

- With $\mathcal{A} = (\bar{a}_1, \ldots, \bar{a}_n)$, the Bellman equation becomes

$$v_{ij} = \max_{h \in \{1,2,\ldots,n\}} \left\{ u[(1+r)\bar{a}_i + w\bar{s}_j - \bar{a}_h] + \beta \sum_{k=1}^{m} P_{jk}\, v_{hk} \right\}. \qquad (7.12)$$

where i is the index for current asset holdings,

h is the index for next period's asset holdings,

j is the index for employment status.

- Let $R_{ihj} \equiv u[(1+r)\bar{a}_i + w\bar{s}_j - \bar{a}_h]$, and $\underset{(n\times m)}{V}$ be the vector whose *(i,j)* element is v_{ij}.

$$v_{ij} = \max \left\{ \underset{(1\times n)}{R_{i,\bullet,j}} + \beta \underset{(1\times m)}{P_{j,\bullet}} \underset{(m\times n)}{V'} \right\}, \qquad (7.13)$$

"max" is the "largest element".

where $\underset{(n\times m)}{V} = [\underset{(n\times 1)}{v_1}, \ldots, \underset{(n\times 1)}{v_j}, \ldots, \underset{(n\times 1)}{v_m}].$

Bellman using Matrix Algebra

- Writing equation (7.13) in full:

$$v_{ij} = max \left\{ \underset{(1\times n)}{R_{i,\bullet,j}} + \beta \underset{(1\times m)}{P_{j,\bullet}} \underset{(m\times n)}{V'} \right\}$$

$$= max \left[R_{i1j} + \beta \sum_{k=1}^{m} P_{jk}\, v_{1k} \;\cdots\; R_{ihj} + \beta \sum_{k=1}^{m} P_{jk}\, v_{hk} \cdots\; R_{inj} + \beta \sum_{k=1}^{m} P_{jk}\, v_{nk} \right]$$

- " $\underset{(1\times m)}{P_{j,\bullet}} \underset{(m\times n)}{V'}$ " does not depend on *i*. Hence (restating LS (4.3.1)):

$$\underset{(n\times 1)}{v_j} = max \left\{ \underset{(n\times n)}{R_{\bullet,\bullet,j}} + \beta \underset{(n\times 1)}{1_n} \underset{(1\times m)}{P_{j,\bullet}} \underset{(m\times n)}{V'} \right\}, \quad j = 1, 2, \ldots, m. \qquad (7.14)$$

"max A" is a column vector whose i-th element is the largest element of the i-th row of A.

Non-negativity Constraints

- Reproducing equation (7.14)

$$\underset{(n\times 1)}{v_j} = max \left\{ \underset{(n\times n)}{R_{\bullet,\bullet,j}} + \beta \underset{(n\times 1)}{1_n} \underset{(1\times m)}{P_{j,\bullet}} \underset{(m\times n)}{V'} \right\}, \quad j = 1, 2, \ldots, m.$$

- Recall that $R_{ihj} \equiv u\big[(1 + r)\bar{a}_i + w\bar{s}_j - \bar{a}_h\big]$.
- What about $0 \le a' \le (1+r)a + w\bar{s}$?
  - Not including negative numbers in $\mathcal{A}$ takes care of $0 \le a'$.
  - The non-negativity constraint on consumption (i.e., $(1+r)a + ws_j - a' \ge 0$) can be taken care of by assigning a large negative number to R_{ihj} when $(1+r)a + ws_j - a' < 0$.

Matlab Implementation - Preliminaries

$$v(a,s) = \max_{a'} \left\{ u[(1+r)a + ws - a'] + \beta E[v(a',s')|a,s] \right\}$$

- 1.- Discretize asset space: $\mathcal{A} = (\bar{a}_1, \ldots, \bar{a}_n)$

```
a    = (0:0.05:20)';        % State space for assets
n    = size(a,1);           % n is the number of states for assets
```

- 2.- Specify Markov chain: $\mathcal{S} = (\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_m)$ and P.

```
s    = [0.1;1];             % State space for employment
P    = [0.6, 0.4;0.3,0.7];  % Markov transition matrix
m    = size(s,1);           % m is the number of employment states
```

- 3.- Choose parameter values

```
bet = 0.95;   sig = 1;      % Discount factor and degree of risk aversion.
r   = 0.04;   w   = 1;      % Rate of return on assets and wage rate when employed.
MxIt= 1000;   eps = 0.001;  % Maximum number of iterations and Convergence criterium
```

- 4.- Calculate matrix of current returns $\underset{(n\times n\times j)}{R}$, where $R_{ihj} \equiv u[(1+r)\bar{a}_i + w\bar{s}_j - \bar{a}_h]$

```
R=zeros(n,n,m);          % Initialize R, which is n by n by j (j is the emp. state)
% Calculate the matrix of current returns for each j (employment state)
for j=1:m;               % Calculation for each j (employment state)
    % Calculate matrix of consumption (n by n) for assets states (i,h) [current,future] given j
    C=(1+r)*a*ones(1,n)+w*s(j)*ones(n,n)-ones(n,1)*a';
    % Calculate matrix of current returns for each c (if cons. is neg. util = -999)
    R(:,:,j)=UtilFn(C.*(C>=0),sig).*(C>=0)-999*(C<0);
end;
```

Matlab Implementation - Value Function Iteration

$$v(a,s) = \max_{a'} \left\{ u[(1+r)a + ws - a'] + \beta E[v(a',s')|a,s] \right\} \text{ or } \underset{(n\times 1)}{v_j} = \max \left\{ \underset{(n\times n)}{R_{\bullet,\bullet,j}} + \beta \underset{(n\times 1)}{1_n} \underset{(1\times m)}{P_{j,\bullet}} \underset{(m\times n)}{V'} \right\}$$

for $j = 1,2,\ldots,m$

- **5.- Iterate** on the bellman equation until it converges:

```
V = ones(n,m); Vnew = ones(n,m); Pol = ones(n,m); % Initialize value fns, pol. fn
for l=1:MxIt;                       % Iterate until the max iterations has been reached
    for j=1:m;                      % Calculation for each j (employment status)
```

  - **5.1.-** Calculate RHS of the Bellman eq before finding the max: $R + \beta 1_n P_{j,\bullet} V'$

```
        W=R(:,:,j)+bet*ones(n,1)*P(j,:)*V'; % RHS of Bellman eq for i and h, given j.
```

  - **5.2.-** Calculate the max (largest element), the LHS of the Bellman eq and the policy fn:

```
        [W,index]=max(W');          % Find the max over h (next period asset)
```

  - **5.3.-** LHS of the Bellman equation and policy function:

```
        Vnew=W';                    % LHS of Bellman eq
        Pol=index';                 % Calculates policy fn (choice of h for each i, given j)
    end;
```

  - **5.4.-** Check for convergence:

```
    if max((max(abs(Vnew-V)))')<eps; break;
        fprintf(1,'The value function has CONVERGED in %2.0f iterations.\n',l);
    end;
```

  - **5.5.-** If no convergence, update value function and continue until convergence:

```
    V=Vnew;                         % Update value function
end;
```

Matlab Implementation - Plot Results

$$v(a,s) = \max_{a'} \left\{ u[(1+r)a + ws - a'] + \beta E[v(a',s')|a,s] \right\} \text{ or } \underset{(n\times 1)}{v_j} = \max \left\{ \underset{(n\times n)}{R_{\bullet,\bullet,j}} + \beta \underset{(n\times 1)}{1_n} \underset{(1\times m)}{P_{j,\bullet}} \underset{(m\times n)}{V'} \right\}$$

for $j = 1,2,\ldots,m$

- **6.- Plot** the value and policy functions:

```
% value function
subplot(2,1,1)
plot(a,V(:,1),'r:',a,V(:,2),'b-')
title('Value function');
legend('Unemployed','Employed',4);
xlabel('Current asset level');

% policy function
subplot(2,1,2)
plot(a,a(Pol(:,1)),'r:',a,a(Pol(:,2)),'b-')
title('Policy function');
legend('Unemployed','Employed',4)
xlabel('current asset level');
ylabel('Next period asset level');
```

Matlab Implementation - Time Series Simulation

$$v(a,s) = \max_{a'} \left\{ u[(1+r)a + ws - a'] + \beta E[v(a',s')|a,s] \right\} \text{ or } \underset{(n\times 1)}{v_j} = \max \left\{ \underset{(n\times n)}{R_{\bullet,\bullet,j}} + \beta \underset{(n\times 1)}{1_n} \underset{(1\times m)}{P_{j,\bullet}} \underset{(m\times n)}{V'} \right\}$$

for $j = 1,2,\ldots,m$

- 7.- Simulation preliminaries:

```
maxT = 5000;                                  % Max periods
initial_emp = 1;                              % Initial employment state is 1
initial_asset = floor(n/3);                   % Initial assets state
[chain,state_emp] = markov(P,maxT,initial_emp,s');  % Generate Markov chain
state_emp = (1:1:m)*state_emp;                % Employment state for t=2,...,T.
state_emp = [initial_emp state_emp];          % add employment state for t=1
```

- 8.- Calculate the time series for consumption and assets:

```
c     = zeros(maxT,1);    % Initialize consumption series
asset = zeros(maxT,1);    % Initialize asset level series
i     = initial_asset;    % Initialize the asset state

for t=1:maxT;
    j=state_emp(t);        % Employment state in t is j.
    h=Pol(i,j);            % Policy rule states to move from i to h.
    c(t)=(1+r)*a(i)+w*s(j)-a(h); % Implied consumption
    asset(t)=a(i);         % Translate asset state i to asset level
    i=h;                   % Update the current asset state
end;

euler = zeros(maxT-1,1); % Euler error
for t=1:maxT-1;
    euler(t)=bet*(1+r)*(MgUtilFn(c(t+1),sig)./MgUtilFn(c(t),sig))-1;
end;
```

Matlab Implementation - Plot Simulation Results

$$v(a,s) = \max_{a'} \left\{ u[(1+r)a + ws - a'] + \beta E[v(a',s')|a,s] \right\} \text{ or } \underset{(n\times 1)}{v_j} = \max \left\{ \underset{(n\times n)}{R_{\bullet,\bullet,j}} + \beta \underset{(n\times 1)}{1_n} \underset{(1\times m)}{P_{j,\bullet}} \underset{(m\times n)}{V'} \right\}$$

for $j = 1,2,\ldots,m$

- 9.- Plot Simulation Results:

```
T = 100;

% Employment state
subplot(4,1,1)
x = (1:1:T-1);
bar(x,state_emp(1:T-1)-1,1)
title('Employment status');
xlabel('Period');

% Assets
subplot(4,1,2)
x=(1:1:T);
plot(x,asset(1:T),'b-'),title('Asset path');
xlabel('Period');

% Consumption
subplot(4,1,3)
x=(1:1:T);
plot(x,c(1:T),'b-'),title('Consumption path');
xlabel('Period');

% Euler error
subplot(4,1,4)
x=(1:1:T);
plot(x,euler(1:T),'b-'),title('Euler error');
xlabel('Period');
```

Matlab Implementation - Plot Simulation Results

$$v(a,s) = \max_{a'} \left\{ u[(1+r)a + ws - a'] + \beta E[v(a',s')|a,s] \right\} \text{ or } \underset{(n\times 1)}{v_j} = \max \left\{ \underset{(n\times n)}{R_{\bullet,\bullet,j}} + \beta \underset{(n\times 1)}{1_n} \underset{(1\times m)}{P_{j,\bullet}} \underset{(m\times n)}{V'} \right\}$$

for $j = 1,2,...,m$

- 9.- Plot Simulation Results:

```
% Scattterplot current Euler error on as a function of current assets

subplot(1,1,1)
x=asset(1:maxT-1);
scatter(x,euler,10,'*');
xlabel('Current assets');
ylabel('Euler error');
```