Computational Macroeconomics

Dr Panayiotis M. Pourpourides
Dynamic Programming

1 The value function

Let x_t denote a vector of deterministic state variables, u_t denote a vector of control variables and $R(x_t, u_t)$ denote the return function of an agent. Suppose the state vector evolves according to $x_{t+1} = g(x_t, u_t)$. Then, the dynamic problem of the agent can be written as

$$V(x_t) = \max_{u_t} \{R(x_t, u_t) + \beta V(x_{t+1})\} \tag{1}$$

$$\begin{gathered} \text{subject to} \\ x_{t+1} = g(x_t, u_t) \end{gathered} \tag{2}$$

where β is the discount factor and V is the value function.

Now suppose there is also another state vector z_t which consists of state variables determined by nature - these are stochastic state variables. Then, the stochastic representation of the dynamic problem is

$$V(x_t, z_t) = \max_{u_t} \{R(x_t, z_t, u_t) + \beta E_t V(x_{t+1}, z_{t+1})\} \tag{3}$$

$$\begin{gathered} \text{subject to} \\ x_{t+1} = g(x_t, z_t, u_t) \end{gathered} \tag{4}$$

where E_t is the expectation operator which is defined over the sequence of shocks $\{z_j\}_{j=t}^{\infty}$.

1.1 An example - The Neoclassical Growth Model (stochastic case)

Consider the Brock and Mirman (1972) model.[1] Let $R(x_t, z_t, u_t)$ denote the utility function of a representative agent:

$$u(c_t) = \ln c_t$$

Note that in this example it happens that the return function depends only on the control (choice) variable. Furthermore, suppose that output, y_t, is produced by the following production technology

$$f(k_t, \theta_t) = \theta_t k_t^{\alpha}, \ \ \alpha \in (0,1) \tag{5}$$

where θ_t and k_t denote the level of technology and physical capital, respectively. Capital evolves according to the standard law of motion:

$$k_{t+1} = (1-\delta) k_t + i_t \tag{6}$$

where δ is the depreciation rate and i_t is investment. Assume that θ_t is a stochastic which evolves according to the following $AR(1)$ process:

$$\ln \theta_t = (1-\rho) \ln \overline{\theta} + \rho \ln \theta_{t-1} + \varepsilon_t \tag{7}$$

where $\rho \in (0,1)$ which implies that θ_t is stationary and $\varepsilon_t \tilde{} iidN(0,1)$. The resource constraint is

$$y_t = c_t + i_t \tag{8}$$

[1] Brock, W.A., Mirman, L., 1972. Optimal Economic Growth and Uncertainty: The Discounted Case. Journal of Economic Theory, 4 (3), 479-513.

For simplicity we will assume 100% depreciation (i.e $\delta = 1$). Then, using (5) and (6), (8) reduces to

$$k_{t+1} = \theta_t k_t^{\alpha} - c_t \tag{9}$$

Notice that $g(x_t, z_t, u_t) \equiv \theta_t k_t^{\alpha} - c_t$ where $x_t = k_t$, $z_t = \theta_t$ and $u_t = c_t$. The problem of the representative agent is to maximize his lifetime utility subject to the resrouce constraint:

$$E_t \sum_{j=t}^{\infty} \beta^{j-t} \ln c_j$$
$$\text{subject to } k_{t+1} = \theta_t k_t^{\alpha} - c_t$$

The value function representation of this problem is

$$V(k, \theta) = \max_c \{\ln c_t + \beta E V(k', \theta')\}$$
$$\text{subject to } k' = \theta k^{\alpha} - c$$

where the prime ($'$) denotes next period. It is more convenient to substitute out c using the resource constraint and deal with the following problem

$$V(k, \theta) = \max_{k'} \{\ln(\theta k^{\alpha} - k') + \beta E V(k', \theta')\} \tag{10}$$

This problem can be solved analytically. The first-order condition is

$$\frac{1}{\theta k^{\alpha} - k'} = \beta E V_k(k', \theta') \tag{11}$$

By the envelope theorem we obtain

$$V_k(k, \theta) = \frac{\alpha \theta k^{\alpha - 1}}{\theta k^{\alpha} - k'} \tag{12}$$

Proof of (12): Let the policy rule be $k' = \phi(k, \theta)$. Use the latter to substitute out k' from

(10):

$$V(k,\theta) = \ln\left(\theta k^{\alpha} - \phi(k,\theta)\right) + \beta E V\left(\phi(k,\theta), \theta'\right)$$

Maximize with respect to k:

$$V_k(k,\theta) = \frac{1}{\theta k^{\alpha} - k'}\left(\theta\alpha k^{\alpha-1} - \phi_k(k,\theta)\right) + \beta E V_k(k',\theta')\,\phi_k(k,\theta)$$

Rearranging

$$V_k(k,\theta) = -\phi_k(k,\theta)\underbrace{\left[\frac{1}{\theta k^{\alpha} - k'} - \beta E V_k(k',\theta')\right]}_{\text{zero from the foc}} + \frac{\theta\alpha k^{\alpha-1}}{\theta k^{\alpha} - k'}$$

$$\Longrightarrow V_k(k,\theta) = \frac{\alpha\theta k^{\alpha-1}}{\theta k^{\alpha} - k'}$$

Using (12) in (11), the Euler equation becomes

$$\frac{1}{\theta k^{\alpha} - k'} = \beta E \frac{\alpha\theta\left(k'\right)^{\alpha-1}}{\theta\left(k'\right)^{\alpha} - k''} \tag{13}$$

where double prime denotes two periods ahead.

Guess and verify $\phi(k,\theta)$:

$$k' = \beta\alpha\theta k^{\alpha} \tag{14}$$

Then,

$$c = (1 - \beta\alpha)\,\theta k^{\alpha} \tag{15}$$

It can also be shown that

$$V(k,\theta) = \tfrac{\alpha}{(1-\beta\alpha)}\ln k + \tfrac{1}{(1-\beta\alpha)}\tfrac{1}{(1-\beta\rho)}\ln\theta +$$
$$\tfrac{1}{1-\beta}\ln(1-\beta\alpha) + \tfrac{\beta}{(1-\beta\rho)}\tfrac{1-\rho}{(1-\beta\alpha)}\ln\overline{\theta} + \beta\tfrac{\alpha}{(1-\beta\alpha)}\ln\beta\alpha$$

- The model above can be solved analytically due to our assumptions of log utility and

full depreciation ($\delta = 1$). However, when we deviate from these assumptions we can no longer solve analytically. So instead we use numerical methods to compute the value function and the policy rule for capital.

1.2 Numerical method for the value function

1.2.1 Markov Chains

Let us begin by discussing properties of the stochastic process, z_t. It is often assumed that a stochastic process satisfies the *Markov property.*[2] That is, the probability of a particular realization of z' depends only on the previous period's realization of z. Formally, a stochastic process $\{z_t\}$ is said to have the Markov property if for all $d \geq 2$ and all t,

$$\text{Prob}\left(z_{t+1} \mid z_t, z_{t-1}, ..., z_{t-d}\right) = \text{Prob}\left(z_{t+1} \mid z_t\right)$$

Then, a stochastic process that satisfies the Markov property is called a *Markov chain.* A time-invariant Markov chain is defined by a triple of objects: an n-dimensional vector $\overline{z} \in \Re^n$ that records the possible values of the *state* of the system, an n x n *transition matrix* Π, which records the probabilities of moving from one value of the state to another in one period, and an n x 1 vector ξ_0 recording the probabilities of being in state i at time 0:

$$\Pi_{ij} = \text{Prob}\left(z_{t+1} = \overline{z}_j \mid z_t = \overline{z}_i\right)$$

$$\xi_{0i} = Prob\left(z_0 = \overline{z}_i\right)$$

Furthermore, to be valid Π and ξ must satisfy:

$$\sum_{j=1}^{n} \Pi_{ij} = 1 \qquad \text{(P1)}$$

[2]The discussion follows that of Ljungqvist and Sargent, Recursive Macroeconomic Theory.

$$\sum_{i=1}^{n} \xi_{0i} = 1 \tag{P2}$$

Thus, element Π_{ij} of matrix Π is the probability from moving from state $\overline{z}_i$ to state $\overline{z}_j$. The probability of moving from any value of the state to any other two periods is determined by Π^2 because

$$\begin{aligned}\text{Prob}\left(z_{t+2} = \overline{z}_j \mid z_t = \overline{z}_i\right) &= \textstyle\sum_{h=1}^{n} \text{Prob}\left(z_{t+2} = \overline{z}_j \mid z_{t+1} = \overline{z}_h\right) \text{Prob}\left(z_{t+1} = \overline{z}_h \mid z_t = \overline{z}_i\right) \\ &= \textstyle\sum_{h=1}^{n} \Pi_{ih}\Pi_{hj} = \Pi_{ij}^{(2)}\end{aligned}$$

where $\Pi_{ij}^{(2)}$ is the i,j element of matrix Π^2.

Tauchen (1986), shows that we can approximate an autoregressive process by a markov chain described above. In particular, in his article in Economics Letters (1986), he developes a procedure for finding a discrete-valued Markov chain whose sample paths approximate well those of a vector autoregression.

1.2.2 Value function

Recall that our problem is to solve

$$V(x,z) = \max_{x'} \left\{R(x,x',z) + \beta EV(x',z')\right\}$$

where x and z are two single variables. Define the state vector $s = [x,z]'$ and assume that z and k can only take a finite number of values:

$$z \in Z = \left\{z_1, z_2, ..., z_{n^z}\right\}$$

$$x \in X = \left\{x_1, x_2, ..., x_{n^x}\right\}$$

In this way we have *discretized* the state space; the different values for z and x are the *grid points*..Contrary to x, z is an exogenous stochastic variable determined by nature. Assume

that z is a Markov process which is characterized by the transition matrix Π, where $\Pi_{ij} = \text{Prob}(z_{t+1} = \overline{z}_j \mid z_t = \overline{z}_i)$. Recall that if z is an AR process, it can always be approximated as a Markov process using Tauchen's method. For instance, if you assume that z has two states (i.e $n^z = 2$), the transition matrix will be

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

where $\Pi_{11} + \Pi_{12} = 1$ and $\Pi_{21} + \Pi_{22} = 1$.

Consequently, the value function V is an n^z x n^x matrix and the Bellman equation can be written as

$$V_{i,r} = \max_{s \in \{1,2,...,n^x\}} R\left(z_i, x_r, x_s\right) + \beta \sum_{j=1}^{n^z} \Pi_{i,j} V_{j,s} \tag{16}$$

where V_{ij} and Π_{ij} are the ijth elements of matrices V and Π, respectively. How do we find V?

Under certain conditions, iterations on the following value function will converge to V:[3]

$$V_{i,r}^{(n+1)} = \max_{s \in \{1,2,...,n^x\}} R\left(z_i, x_r, x_s\right) + \beta \sum_{j=1}^{n^z} \Pi_{i,j} V_{j,s}^{(n)} \tag{17}$$

The iteration works as follows:

1. Start from $n = 0$ and guess an initial value for $V^{(0)}$. One $V^{(0)}$ is an n^z x n^x matrix of zeros.

2. For each combination of z_i and x_r, calculate the value $V_{i,r,s}^{(n+1)}$ for each possible x_s. The optimal choice for x' is the x_s that delivers the highest value, i.e

$$V_{i,r}^{(n+1)} = \max \left\{ V_{i,r,1}^{(n+1)},\, V_{i,r,2}^{(n+1)},\, ...,\, V_{i,r,n^x}^{(n+1)} \right\}$$

[3] $R\left(x, x', z\right)$ is concave function and the set $\left\{\left(x', x, z\right) : x' \leq g\left(x, z, u\right), u \in \Re^{\phi}\right\}$ is convex and compact - see Ljungqvist and Sargent, chapter 2.

Keep both the value as well as the optimal decision grid point. Construct matrix $V^{(n+1)}$ and compute the distance d^{n+1}, between $V^{(n+1)}$ and $V^{(n)}$. One definition of d is the sup norm:

$$d^{(n+1)} = \max_{j \in \{1,...,n^x\}} \left\| V_j^{(n+1)} - V_j^{(n)} \right\|$$

3. Repeat 2. for $(n+2)$, $(n+3)$, $(n+4)$ etc, until V converges, i.e $d^h < \varepsilon$, where ε is the error tolerance level which is chosen to be a very small number.

1.3 The problem used in the Matlab code

Consider the following problem:

$$\max_{\{c_t, k_{t+1}\}_{t=0}^{\infty}} E_0 \sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\sigma}}{1-\sigma}$$

$$\text{st} \quad c_t + k_{t+1} = z_t k_t^{\theta} + (1-\delta) k_t$$

$$z_{t+1} = \rho z_t + \epsilon_{t+1}$$

where ϵ_{t+1} is a shock.

The Lagrangian and corresponding first-order conditions are

$$E_0 \left\{ \sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\sigma}}{1-\sigma} - \lambda_t \left[k_{t+1} + c_t - z_t k_t^{\theta} - (1-\delta) k_t \right] \right\}$$

$$\partial c_t: \quad \beta^t c_t^{-\sigma} = \lambda_t \tag{18}$$

$$\partial k_{t+1}: \quad \lambda_t = E_t \lambda_{t+1} \left[\theta z_{t+1} k_{t+1}^{\theta-1} + (1-\delta) \right] \tag{19}$$

$$\partial \lambda_t: \quad k_{t+1} + c_t = z_t k_t^{\theta} + (1-\delta) k_t \tag{20}$$

Combining the first-order conditions, the equilibrium can be characterized by two equations:

$$1 = \beta E_t \left(\frac{c_{t+1}}{c_t} \right)^{-\sigma} \left[\theta z_{t+1} k_{t+1}^{\theta-1} + (1-\delta) \right] \tag{21}$$

$$k_{t+1} + c_t = z_t k_t^{\theta} + (1-\delta) k_t$$

1.3.1 Steady state

At the steady state (21) implies

$$\frac{k}{y} = \frac{\beta\theta}{1-\beta(1-\delta)}$$

Note that the steady state of z is normalized to unity. Then, the steady state of output, capital and consumption can be written as follows:

$$y = \left(\frac{k}{y} \right)^{\frac{\theta}{1-\theta}} = \left(\frac{\beta\theta}{1-\beta(1-\delta)} \right)^{\frac{\theta}{1-\theta}}$$

$$k = \frac{k}{y} y = \left(\frac{\beta\theta}{1-\beta(1-\delta)} \right)^{\frac{1}{1-\theta}}$$

$$c = y - \delta k = \left(\frac{\beta\theta}{1-\beta(1-\delta)} \right)^{\frac{1}{1-\theta}} \left(\frac{1-\beta(1-\delta)-\beta\theta\delta}{\beta\theta} \right)$$

1.3.2 Value function

The problem can also be reformulated in a value function form:

$$V(z,k) = \max_{k'} \left\{ \frac{\left[zk^{\theta} + (1-\delta)k - k' \right]^{1-\sigma}}{1-\sigma} + \beta E V(z',k') \right\} \tag{22}$$

The matlab code solves (22). The simplest way to discretize the state space for k is to evenly space the n^k grid points along the real line between a lower and an upper bound. Another way to choose grid points is to place more grid points in places where the value function is more non-linear. Ideally, we would like to choose very large n^z and n^k to obtain as

good an approximation to the value function as possible. However, as n^z and n^k increase so do the time required to compute the approximation at each iteration. The cost of efficiency is sometimes large and prevents us from increasing much the dimensions of k and z. The latter effectively decreases the accuracy of the value function. The latter is referred to as the *curse of dimensionality.* The problem is more evident when there are more than two state variables in the model.

Description of the matlab code

1. Set the number of grid points for k (nk) and z (nz)

clear all

nk = 1000;

nz = 2; % **if you change this number you also need to change Z and PI**

2. Set values for the structural parameters

sigma = 2;

theta = 0.40;

delta = 0.10;

beta = 0.98;

3. Solve for the steady states for capital (k = k_ss), output (y = y_ss) and consumption (c = c_ss)

ky_ss = beta*theta/(1-beta*(1-delta));

y_ss = ky_ss^(theta/(1-theta));

k_ss = ky_ss * y_ss;

c_ss = y_ss - delta*k_ss;

z_ss = 1; % normalize the steady state for z to unity

4. Discretize the state space: Z is an n^z x 1 vector containing the states for

z_t and K is an n^k vector containing the states for k_t. PI is the Π matrix which contains the transition probabilities for z

Z = [0.975 1.025];

PI = [0.975 0.025; ...

0.025 0.975];

Kstep = (1.2*k_ss-0.8*k_ss) / (nk-1); % **increment for the k grid points**

K = 0.8*k_ss:Kstep:1.2*k_ss; % **the sequence of grid points starts from 0.8*k_ss and increases by Kstep until it reaches 1.2*k_ss**

5. Set the inital value of the value function: $V^{(0)}$

u = c_ss^(1-sigma) / (1-sigma);

V = ones(nz,nk) * u / (1-beta); % **This is the infinite sum of discounted utilities, when consumption remains always at the steady state.**

6. Iterate on value function until convergence

maxiter = 100;

maxdiff = 1e-4; % = 0.0001

diffV = 1;

iter = 1;

while (iter <= maxiter) & (diffV > maxdiff) % **as long as both statements hold the iteration continues**

% Recall that V(k,z)=max{u(k,z,k')+ beta*EV(k',z')}

% **Calculate expected V**

EV = zeros(nz,nk);

for iz = 1:nz

for ik_prime = 1:nk

for iz_prime = 1:nz

EV(iz,ik_prime) = EV(iz,ik_prime) + PI(iz,iz_prime) * V(iz_prime,ik_prime);

end

end

end

For example, if $\mathbf{n^k = n^z = 2}$, the above loops construct the following matrix

$$EV = \left[\begin{array}{c|c} \Pi_{11}V(z_1', k_1') + \Pi_{12}V(z_2', k_1') & \Pi_{11}V(z_1', k_2') + \Pi_{12}V(z_2', k_2') \\ \hline \Pi_{21}V(z_1', k_1') + \Pi_{22}V(z_2', k_1') & \Pi_{21}V(z_1', k_2') + \Pi_{22}V(z_2', k_2') \end{array}\right]$$

The first loop controls the rows of the matrix, the second loop controls the columns of the matrix and the third loop controls the sum in each cell.

7. Compute new V

newV = zeros(nz,nk);

iDecK = zeros(nz,nk); % index to decisions in k grid

for iz = 1:nz

for ik = 1:nk

% **Calculate c for each k'**

% **c is a 1*nk vector**

c = Z(iz)*K(ik)^theta + (1-delta)*K(ik) - K;

% **c must be non-negative. If there is a negative c replace it with a very small number**

% **Negative or small c's wouldn't be chosen anyway**

c = max(c,1e-8);

% **Compute the value function**

v = c.^(1-sigma)/(1-sigma) + beta*EV(iz,:); % **This is a 1 x $\mathbf{n^k}$ vector**

[newV(iz,ik), iDecK(iz,ik)] = max(v); % **pick the highest V, and save it in newV in the corresponding location. iDecK(iz,ik) is the location of optimal k′**

as a function of the location of the current state $(\mathbf{z}, \mathbf{k})$

end

end

8. Compute the distance between $\mathbf{V}^{(n+1)}$ and $\mathbf{V}^{(n)}$

diffV = max(abs((newV(:)-V(:))./V(:)));

iter = iter + 1;

V = newV;

end

fprintf(1,'Found value function after %5.0d iterations.',iter-1)

9. Plot optimal rule for k′

figure (1) % **number the figures. If you omit this command, only the last figure will be displayed**

plot(K,K(iDecK)')

title('Optimal decision rules for k-prime as a function of k. Different lines represent different z.')

10. Simulate the model using the optimal rules for T periods

T = 1000;

simiZ = zeros(1,T);

simiK = zeros(1,T);

simiZ(1) = 1;

simiK(1) = 1;

rand('state',150);

for i = 2:T

r = rand; % draw random number from uniform distribution on [0,1] interval

if r<PI(simiZ(i-1),1), iz = 1; else iz=2;

end

ik = simiK(i-1); % timing: simK(t) = choice of k' in period t, i.e. k in period t+1

newiK = iDecK(iz,ik);

simiZ(i) = iz;

simiK(i) = newiK;

end

% **There are two states for z, z_1 and z_2. Suppose that at i = 0, z_1 was relalized. At i = 1, draw a random number r$\in [0,1]$. If r is less than Π_{11} this means that the economy remains in state z_1. If r is greater than or equal to Π_{11}, then technology switches to z_2. Given, the realized state of z and the state of k we pick the optimal k$'$ whose location in K is given by iDecK(iz,ik). At i = 2, current k is the previous period k$'$. The same procedure is repeated T times.**

11. Computed the implied y and c using the simulated k

simZ = Z(simiZ(2:end)); % **a vector that contains the simulated value of** z **at each date**

simKnext = K(simiK(2:end)); % **a vector that contains the simulated grid point of** k' **at each date**

simK = K(simiK(1:end-1)); % **a vector that contains the simulated grid point of** k **at each date**

% **y and c**

simY = simZ.*simK.^theta;

simC = simY + (1-delta)*simK - simKnext;

12. Plot simulated k, y and c

figure (2)

subplot (221)

plot([simK'/k_ss])

```
title('Simulated k relative to its steady state')

subplot (222)
plot([simC'/c_ss])
title('Simulated c relative to its steady state')

subplot (223)
plot([simY'/y_ss])
title('Simulated y relative to its steady state')
```