Chapter 8

Stochastic Dynamic Programming

> "Life can only be understood going backwards but it must be lived going forwards." Søren Kierkegaard.

This chapter is about numerical stochastic dynamic programming and it is demonstrated for a number of different models how to compute the value function and policy function(s). Applications of numerical stochastic dynamic programming and value function iterations is treated very briefly in Christiano (1990a,b). The mathematical theory of stationary stochastic dynamic programming is treated in chapter 5 and I do not go into further details about the mathematics of stochastic dynamic programming in this chapter.

Bellman's equation is in itself almost an algorithm for computing the solution since the application of Bellman's (1957) method of successive iterations or **value function iterations** implies that starting from any arbitrary value function, iterating on Bellman's equation, leads to a sequence of value functions which converges to the true value function as the number of iterations goes to infinity. This convergence of the sequence of value functions can be proved by applying the n-stage Contraction Mapping Theorem, Stokey and Lucas with Prescott (1989).

When iterating on Bellman's equation, we compute for each iteration two functions, a value function and a policy function. An alternative and computationally faster algorithm is to iterate on the policy function and from the optimal policy function compute the value function. This is Howard's (1960) method of **policy function iterations**.

Stationary stochastic dynamic programming and value function iterations or policy function iterations can be used to numerically compute the solution to stochastic growth models which satisfy the conditions under which the competitive equilibrium is Pareto-optimal so that we can set up and solve a social planner's problem. The reason for this restriction is that the social planner directly controls

the aggregate per capita capital stock and consumption while the individual household only controls individual capital stock and consumption, that is, the individual policy functions are not in general identical to the aggregate policy functions when the competitive equilibrium is not Pareto-optimal. When solving models with a suboptimal equilibrium, we need to rely on other methods.

The solution procedure of value function iterations is as follows:

1. Set up Bellman's equation
2. Substitute (if possible) the constraints into the value function.
3. Define an operator which is a contraction mapping which is mapping a sequence of functions into itself.
4. Find the fixpoint of the operator by iterating on Bellman's equation.
5. Compute the optimal policy functions from the computed approximate optimal value function.

The method is applied for four models of increasing complexity. First, in section 8.1, a non-stochastic version of the Brock-Mirman model is solved using deterministic dynamic programming and value function iterations. The model is used to analyze the convergence properties of the model when starting from an initial level of capital below the steady state level of capital. In section 8.2, the Brock-Mirman model is exposed to a discrete state *i.i.d.* shock taking two different values, next a general neoclassical model of optimal economic growth is exposed to correlated productivity shocks following a discrete-time Markov chain in section 8.3, and finally an RBC model is analyzed in section 8.4 which is exposed to shocks from a seven-state Markov chain. In section 8.5, we introduce the method of policy function iterations and the problem of "the curse of dimensionality" is mentioned in section 8.6.

8.1 Deterministic Dynamic Programming

The non-stochastic version of the model of Brock and Mirman (1972) is a neoclassical model of optimal economic growth with constant labor supply, a logarithmic utility function, $u_t = \ln(c_t)$, a Cobb-Douglas production function, $y_t = Ak_t^{\alpha}$, where A is some constant, and complete depreciations in each period, $\delta = 1$, so that $i_t = k_{t+1}$. This specification of preferences and technology ensures that an analytical solution exists and we can directly compare the numerical solution with the analytical solution. Long and Plosser (1983) extend this model to a stochastic multi-sector economy which still has an analytical solution.

In the sequence problem, the social planner chooses an infinite sequence of consumption and next period's capital stock $\{c_t, k_{t+1}\}_{t=0}^{\infty}$ in order to maximize

$$\max_{(c_t,k_{t+1})_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t \cdot \ln c_t, \qquad 0 < \beta < 1, \tag{8.1}$$

subject to

$$\begin{aligned} c_t + k_{t+1} &\leq Ak_t^{\alpha} && (8.2)\\ k_0 &> 0, \quad \text{given} && (8.3)\\ c_t, k_t &\geq 0, \quad \forall t, && (8.4) \end{aligned}$$

The state variable of the model is k_t, and the control variables are c_t and k_{t+1}. Since we solve a social planner's problem, we do not distinguish between individual and aggregate state variables, since the social planner is assumed to operate directly on the aggregate variables.

After substituting the constraint (8.2) into the utility function, we can reformulate the optimization problem as a dynamic programming problem where the social planner has to choose a policy function for next period's capital, $k_{t+1} = g(k_t)$ in order to solve Bellman's equation, where a prime denotes next period's value

$$v(k) = \max_{k' \in X} \left\{ \ln \left(Ak^{\alpha} - k' \right) + \beta v \left(k' \right) \right\}.$$

The problem is to translate this mathematical expression into MATLAB code. For the stationary (no-growth) Brock-Mirman model, capital can only take values within some bounded range $k \in X = \left(0, \bar{k}\right)$, where $\bar{k}$ is the upper limit on capital. $\bar{k}$ is found from (8.2) by removing time subscripts and setting $c = 0$ and then $\bar{k} = \left(\frac{1}{A}\right)^{\frac{1}{\alpha - 1}}$. First determine the discrete set of values that capital can take. In the MATLAB program, the capital stock thus becomes a vector of values between zero and an upper limit, or we may want to limit the grid on capital even further. The value function is a function of the capital stock and is also a vector and so is the policy function for next period's capital stock, since $k' = g(k)$. When iterating on Bellman's equation, we start with some arbitrary initial value of the last period value function, $v_0(k')$, which may be a zero-vector, and compute the policy function and the value function $v_1(k)$

$$v_1(k) = \max_{k' \in X} \left\{ \ln \left(Ak^{\alpha} - k' \right) + \beta v_0 \left(k' \right) \right\}.$$

We therefore need to compute the numerical value of the utility function which is a matrix in order to solve the maximization problem. In the computational procedure, use the following steps:

1. Define the parameters, compute the steady state value of the capital stock, k^*, discretize the state space by constructing a grid on the capital stock with g values $k \in X = [k_1 < k_2 < ... < k_g]$ with $k_1 > 0$ in the neighborhood of the steady state, say from $0.95k^*$ to $1.01k^*$.[1]

2. Compute a $(g \times g)$-dimensional consumption matrix, $C(i,j)$, with the value of consumption for all the $(g \times g)$ values of k and k'. Then next compute a $(g \times g)$-dimensional matrix with the utility of consumption for all the $(g \times g)$ values of k and k'.

3. Define an initial $(g \times 1)$-dimensional vector of zeros for the initial value function, v_0. Compute the $(g \times 1)$-dimensional vector of one-period value functions: $v_1(k) = \max_{(k' \in X)} \{\ln(Ak^\alpha - k') + \beta v_0(k')\}$.

4. Continue by iterating on Bellman's equation until convergence whereby we have computed a close approximation to the true value function, v.

5. Compute the approximated true policy functions, $k' = g(k)$ and $c = Ak^\alpha - k'$, based on the approximated true value function by solving Bellman's equation.

6. Economic analysis of convergence towards steady state: Compute the time path of capital and consumption starting from an initial level of capital at 5 percent below its steady state level.

Step 1. Define parameters, compute the steady state, and construct the grid on the capital stock.

For the computations below, the following parameter values have been used (corresponding to quarterly data): $A = 5$, $\beta = .9888$, and $\alpha = 0.4$.

The steady state value of capital is computed as follows. Derive the *first-order condition* by differentiating Bellman's equation with respect to k' and set it equal to zero

$$\frac{1}{Ak^\alpha - k'} = \beta \frac{\partial v(k')}{\partial k'}, \tag{8.5}$$

and the *envelope condition* is derived by differentiating Bellman's equation with respect to k

[1] We want to get as fine a grid on the capital stock as possible in order to get as close approximation as possible. However, the "curse of dimensionality" limits the number of grid points, which is essentially limited by the amount of memory in the computer. Start with, say, 101 grid points. Use, say, 1001 grid points for the final computations or more if you have access to a fast computer with a large amount of RAM. It is often advisable to choose an uneven number of grid points which are symmetric around the steady state capital stock to ensure that the steady state level of capital is one of the grid points. In this section, we use the model to analyse the transitional dynamics of the model (compute initial response functions) and therefore choose a grid from -5 percent of the steady state capital stock to +1 percent.

$$\frac{\partial v(k)}{\partial k} = \frac{\alpha A k^{\alpha-1}}{Ak^{\alpha} - k'}, \tag{8.6}$$

lead the envelope condition (8.6) one period, substitute into the first-order condition, and set $k = k' = k''$ in steady state. The steady state capital stock is

$$k^* = \left(\frac{1}{\alpha\beta A}\right)^{\frac{1}{\alpha-1}}. \tag{8.7}$$

The steady state level of capital per capita is $k^* = 3.11$ for the parameters values given above.

Define a grid on capital around the steady state value of capital with g points. Start by setting $g = 101$ in order to keep computing time low. We use the following MATLAB commands, where $k^* =$ `kstar`.

```
kstar = (1/(alpha*beta*A))^(1/(alpha-1));
g = 101;
k = (0.95*kstar:(0.06*kstar/(g-1)):1.01*kstar)';
```

The last MATLAB command creates a grid on capital between .95 and 1.01 times the steady state value of capital with a distance between the grid points determined by g and the distance between the lowest and the highest value of capital.

Step 2. Construct consumption and welfare matrices

Create first the matrix of consumption from

$$c = Ak^{\alpha} - k',$$

for each value of k and k'. Notice that k and k' can take the same values, since $k' \in X$ and $k \in X$. The consumption matrix can be created in either of two ways, either by using a double loop over the indexes $i = 1 : g$ and $j = 1 : g$, or by the use of vectorization methods, which is much faster. Notice that y denotes k' in the MATLAB program below. The consumption matrix with elements $U(i, j)$ is constructed so that the i-index denotes the k' variable and the j-index denotes the k variable. The value of consumption is increasing to the right with an increasing value of k and decreasing down in the consumption matrix with an increasing value of investments, k'.

$$\overbrace{\begin{matrix} k' \\ \downarrow \end{matrix}\left\{\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1g} \\ c_{21} & c_{22} & & \\ \vdots & & \ddots & \vdots \\ c_{g1} & \cdots & & c_{gg} \end{bmatrix}\right.}^{k\rightarrow}$$

Impose the constraint that consumption is non-negative. Next, compute the utility of consumption as the natural logarithm of the consumption matrix.

MATLAB program: using for loop

```
c = zeros(g,g);
y = k;
for i = 1:g;
 for j = 1:g;
  if (A*((k(j))^alpha) - y(i)) > 0;
   c(i,j) = (A*((k(j))^alpha) - y(i));
  end;
 end;
end;
U = log(c);
```

Vectorization

```
I = ones(g,1);
c = I * (A*k.^alpha)' - y*I';
for i = 1:g;
 for j = 1:g;
  if c(i,j) <= 0;
   c(i,j) = 0;
  end;
 end;
end;
U = log(c);
```

Step 3. Define the initial value function and compute the first-period value function

Set the initial value function equal to the zero column vector: `v = zeros(g,1).`

Define the operator T as below and compute the first-period value function from Bellman's equation

$$\begin{aligned} Tv &= v_1(k) = \left(\max_{k' \in X} \left\{ \ln \left(Ak^{\alpha} - k' \right) + \beta v_0(k') \right\} \right)' \\ &= \left(\max_{k' \in X} \left\{ U + \beta v_0(k') \right\} \right)'. \end{aligned} \tag{8.8}$$

T is an operator that maps bounded continuous functions into bounded continuous functions. T is a contraction mapping, chapter 5.

The value function $Tv = v_1(k)$ is a $(g \times 1)$-dimensional column vector. The right hand side in the outer parenthesis is a $(g \times g)$-dimensional matrix which is the sum of the $(g \times g)$-dimensional matrix U and a $(g \times g)$-dimensional zero matrix constructed by multiplying the $(g \times 1)$-dimensional column vector $v_0(k')$ of zeros with a $(1 \times g)$ row vector of ones multiplied by the scalar β (element by element).

The maximization is for given values of k (that is, we find the maximum value in each column). The object is to choose the value of k' (the row number) which maximizes the value of the right hand side of (8.8) for each value of k (columns)

$$\begin{bmatrix} v_1(k_1) \\ v_1(k_2) \\ \vdots \\ v_1(k_g) \end{bmatrix} = \left(\max_{k' \in X} \left\{ \begin{bmatrix} U_{11} & U_{12} & \cdots & U_{1g} \\ U_{21} & U_{22} & \cdots & U_{2g} \\ \vdots & \vdots & \ddots & \vdots \\ U_{g1} & U_{g2} & \cdots & U_{gg} \end{bmatrix} + \beta \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \right\} \right)',$$

where the prime denotes the transpose. We hereby implicitly compute policy functions for capital, since $k' = k(j)$ so for each iteration, we compute two functions, a value function and a policy function.

Use the MATLAB command `max` to do the maximization. The max command returns an $(1 \times g)$ vector, which is transposed to give the $(g \times 1)$ vector, $v(k)$. The first row of the new vector of the value function, Tv, contains the maximum value in the first column on the right hand side, the second row the maximum value in the second column etc.

Computing the first value function is simple. This is the first row of the right hand side matrix where k' is lowest. Use the MATLAB commands, where `Tv` is the first-period value function.

MATLAB program

```
v = zeros(g,1);
o = ones(1,g);
Tv = (max(U + beta * v * o))';
v = Tv;
```

Step 4. Iterating on Bellman's equation

Set $v = Tv$ and continue iterations

$$Tv = \left(\max_{k' \in X} \{U + \beta v(k')\} \right)'.$$

The value function is converging to the true value function, figure 8.1. The number of iterations can be determined using a distance measure and the matrix norm so that iterations stop when the distance between the last two iterations is less than some prespecified limit, see section 8.1.1.

In the following MATLAB program, `limit` is the convergence criterium, `distance` is a distance measure using the matrix norm to determine the distance between the last two iterations, `n` is a counter, and `v` is the value function.

MATLAB program

```
n = 1;
limit = 1 * 10^(-8);
distance = 1;
```

```
while distance > limit;
 v = Tv;
 Tv = (max(U + beta*v*o))';
 distance = norm(abs(Tv-v));
 n=n+1;
end;
v = Tv;
```

The true value function is computed as

$$v(k) = (1-\beta)^{-1}\left[\ln\left(A\left(1-\alpha\beta\right)\right) + \frac{\alpha\beta}{1-\alpha\beta}\ln A\alpha\beta\right] + \frac{\alpha}{1-\alpha\beta}\ln k,$$

as seen from Sargent (1987b). The true value function is the upper line in figure 8.1, and it is seen how iterating on Bellman's equation leads to convergence of the value function. Figure 8.1 is produced using 250 grid points. The lower dashed line is the value function computed for 600 iterations, the dotted line is the value function computed for 700 iterations, and the third line is computed for 800 iterations. It is seen that more iterations lead to further convergence.

Step 5. Computation of the optimal policy functions

We want to compute the optimal policy function or transition function for capital

$$k' = g(k).$$

The true policy function for capital is given by

$$k' = \alpha\beta Ak^{\alpha},$$

so that the true policy function for consumption is

$$c = Ak^{\alpha} - g(k) = Ak^{\alpha} - \alpha\beta Ak^{\alpha} = \left(1-\alpha\beta\right)Ak^{\alpha}.$$

The approximated optimal policy function for capital is computed as the k' which solves

$$Tv = \left(\max_{k'\in X}\left\{U + \beta v^{*}(k')\right\}\right)',$$

for each k' where $v^{*}(k')$ is the approximate optimal value function.

Use the following steps:

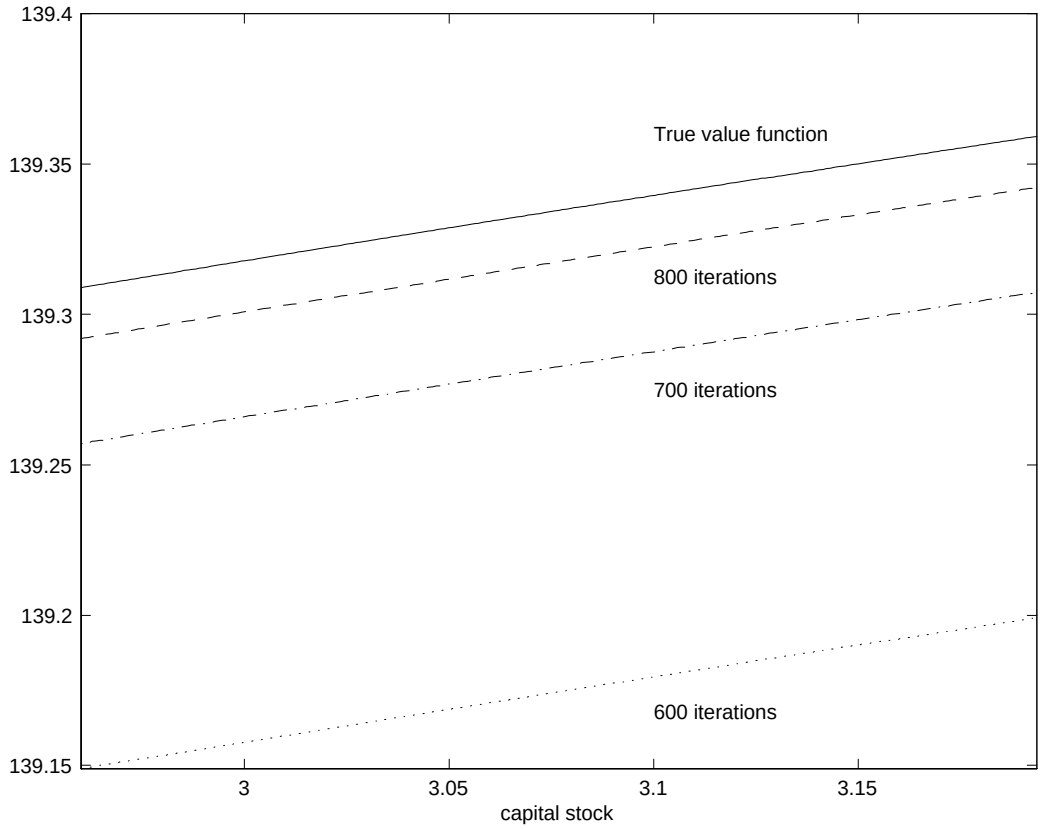

Figure 8.1: Convergence towards the true value function.

1. Maximize Bellman's equation and find the row number, j, for the maximum value

$$Tv = \left(\max_{k' \in X} \left\{ U + \beta v(k') \right\} \right)',$$

2. Compute the approximate policy functions of capital and consumption

$$\begin{aligned} k' &= g(k) = k(j) \\ c &= Ak^{\alpha} - k', \end{aligned}$$

where the ith value of k' is the jth value of k for row number i.

In the following MATLAB program, `policyk` is the approximate policy function for capital, `policyc` is the approximate policy function for consumption, `Tv`

is the maximum value for each column in the right-hand side matrix, and `j` is the row-index number where the maximum value is found.

MATLAB program

```
[Tv,j] = max(U + beta * v*o);
policyk = k(j);
policyc = (A*k.^alpha)' - policyk;
```

The true and approximated policy functions are illustrated in figure 8.2. The finer the grid on capital, the smaller are the deviations of the approximated from the true policy function. The upper panel of figure 8.2 depicts the policy function for capital in a (k, k') diagram together with a $k' = k$ line (a 45^o-line if the axes are the same).

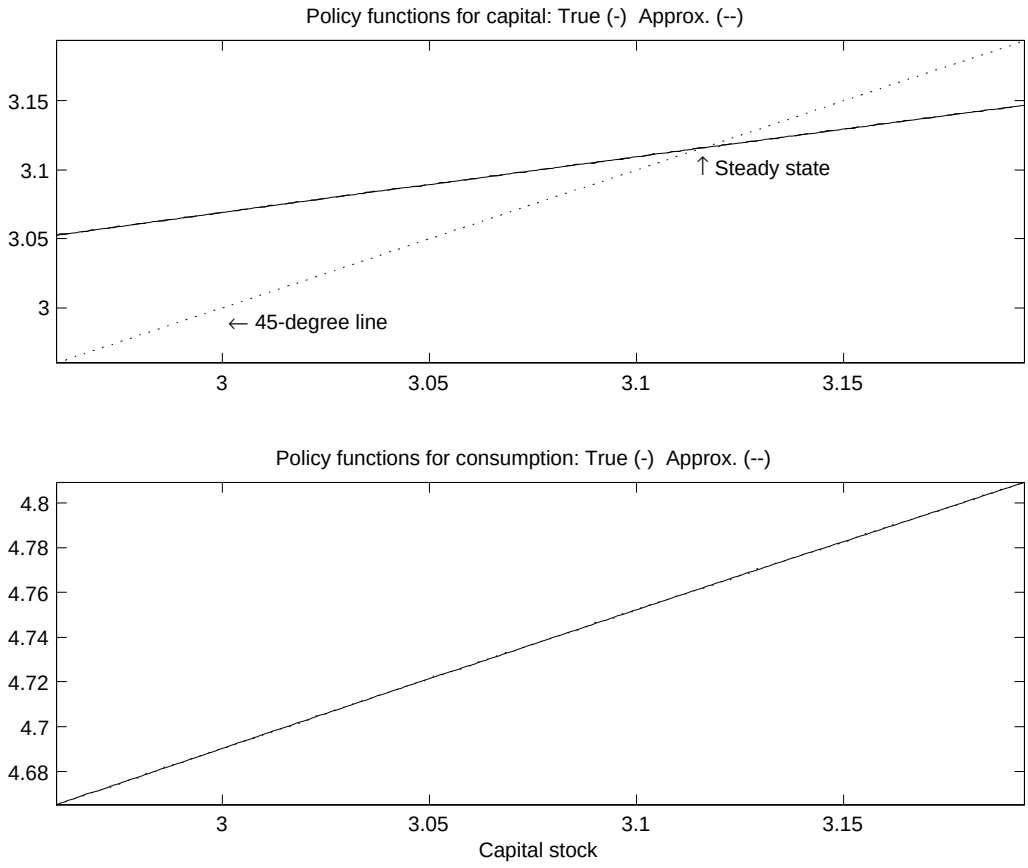

Figure 8.2: True and approximated policy functions for capital and consumption based on 500 grid points for capital and 2000 iterations.

A visual inspection of the plot of the policy functions shows that the slope of the policy function is positive and less than unity, the *policy functions are increasing in* k, *and concave*. It is seen that the policy function for capital, $k' = g(k)$, is

a strictly concave and a monotone, non-decreasing function of k. It is a first-order non-linear difference equation and it crosses the 45^o-line from above and is therefore globally asymptotically stable. The Brock-Mirman model has a unique non-zero steady state and the state variable converges towards its steady state, regardless of the initial conditions. Taking logarithms of the analytical solution to the policy function for capital, we have

$$\ln k_{t+1} = \ln(\alpha\beta A) + \alpha \ln k_t,$$

and the linear first-order difference equation is globally asymptotically stable when $|\alpha| < 1$ and in the limit when time goes to infinity

$$\lim_{t\to\infty} \ln k_t = \frac{\ln(\alpha\beta A)}{1-\alpha}.$$

Step 6. Convergence analysis: compute the time path of capital, consumption, and output. Initial response functions.

We can illustrate the convergence properties of the Brock-Mirman model by computing the initial response functions and thereby illustrate how output, capital, and consumption over time converge towards their steady state, figure 8.3. The procedure is as follows:

1. Start from the initial level of capital minus five percent of its steady state level.
2. Compute the first period value of capital from the computed approximate optimal policy function computed above.
3. Make a loop over the time index for T periods.
   (a) Use the find command to find the index of the grid on capital where the value of next period's capital is.
   (b) Use this index to determine the time $t+1$ period value of capital from the policy function.
   (c) Use this value and the initial capital stock to compute the time t period value of consumption and output.
   (d) Continue until the end of the loop.

MATLAB program

```
ktime(1,1) = policyk(1);
for t = 1:T;
 j = find(k == ktime(t,1));
 ktime(t+1,1) = policyk(j);
 ctime(t,1) = policyc(j);
```

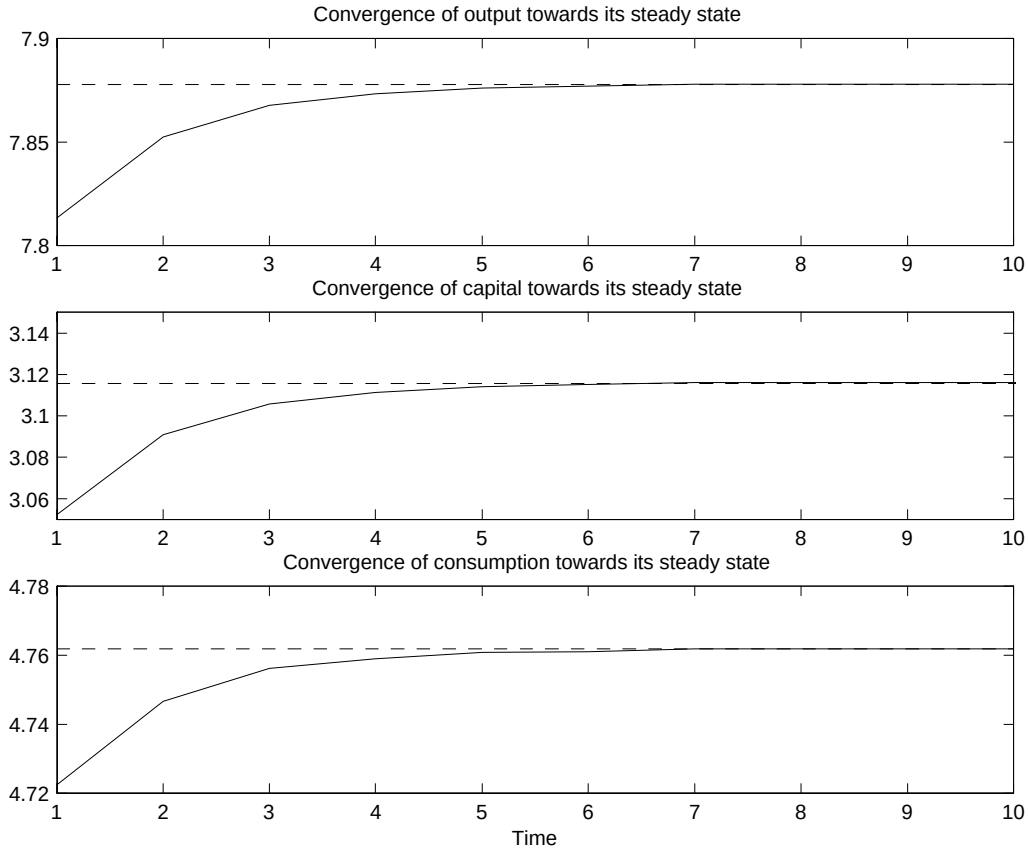

Figure 8.3: The convergence of output, capital, and consumption towards their steady states.

```
 ytime(t,1) = A*(ktime(t,1)^alpha);
end;
```

The computational method of this section is implemented in the program `Dynprogch81.m`.

8.1.1 Convergence Criterium and the Matrix Norm

The number of iterations on the value function is determined by a convergence criterium so that iterations continue as long as some measure of the distance (a norm) between the value functions v_j and v_{j+1} is greater than some pre-defined distance. The limit used is $1 \cdot 10^{-8}$, and the distance between two value functions is measured as the matrix norm `distance = norm(abs(Tv-v));`

For vectors, the MATLAB command `norm(V)` is the 2-norm or Euclidean norm of the vector V which is written as $\|V\|$

$$\|V\| = \sqrt{V_1^2 + V_2^2 + ... + V_n^2}$$

For matrices, the MATLAB command `norm` is the largest singular value of a matrix, that is, `norm(X) = max(svd(X))`, the maximum of the singular value decomposition of the matrix under consideration. A matrix can be given a singular value decomposition by using the MATLAB command `[u,s,v] = svd(X)`, where `X = u*s*v'` and `s` is a diagonal matrix of the same dimension as `X` and with the nonnegative singular values on the diagonal in decreasing order

$$s = \begin{bmatrix} s_1 & 0 & \cdots & 0 \\ 0 & s_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & s_n \\ \vdots & & & 0 \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

The singular values are the non-negative square roots of the eigenvalues of $X'X$, which are real and non-negative, Lindfield and Penny (1995).

8.2 Stochastic Dynamic Programming

We now study the stochastic Brock-Mirman model when it is exposed to an *i.i.d.* stochastic shock in each period. We have now two state variables, k_t and z_t with $k_t \in X$ and $z_t \in Z$ which are both compact (closed and bounded) and convex subsets of the Euclidean space. The product space is given by $S = X \times Z$. The state variables of the model are $s_t = \{k_t, z_t\}_{t=0}^{\infty}$ where $\{z_t\}_{t=0}^{\infty}$ is a sequence of discrete state *i.i.d.* uniformly distributed stochastic technology shocks with an m-dimensional state space, here $m = 2$

$$z_t \in Z = [0.9835, 1.0165]$$

each with the associated probabilities

$$\begin{aligned} \pi_1 &= \Pr\{z_t = z_1\} = 0.5 \\ \pi_2 &= \Pr\{z_t = z_2\} = 0.5 \end{aligned}$$

so that $\pi = \begin{bmatrix} \pi_1 & \pi_2 \end{bmatrix}$ is the probability distribution over the state next period.

The control variables are $\{c_t, k_{t+1}\}_{t=0}^{\infty}$. The stochastic formulation implies that the value function(s) and policy functions depend on the two state variables: k_t

and z_t. There are m value functions and policy functions, one for each value of the exogenous state variable z_t.

Under uncertainty, the social planner chooses contingent claim policy functions for next period's capital stock. Start by formulating Bellman's equation where a prime denotes next period's value

$$v(k,z) = \max_{k' \in X} \left\{ \ln\left(zAk^{\alpha} - k'\right) + \beta E v\left(k', z'\right) \right\}.$$

When solving the stochastic model, substitute for the expected value

$$\begin{aligned} v(k,z) &= \max_{k' \in X} \left\{ \ln\left(zAk^{\alpha} - k'\right) + \beta \sum_{i=1}^{2} \left(v_i\left(k', z_i'\right) \cdot \pi_i \right) \right\} \\ &\max_{k' \in X} \left\{ \ln\left(zAk^{\alpha} - k'\right) + \beta \left[v_1\left(k', z_1'\right) \cdot \pi_1 + v_2\left(k', z_2'\right) \cdot \pi_2 \right] \right\}. \quad (8.9) \end{aligned}$$

As in the deterministic model, we are interested in studying the long-run behavior of the state variables $s_t = \{k_t, z_t\}_{t=0}^{\infty}$ when time goes to infinity which are governed by some *transition function* P which again is determined by the transition function for capital, the policy function for consumption, and the distribution of the stochastic shock.

It can be shown that if the policy function for consumption is a measurable function of the endogenous state variable and the partial history of shocks, then the pair of state variables $\{k_t, z_t\}$ follows a Markov process on the product space $(S, \mathcal{S}) = (X \times Z, \mathcal{X} \times \mathcal{Z})$ with a transition function P, Stokey and Lucas with Prescott (1989). The state variables as well as the policy-variables and other variables in the model are stochastic processes. The stochastic analogue of a steady state is a limiting probability distribution or limiting probability measure. In chapter 3, section 3.3.2, we stated some conditions under which a limiting distribution of the state variables exists with a stationary or invariant probability distribution.

For the stochastic Brock-Mirman model, it is easy to show the distribution of the state variables as time goes to infinity. The analytical solution to the stochastic Brock-Mirman model involves the policy-function for capital

$$k_{t+1} = z_t \alpha \beta A k_t^{\alpha},$$

which can be loglinearized as

$$\ln k_{t+1} = \ln\left(\alpha \beta A\right) + \alpha \ln k_t + \ln z_t.$$

This is a first-order stochastic difference equation. In the non-stochastic case we have that the difference equation is globally asymptotically stable when $|\alpha| < 1$

and capital, output, and consumption converge to their steady state values. In the presence of uncertainty, there is no steady state in the non-stochastic sense. By contrast, depending on the specification of the shock process, capital, output, and consumption converge towards their stationary distributions. Let the stochastic shock have mean $E(\ln z_t) = \mu$ and variance $var(\ln z_t) = \sigma^2$, we then have from recursively substituting backwards that

$$E(\ln k_t) = \alpha^t \ln k_0 + \left(\frac{1-\alpha^t}{1-\alpha}\right) \ln[\alpha\beta A] + \left(\frac{1-\alpha^t}{1-\alpha}\right) \mu,$$

and

$$\begin{aligned} \lim_{t\to\infty} [E(\ln k_t)] &= \lim_{t\to\infty} \left[\alpha^t \ln k_0 + \left(\frac{1-\alpha^t}{1-\alpha}\right) \ln[\alpha\beta A] + \left(\frac{1-\alpha^t}{1-\alpha}\right) \mu\right] \\ &= \left(\frac{1}{1-\alpha}\right) (\ln[\alpha\beta A] + \mu) \end{aligned} \tag{8.10}$$

that is, the mean of the distribution of capital converges to the right hand side expression, $\left(\frac{1}{1-\alpha}\right)(\ln[\alpha\beta A] + \mu)$. The variance is equally given by

$$var(\ln k_t) = \left(\frac{1-\alpha^{2t}}{1-\alpha^2}\right) \sigma^2,$$

and

$$\lim_{t\to\infty} [var(\ln k_t)] = \lim_{t\to\infty} \left[\left(\frac{1-\alpha^{2t}}{1-\alpha^2}\right) \sigma^2\right] = \left(\frac{1}{1-\alpha^2}\right) \sigma^2. \tag{8.11}$$

The conclusion is that the logarithm of capital is a stochastic process which is distributed with mean $\left(\frac{1}{1-\alpha}\right)(\ln[\alpha\beta A] + \mu)$ and variance $\left(\frac{1}{1-\alpha^2}\right)\sigma^2$. Output is a function of the capital stock and is therefore also a stochastic process: $y_t = z_t A k_t^\alpha$, and $\ln y_t = \ln A + \alpha \ln k_t + \ln z_t$.

The calibrated parameter values are given in table 8.1.

Table 8.1: Parameter values of stochastic Brock-Mirman model

A	β	α	θ_1	θ_2	π_1	π_2
1.0	0.9888	0.4	1.0165	0.9835	0.5	0.5

Since this is a stochastic model, with the given specification of the transition function, policy function, and shock process, the stochastic processes of capital and consumption converge to stationary probability distributions and we are interested in the behavior of the model generated time series around their non-stochastic steady states. We limit the grid on capital to ± 3 percent of the steady state value of capital.

In the computational procedure use the following steps:

1. Define the parameters, compute the steady state value of the capital stock, k^*, and discretize the state space by constructing a grid on the capital stock with g values around its steady state,[2] and define the two values of the stochastic shock. Define the distribution of the stochastic shock.

2. Compute the two $(g \times g)$-dimensional matrices, C_1 and C_2, conditional on the value of the stochastic shock, where $C_i = z_i A k^{\alpha} - k'$, where $i = 1, 2$. Next, compute the two matrices of utility U_i, $i = 1, 2$ conditional on the stochastic shock.

3. Define two initial $(g \times 1)$-dimensional zero-vectors as the initial value functions, v_{10} and v_{20}. Compute the first-period value function.

4. Continue by iterating on Bellman's equation until convergence.

5. Compute the policy functions based on the final value functions.

6. Stochastic simulations: Compute the time path of capital and consumption around the steady state level of capital by stochastic simulations n times. Compute summary statistics for each stochastic simulation and take the mean of the statistics over the n simulations.

Most of the steps in the stochastic case are repetitions of the steps in the nonstochastic case, so we skip them here and go directly to iterations on the value function.

Step 3. Computing the first-period value function

Set the two initial value functions equal to the zero column vector: `v10 = zeros(g,1)` and `v20 = zeros(g,1)`.

Define two operators T_1 and T_2 and compute the two *first-period* value functions from Bellman's equation starting from initial value functions equal to the zero-vectors

$$\begin{aligned} T_1 v &= v_{11}(k,z) = \left(\max_{k' \in X} \{U_1 + \beta \left(zeros(g,1) \cdot ones(1,g) \right)\} \right)' \\ T_2 v &= v_{21}(k,z) = \left(\max_{k' \in X} \{U_2 + \beta \left(zeros(g,1) \cdot ones(1,g) \right)\} \right)'. \end{aligned}$$

[2] The grid on capital in this section is ± 4 percent around the steady state value of capital.

As before, the value function $T_1 v = v_{11}(k,z)$ is a $g \times 1$ column vector. The right hand side in the curly parentheses $\{\cdot\}$ is one big $(g \times g)$-dimensional matrix.

MATLAB Program

```
v = zeros(g,1);
o = ones(1,g);
Tv1 = (max(U1+beta*v*o))';
Tv2 = (max(U2+beta*v*o))';
```

Step 4. Iterating on the value function

Set $v_1 = T_1 v$ and $v_2 = T_2 v$ and based on (8.9), continue iterations

$$\begin{aligned} T_1 v &= \left(\max_{k' \in X} \left\{ U_1 + \beta \left(\left(\left[v_1(k', z_1') \right] \pi_1 + \left[v_2(k', z_2') \right] \pi_2 \right) \cdot ones\,(1,g) \right) \right\} \right)' \\ T_2 v &= \left(\max_{k' \in X} \left\{ U_2 + \beta \left(\left(\left[v_1(k', z_1') \right] \pi_1 + \left[v_2(k', z_2') \right] \pi_2 \right) \cdot ones\,(1,g) \right) \right\} \right)', \end{aligned}$$

where the expectations operator is substituted for the expected value of next period's value functions: $\beta E v\,(k', z') = \beta \left(\left[v_{10}(k', z_1') \right] \pi_1 + \left[v_{20}(k', z_2') \right] \pi_2 \right)$ as in (8.9) where π_i is the probability of next period shock being θ_i, $i = 1, 2$.

1. The value function as a function of k and z converges to the true value function. The two value functions are plotted in figure 8.4.

MATLAB Program

In the following: `v1` and `v2` are the final value functions, and `n` is a counter.

```
distance1 = 1; distance2 = 1; limit = 1 *10^(-8);
while distance1 > limit | distance2 > limit;
 B = beta*((0.5*v1 + 0.5*v2)*o);
 Tv1 = (max(U1 + B))';
 Tv2 = (max(U2 + B))';
 distance1 = norm(abs(Tv1-v1));
 distance2 = norm(abs(Tv2-v2));
 v1 = Tv1;
 v2 = Tv2;
end;
```

Step 5. Computing the policy functions

The policy functions are computed as in the previous section:

1. Maximize

$$\begin{aligned} T_1 v &= \max_{k' \in X} \left\{ U_1 + \beta \left(\left[v_1(k', z_1') \right] \pi_1 + \left[v_2(k', z_2') \right] \pi_2 \right) \right\} \\ T_2 v &= \max_{k' \in X} \left\{ U_2 + \beta \left(\left[v_1(k', z_1') \right] \pi_1 + \left[v_2(k', z_2') \right] \pi_2 \right) \right\}, \end{aligned}$$

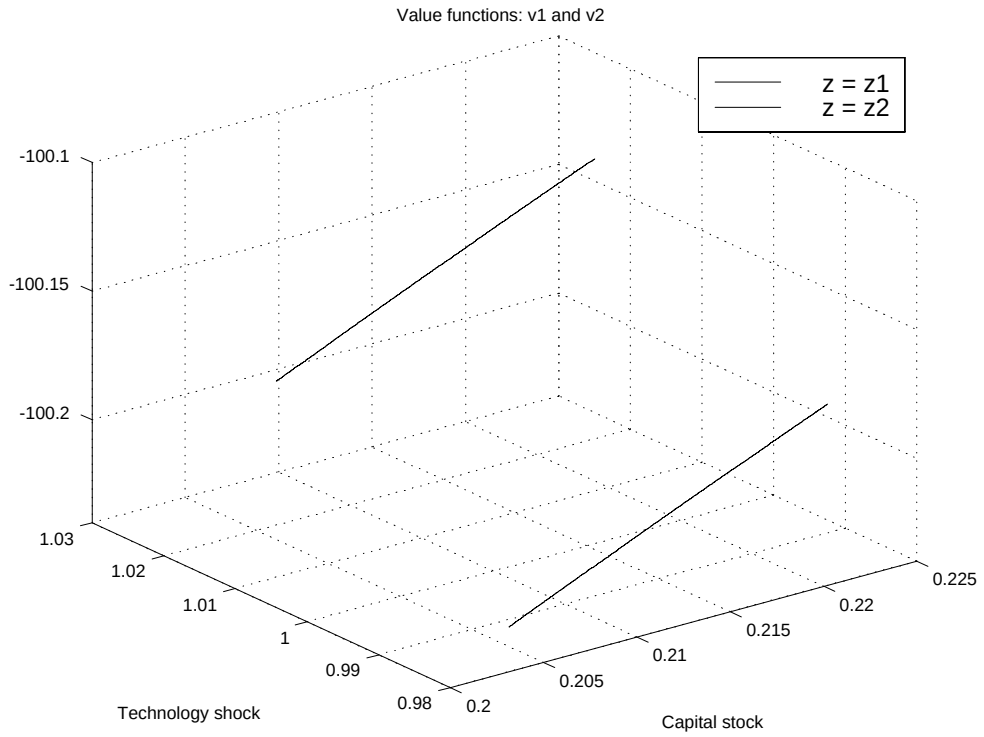

Figure 8.4: Value functions v_1 and v_2, $g = 401$ and 2000 iterations.

where v_1 and v_2 are the final, approximated value functions computed above, and find the index row number, j, where the maximized value is for each k for each of the two value functions.

2. Compute the policy functions for capital from the index, j, $k' = k(j)$. Compute The policy function for consumption residually.

The two policy functions for capital and the two policy functions for consumption are graphed in figure 8.5.

MATLAB Program

```
[Tv1,jj] = max(U1 + B);
[Tv2,jk] = max(U2 + B);
policyk1 = (k(jj))';
policyk2 = (k(jk))';
policyc1 = (theta1*A*(k.^alpha) - policyk1')';
policyc2 = (theta2*A*(k.^alpha) - policyk2')';
```

Step 6. Computing the time path of capital and consumption

We want to make a stochastic simulation of the model. We use the following steps:

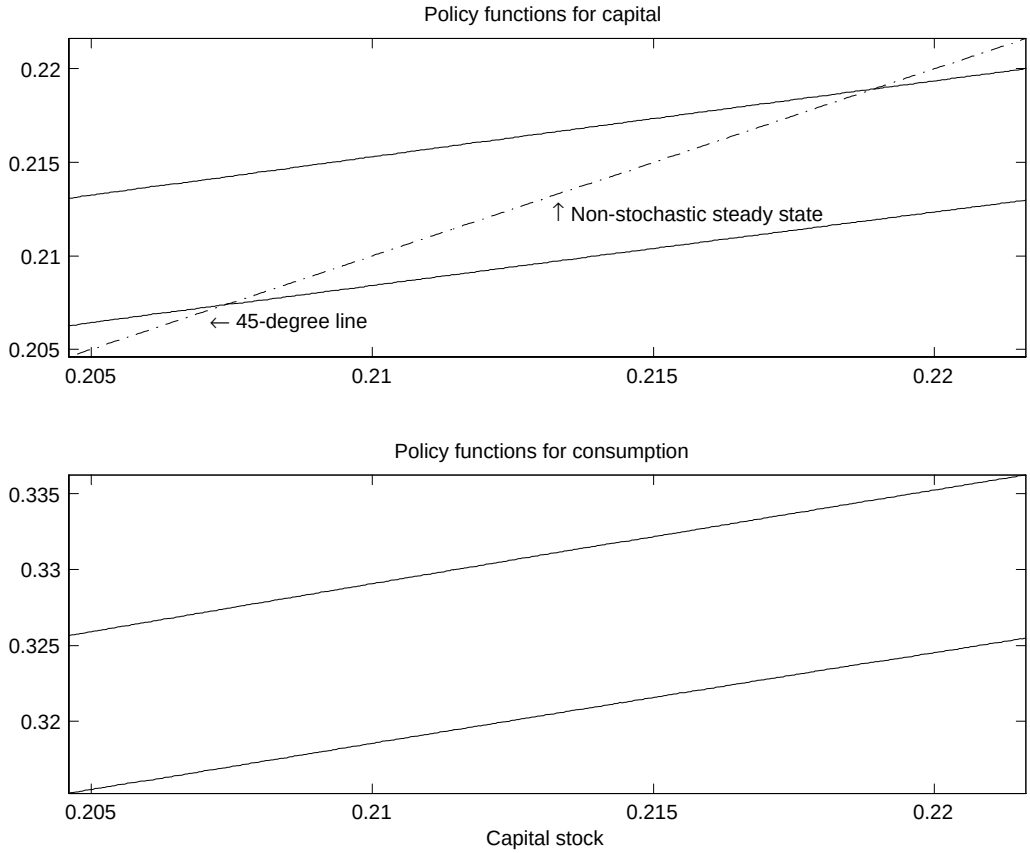

Figure 8.5: Policy functions for capital and consumption.

1. Make a loop over the number of simulations, n.

2. Make a loop over the number of periods $t = 1 : T$. Start with an initial (first-period) value of capital equal to the steady state value of capital, $k(1) = k^*$.

3. Use a random number generator to determine the value of the shock in period t.

   (a) If $\theta = \theta_1$, then next period's capital stock is $k_{t+1} = g_1(k_t, z_t)$ and if $\theta = \theta_2$, then next period's capital stock is $k_{t+1} = g_2(k_t z_t)$. Next period's capital and consumption is determined from the state contingent policy functions found above.

   (b) Use the find command to determine the next period's capital stock.

4. End the if loop, end the time t loop, and end the loop over number of simulations.

MATLAB Program

The following MATLAB code illustrates the method for one simulation. The variables `sk` and `sc` are the time path of capital and consumption respectively.

```
theta = ones(T,1);
sk = zeros(T+1,1);
sc = zeros(T,1);
sk(1,1) = kstar;
normale = randn(T,1);
m = g-((g-1)/2);
for t=1:T;
 if normale(t,1) >= 0;
  sk(t+1,1) = policyk1(m);
  sc(t,1) = policyc1(m);
  m = find(k == sk(t+1,1));
  theta(t,1) = theta1;
 else;
  sk(t+1,1) = policyk2(m);
  sc(t,1) = policyc2(m);
  m = find(k == sk(t+1,1));
  theta(t,1) = theta2;
 end;
end;
```

The computational method of this section is implemented in the program `Dynprogch82.m`.

8.3 Stochastic Neoclassical Growth Model with Markovian Shocks

We study a general neoclassical model of optimal economic growth, a Ramsey model, with a C.I.E.S. utility function, a positive depreciation rate between zero and one, and the model is exposed to a discrete stochastic shock in each period which is generated by a seven state discrete time Markov chain. The state variables of the model are $\{k_t, z_t\}_{t=0}^{\infty}$ where $\{z_t\}_{t=0}^{\infty}$ is a sequence of stochastic technology shocks with a distribution specified below. The control variables are $\{c_t, k_{t+1}\}_{t=0}^{\infty}$.

Start by formulating Bellman's equation

$$v(k_t, z_t) = \max_{c_t, k_{t+1}} \left\{ u(c_t) + \beta E v(k_{t+1}, z_{t+1}) \right\}, \tag{8.12}$$

subject to

$$\begin{aligned} c_t + i_t &\leq z_t f(k_t) && (8.13)\\ k_{t+1} &= i_t + (1-\delta)\, k_t, \qquad 0 < \delta < 1 && (8.14)\\ k_0 &> 0, \qquad \text{given} \\ c_t, k_t &\geq 0, \qquad \forall t, && (8.15) \end{aligned}$$

and substitute the restrictions (8.13)-(8.14) into the value function and reformulate by using the prime notation

$$v(k,z) = \max_{k' \in X} \left\{ u\left(z_t f(k) - k' + (1-\delta)\, k\right) + \beta E v\left(k', z'\right) \right\}.$$

We use the standard C.I.E.S. felicity function and Cobb-Douglas production function

$$\begin{aligned} U(c_t) &= \frac{c_t^{1-\sigma} - 1}{1-\sigma}, \quad \sigma > 0, \sigma \neq 1 \\ y_t &= z_t A k_t^{\alpha}. \end{aligned}$$

The calibrated parameter values are given in table 8.2.

Table 8.2: Parameter values of neoclassical growth model

σ	A	β	α	δ
0.5	1.0	0.9888	0.4	0.0241

The assumption of *i.i.d.* shocks in the previous section is very restrictive and we are often interested in generalizing it to correlated productivity shocks. Donaldson and Mehra (1983) solve the Brock-Mirman model with autocorrelated productivity shocks and they show that the transition functions for output, capital, consumption etc. converge to their stationary distributions.

The discrete state space of the shock process and the transition probability matrix are constructed using the method of Tauchen (1986) for approximating an autoregressive process with a discrete state space, treated in details in section 8.3.1. The Tauchen method is implemented in `Tauchen.m` and is called by the commands (see section 8.3.1 for explanations)

```
sigmaz = .0423;
[stsp,Pi] = tauchen(sigmaz,rho,7,.3) ;
z = stsp + 1;
```

The seven states of the technology shock are given by

$$z_t \in Z \equiv \left[\begin{array}{ccccccc} .9594 & .9729 & .9865 & 1 & 1.0135 & 1.0271 & 1.0406 \end{array} \right], \tag{8.16}$$

with the corresponding transition probabilities π_{ij}

$$\pi_{ij} = \Pr\left\{z_{t+1} = z_j | z_t = z_i\right\}.$$

The conditional probabilities (probability measures), satisfy

$$0 \leq \pi_{ij} \leq 1, \qquad \sum_{j=1}^{4} \pi_{ij} = 1,$$

and a matrix of transition probabilities, Π, constructed from the elements π_{ij}

$$\Pi = \left[\begin{array}{ccccccc} .7960 & .2033 & .0007 & 0 & 0 & 0 & 0 \\ .0780 & .7498 & .1717 & .0005 & 0 & 0 & 0 \\ .0001 & .0966 & .7595 & .1434 & .0003 & 0 & 0 \\ 0 & .0002 & .1184 & .7628 & .1184 & .0002 & 0 \\ 0 & 0 & .0003 & .1434 & .7595 & .0966 & .0001 \\ 0 & 0 & 0 & .0005 & .1717 & .7498 & .0780 \\ 0 & 0 & 0 & 0 & .0007 & .2033 & .7960 \end{array} \right]. \tag{8.17}$$

Notice that the sum of probabilities in each row is unity. The matrix of transition probabilities converges to a limiting probability distribution (here raised to the power 500)

$$\lim_{n\to\infty} \Pi^n = \Pi^* = \left[\begin{array}{ccccccc} .0453 & .1180 & .2097 & .2541 & .2097 & .1180 & .0453 \\ .0453 & .1180 & .2097 & .2541 & .2097 & .1180 & .0453 \\ .0453 & .1180 & .2097 & .2541 & .2097 & .1180 & .0453 \\ .0453 & .1180 & .2097 & .2541 & .2097 & .1180 & .0453 \\ .0453 & .1180 & .2097 & .2541 & .2097 & .1180 & .0453 \\ .0453 & .1180 & .2097 & .2541 & .2097 & .1180 & .0453 \\ .0453 & .1180 & .2097 & .2541 & .2097 & .1180 & .0453 \end{array} \right].$$

Each row of the limiting transition probability matrix Π^*, denoted by $\pi^*_{i\cdot}$, is an invariant or stationary probability distribution satisfying $\pi_{i\cdot}\Pi^* = \pi_{i\cdot}$, that is, it is the fixpoint of the operator Π^*. The expected value of the shock process is one.

The computational procedure is the same as in the previous section:

1. Define the parameters, compute the steady state value of the capital stock, k^*, and discretize the state space for capital. Define the grid on the stochastic shock and the transition probability matrix.

2. Compute the seven $(g \times g)$-dimensional consumption matrices conditional on the value of the stochastic shock, and compute the seven matrices of utility conditional on the stochastic shock.

3. Define seven initial $(g \times 1)$-dimensional zero-vectors as the initial value functions. Compute the first-period value functions.

4. Continue by iterating on Bellman's equation until convergence.

5. Compute the policy functions based on the final value functions.

6. Stochastic simulations: Generate a time path of shocks from the Markov chain and compute the time path of capital and consumption around the steady state level of capital by stochastic simulations n times.

The only difference from the previous section is the specification of the expected value when iterating on Bellman's equation and when computing the stochastic simulations. These steps will be discussed in more details.

Step 5. Iterate on Bellman's equation

The shock z_t is known at time t when decisions are taken so the state of the economy is known. Uncertainty is about next period's state and each row of the transition probability matrix is the probability distribution over next period's state when being in state i at time t. If the economy is in state i, then Bellman's equation is given by

$$\begin{aligned} Tv_i &= \left(\max_{k' \in X} \left\{Ui + \beta E v\left(k', z'\right)\right\}\right)' \\ &= \left(\max_{k' \in X} \left\{Ui + \beta \sum_{j=1}^{m} \pi_{ij} v\left(k', z_j'\right)\right\}\right)' \\ &= \left(\max_{k' \in X} \left\{Ui + \beta\left(\pi_{i1} v_1\left(k', z_1'\right) + \pi_{i2} v_2\left(k', z_2'\right) + \pi_{i3} v_3\left(k', z_3'\right)\right.\right.\right. \\ &\quad \left.\left.\left. + ... + \pi_{im} v_m\left(k', z_m'\right)\right) \cdot \mathbf{1}\right\}\right)' \end{aligned} \tag{8.18}$$

where the value functions are $(g \times 1)$ vectors and $\mathbf{1}$ is a $(1 \times g)$ vector of ones. Of course, there are seven value functions,[3] figure 8.6.

MATLAB program:

In the following, `Pi` is the transition probability matrix.

```
n = 1;
maxit = 5000;
limit = 1 *10^(-8);
distance1 = 1;
```

[3]The figures and the table in this section is computed for 401 gridpoints on capital, 2021 iterations on the value function, the sime series are simulated 5000 times for 196 periods (and then we cut off the first 24 observations).

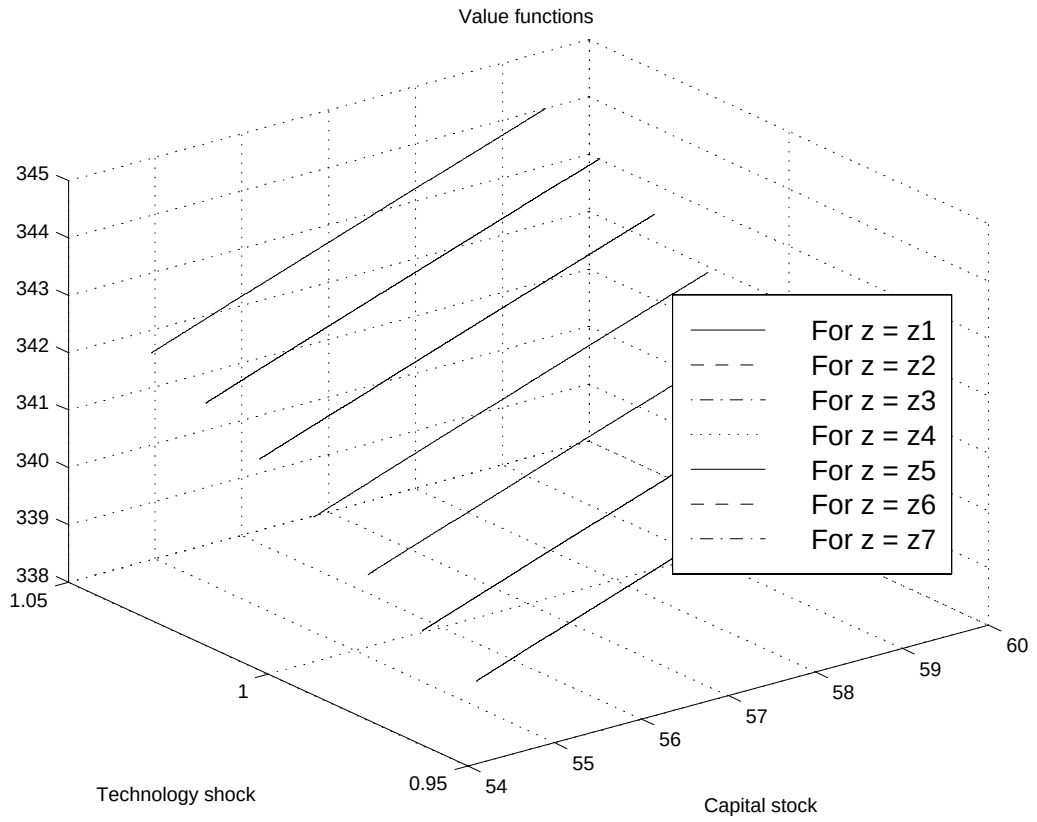

Figure 8.6: Value functions.

```
   .
   .
   distance7 = 1;
   while distance1 > limit | distance2 > limit | distance3 > limit |
distance4 > limit ...
    distance5 > limit | distance6 > limit | distance7 > limit & n <
maxit;
    B1 = beta*((Pi(1,1)*v1 + Pi(1,2)*v2 + Pi(1,3)*v3 + Pi(1,4)*v4 + Pi(1,5)*v5
+ Pi(1,6)*v6 + Pi(1,7)*v7)*o);
    Tv1 = (max(U1 + B1))';
    clear B1;
    distance1 = norm(abs(Tv1-v1(1:g,1)));

    B2 = beta*((Pi(2,1)*v1 + Pi(2,2)*v2 + Pi(2,3)*v3 + Pi(2,4)*v4 + Pi(2,5)*v5
+ Pi(2,6)*v6 + Pi(2,7)*v7)*o);
    Tv2 = (max(U2 + B2))';
```

```
 clear B2;
 distance2 = norm(abs(Tv2-v2(1:g,1)));
 .
 .
 .
 v1 = Tv1;
 .
 .
 v7 = Tv7;
 n = n+1;
end;
```

Step 6. Stochastic simulations

The stochastic simulations follow the same steps as for the *i.i.d.* shocks, though the generation of shocks from a Markov chain is somewhat more complicated. For that purpose, we use the function file `markovchain.m` where we have implemented the generation of a time series S with T observations which consist of integers from 1 to m, where m is the size of the transition probability matrix, here $m = 4$. A more detailed specification of the generation of a shock process from a discrete time Markov chain is given in section 8.3.2.

The value for each of the T observations of the time series of shocks, S, determines the state of the system in the next period and which policy function generates next period's capital stock and this period consumption. We use the following step:

1. Make a loop over number of simulations, $nsim$.
2. Generate a $(T \times 1)$ time series, S, from the Markov chain.
3. Make a loop over time periods, T.
   (a) If the next period state is i, then use policy functions i to determine next period's capital stock and this period consumption.
   (b) Compute the percentage deviations from steady state for k, z, and c. Use these to compute the percentage deviation of output, investments, and real interest rates from their steady state.
   (c) Compute summary statistics for all of these variables and save the summary statistics in a vector.
4. End the if loop, end the time loop, and end the number of simulations loop.

MATLAB program.

```
for i = 1:nsim;
 sk(1,1) = kstar;
 m = g-((g-1)/2);
```

Table 8.3: Population second moments

	Standard deviation	Relative standard deviation	Crosscorrelation with y
y	2.23	1.00	1.00
c	1.74	0.77	0.83
i	4.91	2.26	0.86
r	1.45	0.67	0.68
k	1.64	0.72	0.73

```
S = markovchain(Pi,T,2);
for t=1:T;
 if S(t,1) == 1;
  sk(t+1,1) = policyk1(m);
  sc(t,1) = policyc1(m);
  m = find(k == sk(t+1,1));
  shock(t,1) = z(1,1);
 elseif S(t,1) == 2;
  sk(t+1,1) = policyk2(m);
  sc(t,1) = policyc2(m);
  m = find(k == sk(t+1,1));
  shock(t,1) = z(1,2);
 elseif S(t,1) == 3;
  sk(t+1,1) = policyk3(m);
  sc(t,1) = policyc3(m);
  m = find(k == sk(t+1,1));
  shock(t,1) = z(1,3);
 elseif S(t,1) == 4;
  sk(t+1,1) = policyk4(m);
  sc(t,1) = policyc4(m);
  m = find(k == sk(t+1,1));
  shock(t,1) = z(1,4);
 end;
end;
% Compute summary-statistics here...
end;
```

We have computed standard business cycle summary statistics on the model generated time series based on 5000 simulations over 196 periods and cut off the first 24 periods before computing summary statistics based on the unfiltered series.

The stochastic simulations give the expected results: Investments are more volatile than consumption and more volatile than output, investments are higher positively correlated with output than consumption, table 8.3.

The computational method of this section is implemented in the program `Dynprogch83.m`, `markovchain.m`, and `Tauchen.m`.

8.3.1 Finite State Markov Chain Approximations

The RBC model of chapter 3 was specified with an autoregressive representation of the shock process. Such a continuous state shock process cannot be used when using dynamic programming and value function iterations where we need the shock process to have a discrete state space. When the dimension of the state vector and transition matrix is small, it may well be possible by using ad hoc methods to approximate the autoregressive shock process such that the realization of the Markov chain is a "realistic" approximation to an autoregressive process. With larger state spaces, this may be difficult. Tauchen (1986) has developed a method for finite state Markov chain approximations to univariate and vector autoregressive stochastic processes. We have implemented the method for a univariate autoregressive process in the program `Tauchen.m`.

Let z_t be a zero mean stochastic process generated by a first-order Markov process

$$z_{t+1} = \rho \cdot z_t + \omega_{t+1} \tag{8.19}$$

where ω_{t+1} is Gaussian white noise. The constant variance of z_t is

$$\sigma_z^2 = \frac{1}{1-\rho^2} \cdot \sigma_\omega^2$$

The cumulative distribution function of ω_{t+1} is

$$\Pr\left[\omega_{t+1} \leq u\right] = F\left(\frac{u}{\sigma_\omega}\right)$$

Let z_t be the discrete state stochastic process that approximates the continuous state autoregressive stochastic process (8.19), and let $z_1 < z_2 < ... < z_m$ be the m discrete values of the state space of z_t.

The state space of z_t is determined so that the values are in the range of $\pm r$ times the standard deviation of the shock process. Let z_m be a multiple r of the unconditional standard deviation of the autoregressive process, z_t

$$z_m = r \cdot \sigma_z = r \cdot \sqrt{\frac{1}{1-\rho^2} \cdot \sigma_\omega^2}$$

Let z_1 be given by

$$z_1 = -z_m$$

and let the state space of z_t be constructed as an equispaced grid over the values $\begin{bmatrix} z_1 & z_m \end{bmatrix}$.

The transition matrix, Π, is calculated as follows. The elements of the transition matrix, denoted by π_{ij} are by definition

$$\pi_{ij} = \Pr\left\{z_{t+1} = z_j | z_t = z_i\right\}.$$

Let the distance between two grid point be given by $d = z_j - z_{j-1}$. For each row i, if j is between 2 and $m-1$, set

$$\begin{aligned} \pi_{ij} &= \Pr\left\{z_j - \frac{d}{2} \leq \rho z_i + \omega_t \leq z_j + \frac{d}{2}\right\} \\ &= F\left(\frac{z_j - \rho z_i + \frac{d}{2}}{\sigma_\omega}\right) - F\left(\frac{z_j - \rho z_i - \frac{d}{2}}{\sigma_\omega}\right) \end{aligned}$$

Otherwise

$$\begin{aligned} \pi_{i1} &= F\left(\frac{z_1 - \rho z_i + \frac{d}{2}}{\sigma_\omega}\right) \\ \pi_{im} &= 1 - F\left(\frac{z_m - \rho z_i - \frac{d}{2}}{\sigma_\omega}\right) \end{aligned}$$

The computation of the state space and the transition matrix is implemented in the function file `Tauchen.m`. which is called by the command `[stsp,Pi] = tauchen(sigmaw,rho,m,r);` where the outputs `stsp` is the state space and `Pi` is the transition matrix, the inputs `sigmaw` is the standard deviation of the disturbance term, ω_{t+1}, `rho` is the autocorrelation coefficient of (8.19), `m` is the dimension of the state space, and `r` is the multiplum of the standard deviation for the grid on the state space. The default is `r = .3`.

For the purpose of generating the state space (8.16) and the transition matrix (8.17), we used the command

```
sigmaw = .0423;
[stsp,Pi] = tauchen(sigmaw,.95,7,.3);
z = stsp + 1;
```

The value of `r` can be determined so as to generate approximately a standard deviation of the shock process and an autocorrelation of the shock process which is close to that of the autoregressive shock process of chapter 3. The standard deviation of z_t is given by

$$\sigma_z = \sqrt{\frac{1}{1-\rho^2}\cdot\sigma_\omega^2} = \sqrt{\frac{1}{1-.95^2}\cdot.0923^2} = .2956$$

The value of `r` is found by simulating the stochastic process and computing the autocorrelation coefficient. This is implemented in `tauchenanalysis.m` which simulates the stochastic process from a Markov chain and computes the standard deviation and autocorrelation coefficient of the shock process. The combination of `sigmaw` and `r` can be chosen so that the standard deviation of output in the model is approximately equal to the standard deviation of output in data.

8.3.2 Generating a Realization of a Discrete State Markov Chain

We need to generate a realization of the Markov chain for each stochastic simulation. We have implemented this in the function file `markovchain.m` The command is `S = markovchain(Pi,T,S0);` where `Pi` is the transition probability matrix, `T` is the number of observations, and `S0` is the initial state of the system. In order to limit dependence on the initial state, it is recommended to cut off the first 12 observations in the generated time series before computing summary statistics. Output from the function files is a time series `S` with T observations where we have implemented the generation of a time series `S` with T observations which consist of integers from 1 to n, where n is the size of the transition probability matrix.

8.4 A Real Business Cycle Model

The model in this section is identical to the RBC model of chapter 6. The benevolent social planner maximizes the same utility function as the representative household subject to the aggregate resource constraint and some non-negativity constraints[4]

$$\max_{\{c_t,h_t,k_{t+1}\}_{t=0}^{\infty}} E_0\left[\sum_{t=0}^{\infty}\beta^t u(c_t,1-h_t)\right],$$

subject to

$$\begin{aligned}
\gamma k_{t+1} &\leq i_t + (1-\delta)k_t, \quad \delta\in[0,1] \\
c_t + i_t &\leq z_t f(k_t,h_t) \\
c_t,k_t,h_t,l_t &\geq 0,\ \forall t \\
k_0 &> 0, \quad \text{given} \\
h_t + l_t &= 1,\ \forall t,
\end{aligned}$$

[4]When solving a social planner's problem, there is no distinction between individual and aggregate policy functions and value functions.

where z_t is a technology shock generated by an m-state Markov chain.

Start by substituting the constraints into the utility function and formulate Bellman's equation using the prime notation

$$v\left(k,z\right)=\max_{h\in H,k'\in X}\left\{\left[u(zf(k,h)-\gamma k'+\left(1-\delta\right)k,1-h\right]+\beta Ev\left(k',z'\right)\right\}$$

subject to

$$\begin{aligned} h &\in H=(0,1) \\ k,k' &\in X. \end{aligned}$$

The size and complexity of the model is increased by the fact that there is one more control variable and that the dimension of the utility matrix increases since utility depends both on consumption and on leisure. Let the utility function be given by

$$U(c_t,1-h_t)=\frac{\left(c_t^a(1-h_t)^{1-a}\right)^{1-\sigma}}{1-\sigma},\quad 0<\sigma,\sigma\neq 1,$$

and let the production function be given by

$$y_t=z_tk_t^{\alpha}h_t^{1-\alpha}$$

Table 8.4: Parameter values of RBC model

σ	a	$\bar{h}$	β	γ	α	δ
0.5	0.3730	0.3	0.9888	1.0045	0.4	0.0241

The computational procedure is almost the same as in the previous section:

1. Define the parameters, compute the steady state value of the capital stock, k^*, and discretize the state space of capital. Define the grid on the stochastic shock with m values and the $(m\times m)$ transition probability matrix. Define a grid on labor supply (or leisure) with n values around the steady state value of labor (leisure).

2. Compute the m $(g\times g)$-dimensional consumption and utility matrices conditional on the value of the stochastic shock and for each value of labor supply. Stack the matrices to m $((g\cdot n)\times g)$-dimensional matrices, one for each value of the technology shock.

3. Define m initial $((g \cdot n) \times g)$-dimensional zero-vectors as the initial value functions. Compute the m first-period value functions.

4. Continue by iterating on Bellman's equation until convergence.

5. Compute the policy functions based on the final value functions.

6. Stochastic simulations: Generate a realization of the Markov chain and compute the time path of capital, consumption, and labor supply. Simulate the model $nsim$ times for T periods.

Step 1. Define grid on capital, shocks, and labor supply:

Let the shock process be defined as in the previous section as a seven state Markov chain

$$z_t \in Z \equiv \left[\begin{array}{ccccccc} .9594 & .9729 & .9865 & 1 & 1.0135 & 1.0271 & 1.0406 \end{array} \right],$$

with the corresponding transition probabilities π_{ij}

$$\pi_{ij} = \Pr\left\{z_{t+1} = z_j | z_t = z_i\right\}.$$

The conditional probabilities (probability measures), satisfy

$$0 \leq \pi_{ij} \leq 1, \qquad \sum_{j=1}^{4} \pi_{ij} = 1,$$

and a matrix of transition probabilities, Π, constructed from the elements π_{ij}

$$\Pi = \left[\begin{array}{ccccccc} .7960 & .2033 & .0007 & 0 & 0 & 0 & 0 \\ .0780 & .7498 & .1717 & .0005 & 0 & 0 & 0 \\ .0001 & .0966 & .7595 & .1434 & .0003 & 0 & 0 \\ 0 & .0002 & .1184 & .7628 & .1184 & .0002 & 0 \\ 0 & 0 & .0003 & .1434 & .7595 & .0966 & .0001 \\ 0 & 0 & 0 & .0005 & .1717 & .7498 & .0780 \\ 0 & 0 & 0 & 0 & .0007 & .2033 & .7960 \end{array} \right].$$

Notice that the sum of probabilities in each row is unity. The matrix of transition probabilities converges to a limiting probability distribution (here raised to the power 500)

$$\lim_{n\to\infty} \Pi^n = \Pi^* = \begin{bmatrix} .0453 & .1180 & .2097 & .2541 & .2097 & .1180 & .0453 \\ .0453 & .1180 & .2097 & .2541 & .2097 & .1180 & .0453 \\ .0453 & .1180 & .2097 & .2541 & .2097 & .1180 & .0453 \\ .0453 & .1180 & .2097 & .2541 & .2097 & .1180 & .0453 \\ .0453 & .1180 & .2097 & .2541 & .2097 & .1180 & .0453 \\ .0453 & .1180 & .2097 & .2541 & .2097 & .1180 & .0453 \\ .0453 & .1180 & .2097 & .2541 & .2097 & .1180 & .0453 \end{bmatrix}.$$

The transition probability matrix creates a time series for the shock process which is highly autocorrelated.

Let the grid on labor supply be given by n grid points equally spaced around the steady state labor supply, $\bar{h}$, from $0.9625 \cdot \bar{h}$ to $1.0375 \cdot \bar{h}$.[5]

Step 2. Construction of consumption and utility matrices.

The utility matrix is defined from the utility function where we use the prime notation

$$\begin{aligned} U(c, 1-h) &= \frac{\left(c^a (1-h)^{1-a}\right)^{1-\sigma}}{1-\sigma}, \quad 0 < \sigma, \sigma \neq 1 \\ &= \frac{\left(\left[z k^{\alpha} h^{1-\alpha} + (1-\delta)\, k - \gamma k'\right]^a (1-h)^{1-a}\right)^{1-\sigma}}{1-\sigma} \end{aligned}$$

and we stack utility matrices for each value of labor supply. For each value of the technology shock, $l = 1, 2, ..., 7$, we create a utility matrix which is stacked from n utility matrices, one for each value of labor supply, $i = 1, 2, ..., n$

$$U_l = \begin{bmatrix} U_{l1} \\ U_{l2} \\ \vdots \\ U_{ln} \end{bmatrix}.$$

MATLAB program

The following is an example of the construction of the first consumption and utility matrix for the shock taking the value of the first element in the state space of the shock, `z(1,1)`. We start by constructing the first of the n consumption matrices, then impose the restriction that the elements is non-negative, and then construct the first matrix of the utility matrix, U_{l1}. The utility matrix U_l is constructed with a loop over the n values of labor supply.

```
I = ones(g,1);
for i = 1:n;
```

[5] In this section, we use 131 grid points for capital, 17 grid points for labor supply, 1903 iterations on Bellman's equation, the time series are simulated 5000 times for 196 periods.

```
    c = z(1,1)*(k.^alpha)*((h(i,1))^(1-alpha)) + (1-delta)*k;
    C = I * c' - (gamma*k)*I';
    for j = 1:g;
     for w = 1:g;
      if C(j,w) <= 0;
      C(j,w) = 0;
     end;
    end;
    if sigma == 1;
     U1(i*g-g+1:g*i,:)  = a*ln(C) + (1-a)*ln(1-h(i,1));
    else;
     U1(i*g-g+1:g*i,:)  = (1/(1-sigma)) * ((C.^a)...
     *((1-h(i,1))^(1-a))).^(1-sigma);
    end;
   end;
```

Step 4. Iterations on the value function

The iterations are performed as in the previous section except for the construction of the stacked matrices over which the maximization takes place. Value functions are plotted in figure 8.7.

MATLAB program

The value function iteration procedure is only illustrated for the first of the seven value functions. Iterations continue until convergence where the convergence criteria is determined by a matrix norm.

```
   nn = 1;
   limit = 1 *10^(-8);
   distance1 = 1;
   while distance1 > limit | distance2 > limit | distance3 > limit |
distance4 > limit ...
    distance5 > limit | distance6 > limit | distance7 > limit;
   %while nn < iter ; % Value function iterations
    %disp('Iteration no.');
    disp(nn);
    B1 = beta*((Pi(1,1)*v1 + Pi(1,2)*v2 + Pi(1,3)*v3 + Pi(1,4)*v4 + Pi(1,5)*v5
+ Pi(1,6)*v6 + Pi(1,7)*v7))*o;
    Tv1 = (max(U1 + B1))';
    clear B1 B2 B3 B4 B5 B6 B7;
    distance1 = norm(abs(Tv1-v1(1:g,1)));
    for i = 1:n;
    v1(i*g-g+1:g*i,1) = Tv1;
    end;
    clear Tv1 Tv2 Tv3 Tv4 Tv5 Tv6 Tv7;
    nn = nn+1;
   end;
```

Step 5. Optimal policy functions

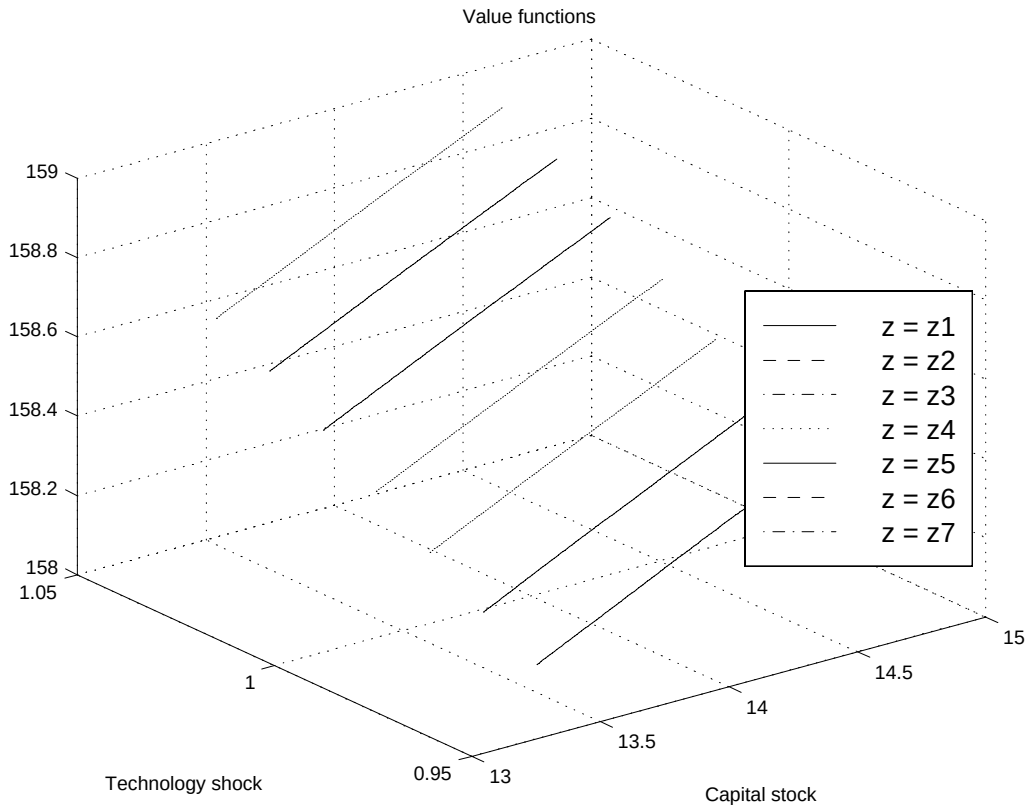

Figure 8.7: Value functions for the RBC model

First the policy functions for capital are computed, then the policy functions for labor supply, and last the policy functions for consumption. The main difference from the computation of policy functions in the last section is the larger matrices so we have to keep track on the indexes for the maximized values of capital and the corresponding for labor supply, figure 8.8.

MATLAB program

The computation of optimal policy functions is illustrated for the first of the seven policy functions for capital, labor supply, and consumption.

```
   B1 = beta*((Pi(1,1)*v1 + Pi(1,2)*v2 + Pi(1,3)*v3 + Pi(1,4)*v4 + Pi(1,5)*v5
+ Pi(1,6)*v6 + Pi(1,7)*v7))*o;
   B2 = beta*((Pi(2,1)*v1 + Pi(2,2)*v2 + Pi(2,3)*v3 + Pi(2,4)*v4 + Pi(2,5)*v5
+ Pi(2,6)*v6 + Pi(2,7)*v7))*o;
   .
   .
   B7 = beta*((Pi(7,1)*v1 + Pi(7,2)*v2 + Pi(7,3)*v3 + Pi(7,4)*v4 + Pi(7,5)*v5
+ Pi(7,6)*v6 + Pi(7,7)*v7))*o;
   %
```

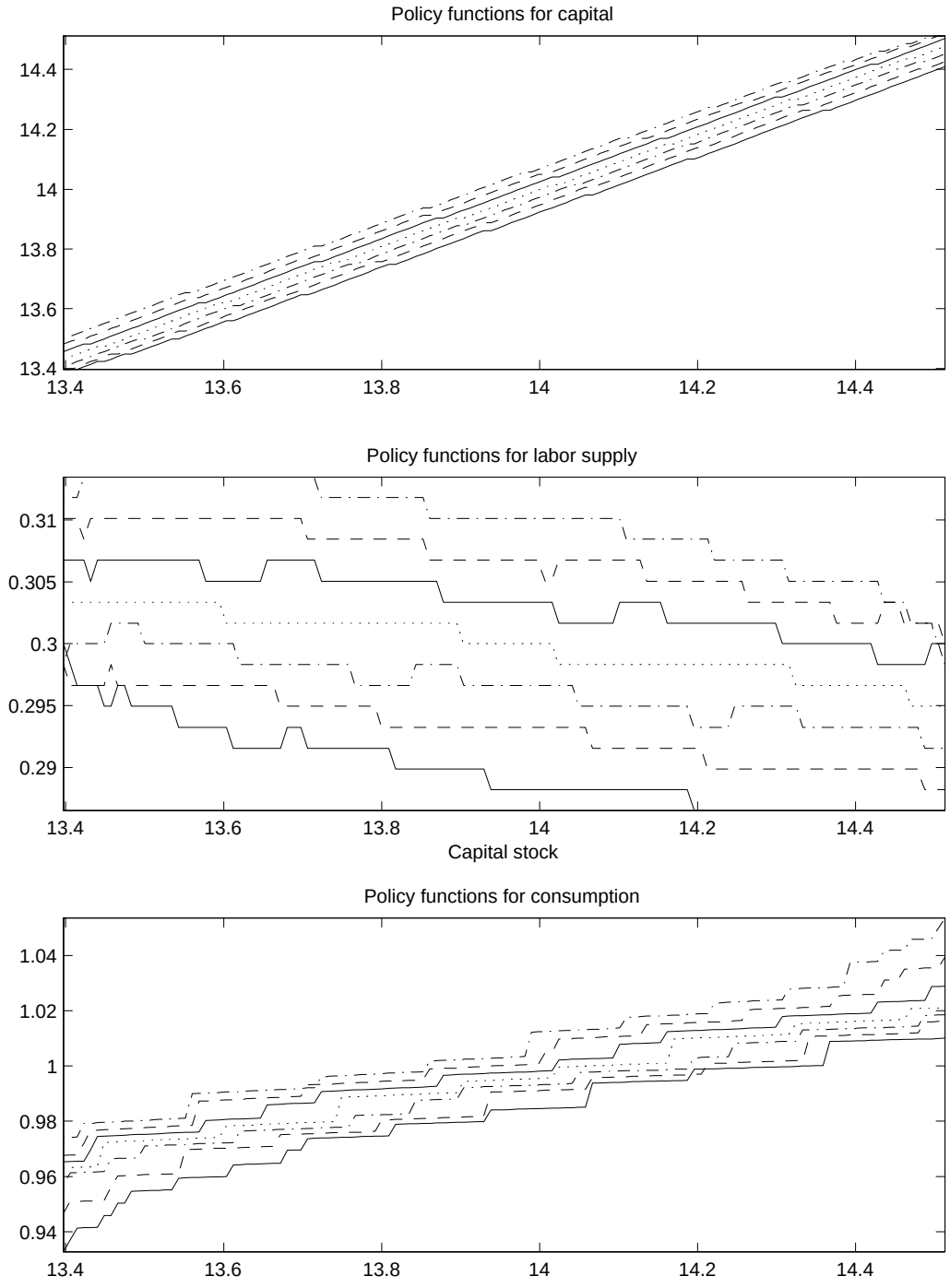

Figure 8.8: Policy functions for capital, labor supply, and consumption

Table 8.5: Population second moments for RBC model

	Standard deviation	Relative standard deviation	Crosscorrelation with y	First autocorrelation
y	2.77	1.00	1.00	0.93
c	2.08	0.74	0.83	0.77
i	5.97	2.19	0.89	0.83
h	1.07	0.40	0.71	0.66
w	2.14	0.76	0.93	0.87
k	1.99	0.71	0.68	0.63
z	1.76	0.64	0.99	0.93

```
[Tv1,j1] = max(U1 + B1);
[Tv2,j2] = max(U2 + B2);
.
.
[Tv7,j7] = max(U7 + B7);
% Index for capital
k1 = j1 - (g * (fix((j1-1)/g)));
% Policy function for capital
policyk1 = (k(k1))';
% Index for labor supply
l1 = fix(j1/(g+1)) + 1;
lll = [l1 l2 l3 l4 l5 l6 l7 ];
% Policy function for labor supply
policyh1 = (h(l1))';
% Policy function for consumption
policyc1 = (z(1,1)*(k.^alpha).*(policyh1.^(1-alpha))' + (1-delta)*k
- gamma*policyk1' )';
```

Step 6. Stochastic simulations

We have computed standard business cycle summary statistics on the model generated time series based on 5000 simulations for 172 periods (196 periods and cut off the first 24 observations).

A single realization of the simulation is illustrated in figure 8.9.

MATLAB program

```
for i = 1:nsim;
 kt = kstar; % Initial capital stock
 sk(1,1) = kstar; % Simulations around steady state
 m = g-((g-1)/2);
 S = markovchain(Pi,T,4); % Start in state 4;
 for t = 1:T;
  if S(t,1) == 1;
   sk(t+1,1) = policyk1(m);
```

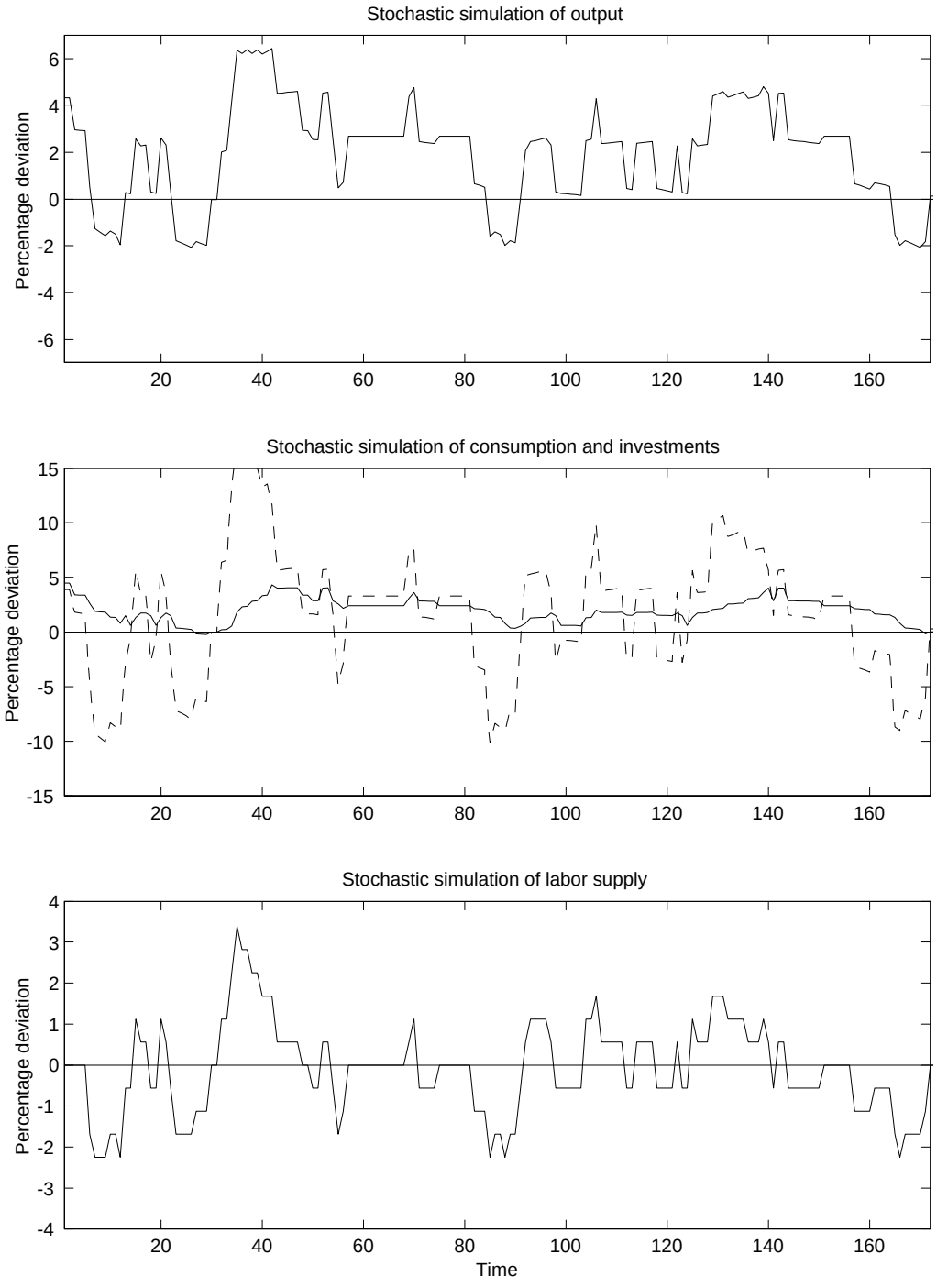

Figure 8.9: Stochatic simulation of RBC-model.

```
      sc(t,1) = policyc1(m);
      sh(t,1) = policyh1(m);
      m = find(k == sk(t+1,1));
      shock(t,1) = z(1,1);
     elseif S(t,1) == 2;
      sk(t+1,1) = policyk2(m);
      sc(t,1) = policyc2(m);
      sh(t,1) = policyh2(m);
      m = find(k == sk(t+1,1));
      shock(t,1) = z(1,2);
     end;
    end;
    sk = sk(1:T,1);
    sc = sc(1:T,1);
    sh = sh(1:T,1);
    % Percentage deviations from steady state:
    shockhat = shock - 1;
    khat = (sk -kstar)/kstar;
    chat = (sc - cbar)/cbar;
    hhat = (sh - hbar)/hbar;
    yhat = shockhat + (alpha * khat) + (1-alpha)*hhat;
    ihat = (1/shi)*yhat - ((shc/shi)*chat);
    Rhat = shockhat + (alpha-1) * khat + (1-alpha)*hhat;
    what = shockhat + alpha*khat + (-alpha)*hhat;
    rhat = Rhat;
   % Compute summmary statistics...;
   end;
```

The computational method of this section is implemented in the program `Dynprogch84.m`, `markovchain.m`, and `tauchen.m`.

8.5 The Curse of Dimensionality

In order to compute the value function, it is necessary to choose a grid on the possible values of the capital stock, labor supply, and the exogenous shock. In order to get a fine approximation, one has to choose a very fine grid. In the computations in section 8.4, we chose 121 grid points for capital and 17 grid points for labor supply. With shocks following a seven state Markov process, this gives 7 matrices of the size $(17 \cdot 131 \times 131)$, that is, 7 matrices with 291740 elements. Manipulating matrices of this size is very RAM and CPU time intensive operations. There are few instances in modern business cycle research where computation time is critical, this is one of them.

8.6 MATLAB Programs

`Dynprogch81.m` Deterministic Brock-Mirman model. Analysis of convergence towards steady state as in section 8.1.

`Dynprogch82.m` Stochastic Brock-Mirman model. Stochastic simulations with an i.i.d. shock taking two values as in section 8.2.

`Dynprogch83.m` Stochastic neoclassical growth model (Ramsey-model) with shocks following a four-state Markov chain. Stochastic simulations as in section 8.3.

`Dynprogch83b.m` The same as above but with shocks following a seven-state Markov chain.

`Dynprogch84.m` RBC model with shocks following a seven-state Markov chain as in section 8.4

`Markovchain.m` Function file which generates a user-specified realization of a Markov chain.

`Tauchen.m` Finite state approximation to a Markov chain

`Tauchenanalysis.m` Analyzes the stochastic properties of the shock process generated by `Tauchen.m`. Computes standard deviation and autocorrelation coefficient. This program is used to determine the parameter inputs in `Tauchen.m`.

`Policyiterations1.m` Solves the stochastic Ramsey model with policy function iterations using Howard's (1960) policy improvement algorithm.