PhD Macroeconomics 1
12. Infinite Horizon Dynamic Programming

Karl Whelan

School of Economics, UCD

Autumn 2023

Infinite Horizon Problems

- In macroeconomics, optimisation problems often have infinite horizons, i.e. they are of the form:

$$\max_{u_t, x_t} \sum_{t=0}^{\infty} \beta^t F(x_t, u_t)$$

subject to

$$x_{t+1} = g(x_t, u_t)$$

- Because there is no terminal period, this problem cannot be solved by backward induction. But there is good news. In the infinite horizon case, the problem starting tomorrow looks exactly as it does starting today, as long as the initial state is the same. Thus, we can drop the time index from the value functions and the Bellman equation has the form

$$V(x_t) = \max_{u_t} \left[F(x_t, u_t) + \beta V(g(x_t, u_t))\right]$$

- These problems will also generally require a transversality condition of the form $\lim_{t\to\infty} \beta^t F(x_t, u_t) = 0$

Iterative Solution for the Value Function

- In general there is no analytical solution for value functions in infinite horizon dynamic programming problems.
- However, in most well-formulated economics problems with $F(x_t, u_t)$ increasing in x_t and with discounting, you can use fancy maths (functional analysis ...) to show that a unique value function exists.
- One strategy is **Value Function Iteration**: Guess a value function and use the Bellman equation to iterate towards the right solution. In other words, guess $V^0(x)$ and then calculate

$$V^1(x) = \max_u \left[F(x,u) + \beta V^0(g(x,u)) \right]$$

- Keep iterating on

$$V^n(x) = \max_u \left[F(x,u) + \beta V^{n-1}(g(x,u)) \right]$$

  until the value functions V^n and V^{n-1} are sufficiently close at all points.
- In most cases in macroeconomics, the set-up of the model means this method will converge to a unique value function.

Numerical Implementation of Value Function Iteration

- In theory, the iterative process just described requires us to solve for $V^n(x)$ at every possible value of x.
- In practice, computers can't evaluate a value function at every single feasible real numbered value for x, so the iteration method is implemented by picking a number of points on a grid and making assumptions about the behaviour of the value function at points in between.
- In our implementations, we will evaluate the value functions at a finite number of grid points for the relevant state variable and then use linear interpolation to represent the points not explicitly on the grid.
- There are more sophisticated methods that can be more efficient. For example, you can approximate the value function as a function of a set of polynomials (e.g. Chebyshev polynomials) updating the set of coefficients with each iteration.

Optimal Consumption in a Representative Agent Economy

- Here we will discuss solving a simple deterministic optimal consumption model. Effectively, it is a deterministic RBC model with labor input set exogenously to one.

$$\max_{C_t, K_t} \sum_{t=0}^{\infty} \beta^t \frac{C_t^{1-\sigma}}{1-\sigma}$$

subject to

$$C_t = AK_{t-1}^{\alpha} + (1-\delta) K_{t-1} - K_t$$

- The value function for this model is

$$V(K_{t-1}) = \max_{C_t} \left[\frac{C_t^{1-\sigma}}{1-\sigma} + \beta V \left(AK_{t-1}^{\alpha} + (1-\delta) K_{t-1} - C_t \right) \right]$$

- Note that solving the value function automatically implies solving for the optimal "policy function" $C(K_{t-1})$ for how much to consume given how much capital you have. Each guess for the value function implies a set of guesses for how much you should be consuming at each capital level.

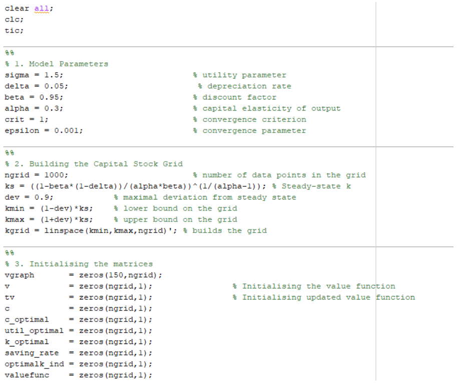
Numerical Method

- The value function for this model is $V(K_{t-1})$.
- As with previous problems, we are not going to come up with an analytical solution but instead guess one numerically.
- And again we will do this by calculating a value function at a set of grid points for the capital stock.
- How do we set the upper and lower grid points for the capital stock to use in calculating the function?
- This model has an analytical solution for the steady-state capital stock

$$K^* = \left(\frac{1 - \beta (1 - \delta)}{\alpha \beta} \right)^{\frac{1}{(\alpha - 1)}}$$

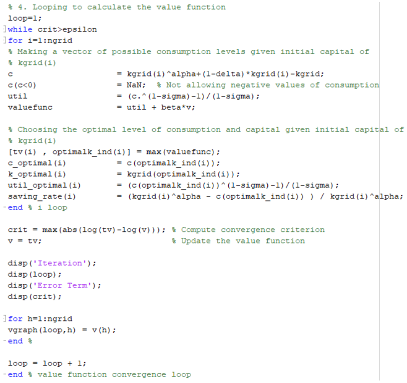
- So we set the grid by searching above and below this level of capital.
- In the example below, we go as low as 10% of the steady-state capital stock and as high as 190% of it.

Value Function Iteration Code: Preliminaries



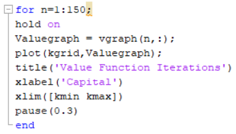
```
clear all;
clc;
tic;

%%
% 1. Model Parameters
sigma = 1.5;                        % utility parameter
delta = 0.05;                        % depreciation rate
beta = 0.95;                        % discount factor
alpha = 0.3;                        % capital elasticity of output
crit = 1;                           % convergence criterion
epsilon = 0.001;                    % convergence parameter

%%
% 2. Building the Capital Stock Grid
ngrid = 1000;                       % number of data points in the grid
ks = ((1-beta*(1-delta))/(alpha*beta))^(1/(alpha-1)); % Steady-state k
dev = 0.9;              % maximal deviation from steady state
kmin = (1-dev)*ks;      % lower bound on the grid
kmax = (1+dev)*ks;      % upper bound on the grid
kgrid = linspace(kmin,kmax,ngrid)'; % builds the grid

%%
% 3. Initialising the matrices
vgraph       = zeros(150,ngrid);
v            = zeros(ngrid,1);              % Initialising the value function
tv           = zeros(ngrid,1);              % Initialising updated value function
c            = zeros(ngrid,1);
c_optimal    = zeros(ngrid,1);
util_optimal = zeros(ngrid,1);
k_optimal    = zeros(ngrid,1);
saving_rate  = zeros(ngrid,1);
optimalk_ind = zeros(ngrid,1);
valuefunc    = zeros(ngrid,1);
```

Value Function Iteration Code: Main Code



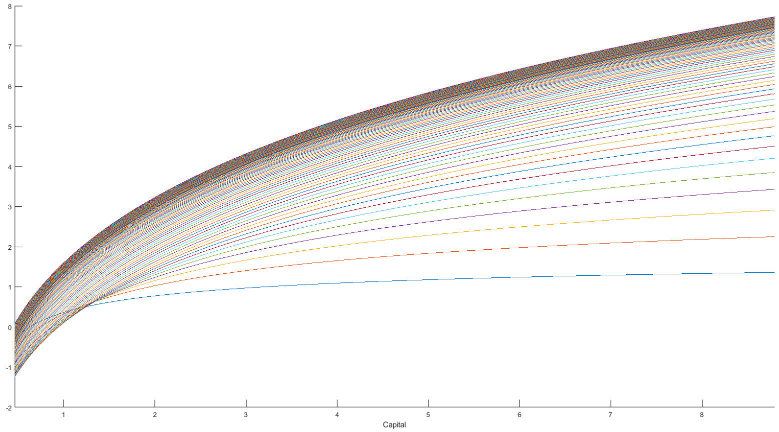
```
% 4. Looping to calculate the value function
loop=1;
while crit>epsilon
for i=1:ngrid
% Making a vector of possible consumption levels given initial capital of
% kgrid(i)
c                     = kgrid(i)^alpha+(1-delta)*kgrid(i)-kgrid;
c(c<0)                = NaN;  % Not allowing negative values of consumption
util                  = (c.^(1-sigma)-1)/(1-sigma);
valuefunc             = util + beta*v;

% Choosing the optimal level of consumption and capital given initial capital of
% kgrid(i)
[tv(i) , optimalk_ind(i)] = max(valuefunc);
c_optimal(i)          = c(optimalk_ind(i));
k_optimal(i)          = kgrid(optimalk_ind(i));
util_optimal(i)       = (c(optimalk_ind(i))^(1-sigma)-1)/(1-sigma);
saving_rate(i)        = (kgrid(i)^alpha - c(optimalk_ind(i)) ) / kgrid(i)^alpha;
end % i loop

crit = max(abs(log(tv)-log(v))); % Compute convergence criterion
v = tv;                          % Update the value function

disp('Iteration');
disp(loop);
disp('Error Term');
disp(crit);

for h=1:ngrid
vgraph(loop,h) = v(h);
end %

loop = loop + 1;
end % value function convergence loop
```

Making a “Movie” of Value Function Iteration



```
for n=1:150;
hold on
Valuegraph = vgraph(n,:);
plot(kgrid,Valuegraph);
title('Value Function Iterations')
xlabel('Capital')
xlim([kmin kmax])
pause(0.3)
end
```

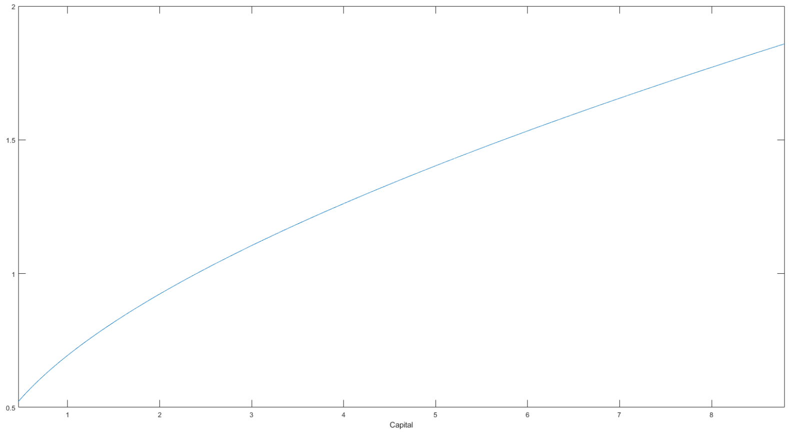
Infinite Horizon Value Function Iteration



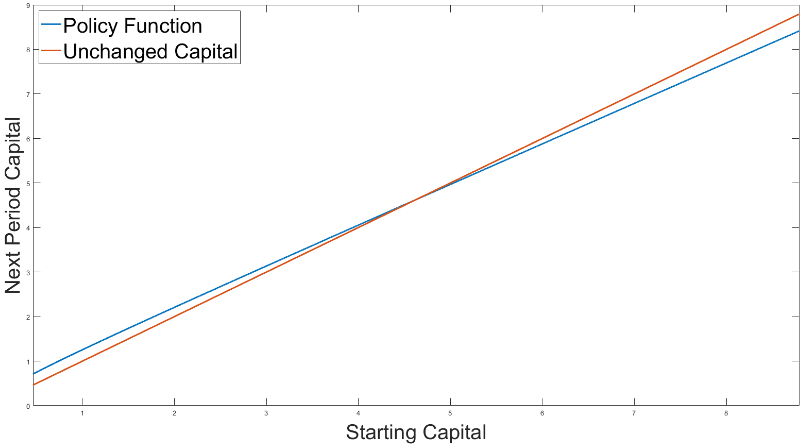
Optimal Consumption in a Representative Agent Economy

- As well as solving for the value function, we have derived the “optimal policy rule”, i.e. for each starting level of capital, what is the optimal level of consumption this period and thus capital next period?
- The next page shows the optimal consumption rule and the following page interprets this as an optimal capital rule.
- The consumption policy rule shows consumption as a concave function of capital with an ever-declining marginal propensity to consume as the capital stock rises.
- The capital policy rule shows that the model converges to its long-run steady state. At levels of capital below K^*, the optimal policy is to increase the capital stock, above K^*, the optimal policy is to reduce capital.
- This means that, whatever the starting stock of capital, the model converges to a specific K^* and its corresponding value of $C^* = A_t \left(K^*\right)^{\alpha} - \delta K^*$
- The two subsequent pages illustrate the transition dynamics for consumption and capital.

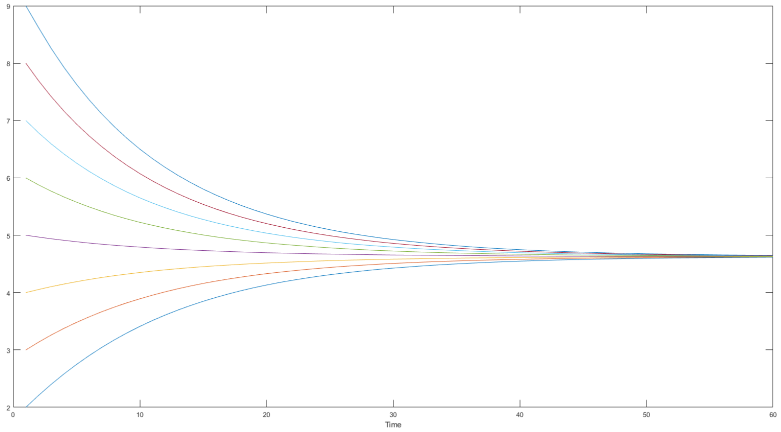
Infinite Horizon Optimal Consumption Rule



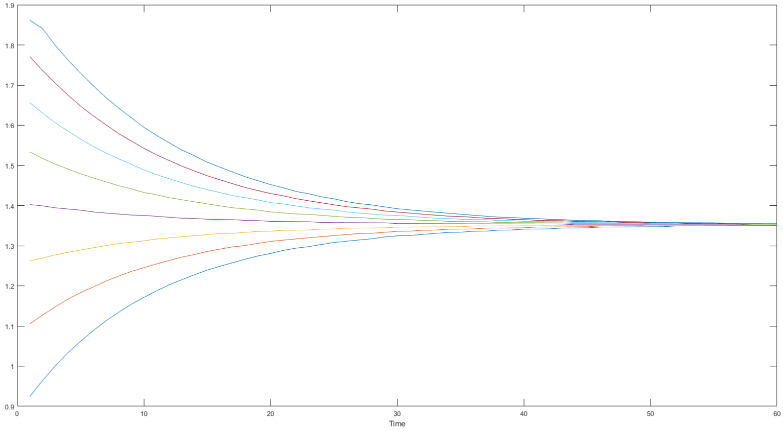
Infinite Horizon Optimal Capital Rule



Transition Dynamics for Capital



Transition Dynamics for Consumption Starting from Different Capital Levels



Part I

Infinite Horizon Stochastic Dynamic Programming

Stochastic Dynamic Programming with Markov Chain Uncertainty

- Suppose we alter the problem to be of the form:

$$\max_{u_t, x_t} E_0 \left[\sum_{t=0}^{\infty} \beta^t F(x_t, u_t, s_t) \right]$$

subject to

$$x_{t+1} = g(x_t, u_t, s_t)$$

where s_t is a stochastic variable that evolves according to a Markov chain with N states. The Bellman equation is now

$$V(x_t, s_t) = \max_{u_t} \left[F(x_t, u_t, s_t) + \beta E_t \left[V(g(x_t, u_t, s_t), s_{t+1}) \mid s_t \right] \right.$$

- Instead of just a single value function, we have separate value function for each of the N possible values of stochastic variable. The term

$$E_t \left[V(g(x_t, u_t, s_t), s_{t+1}) \mid s_t \right]$$

will be a probability-weighted average of each of these value functions, where the probabilities reflect the chance of moving from the current state to each of the N possible states next period.

A Stochastic Version of Our Optimal Consumption Model

- We can use methods we already have seen for finite-horizon stochastic models to solve a stochastic version of our optimal consumption model. Effectively, this is a simplified RBC model.
- Specifically, we will solve the problem

$$\max_{C_{t+k}, K_{t+k}} E_t \sum_{k=0}^{\infty} \beta^t \frac{C_{t+k}^{1-\sigma}}{1-\sigma}$$

subject to

$$K_t = A_t K_{t-1}^{\alpha} - C_t + (1-\delta) K_{t-1}$$

where A_t is a stochastic variable that evolves according to a Markov chain with N states.

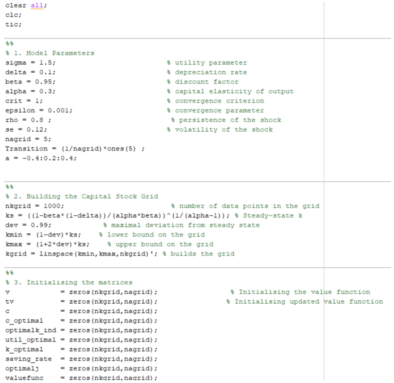
- The value function can be written as

$$V(K_{t-1}, A_t) = \max_{C_t} \left[\frac{C_t^{1-\sigma}}{1-\sigma} + \beta E_t \left[V\left(A_t K_{t-1}^{\alpha} - C_t + (1-\delta) K_{t-1}, A_{t+1}\right) \mid A_t \right] \right]$$

A Programme to Solve a Stochastic RBC Model

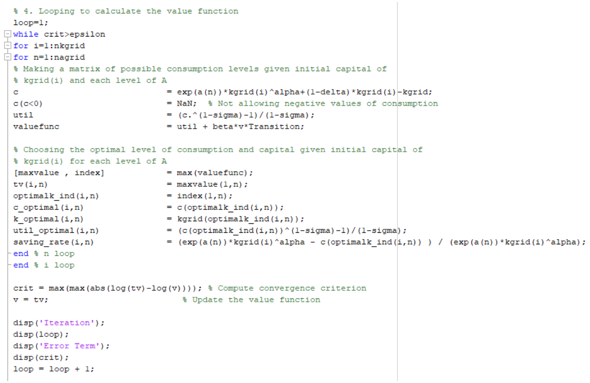
- We need to calculate
$$E_t\left[V\left(K_t, A_{t+1}\right) \mid A_t\right]$$
for each value of A_t and for each feasible value of K_t.
- As with our finite-horizon stochastic dynamic programming code, we set up matrices with value function entries for each possible value of A_t in the columns and for all feasible values of K_t in the rows.
- Again, we multiply this matrix by the transpose of the transition matrix to calculate expected values of the value function for each state and each starting level of capital.
- In the programme illustrated on the following pages, $\log A_t$ follows a 5-state Markov process but all the transition probabilities have been set equal to 0.2, so each of the states are equally likely each period.

Stochastic RBC Model: Preliminaries



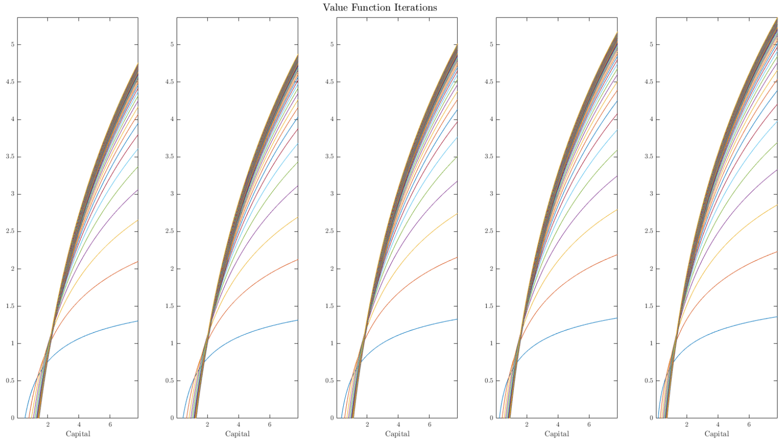
```
clear all;
clc;
tic;

%%
% 1. Model Parameters
sigma = 1.5;                          % utility parameter
delta = 0.1;                          % depreciation rate
beta = 0.95;                          % discount factor
alpha = 0.3;                          % capital elasticity of output
crit = 1;                             % convergence criterion
epsilon = 0.001;                      % convergence parameter
rho = 0.8 ;                            % persistence of the shock
se = 0.12;                            % volatility of the shock
nagrid = 5;
Transition = (1/nagrid)*ones(5) ;
a = -0.4:0.2:0.4;


%%
% 2. Building the Capital Stock Grid
nkgrid = 1000;                         % number of data points in the grid
ks = ((1-beta*(1-delta))/(alpha*beta))^(1/(alpha-1)); % Steady-state k
dev = 0.99;              % maximal deviation from steady state
kmin = (1-dev)*ks;    % lower bound on the grid
kmax = (1+2*dev)*ks;    % upper bound on the grid
kgrid = linspace(kmin,kmax,nkgrid)'; % builds the grid

%%
% 3. Initialising the matrices
v            = zeros(nkgrid,nagrid);                % Initialising the value function
tv           = zeros(nkgrid,nagrid);               % Initialising updated value function
c            = zeros(nkgrid,nagrid);
c_optimal    = zeros(nkgrid,nagrid);
optimalk_ind = zeros(nkgrid,nagrid);
util_optimal = zeros(nkgrid,nagrid);
k_optimal    = zeros(nkgrid,nagrid);
saving_rate  = zeros(nkgrid,nagrid);
optimalj     = zeros(nkgrid,nagrid);
valuefunc    = zeros(nkgrid,nagrid);
```

Stochastic RBC Model: Main Code

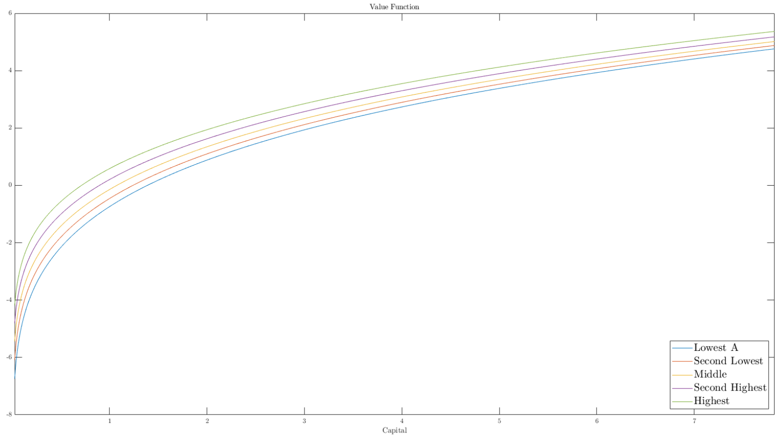


```
% 4. Looping to calculate the value function
loop=1;
while crit>epsilon
for i=1:nkgrid
for n=1:nagrid
% Making a matrix of possible consumption levels given initial capital of
% kgrid(i) and each level of A
c                              = exp(a(n))*kgrid(i)^alpha+(1-delta)*kgrid(i)-kgrid;
c(c<0)                         = NaN;  % Not allowing negative values of consumption
util                           = (c.^(1-sigma)-1)/(1-sigma);
valuefunc                      = util + beta*v*Transition;

% Choosing the optimal level of consumption and capital given initial capital of
% kgrid(i) for each level of A
[maxvalue , index]             = max(valuefunc);
tv(i,n)                        = maxvalue(1,n);
optimalk_ind(i,n)              = index(1,n);
c_optimal(i,n)                 = c(optimalk_ind(i,n));
k_optimal(i,n)                 = kgrid(optimalk_ind(i,n));
util_optimal(i,n)              = (c(optimalk_ind(i,n))^(1-sigma)-1)/(1-sigma);
saving_rate(i,n)               = (exp(a(n))*kgrid(i)^alpha - c(optimalk_ind(i,n)) ) / (exp(a(n))*kgrid(i)^alpha);
end % n loop
end % i loop

crit = max(max(abs(log(tv)-log(v)))); % Compute convergence criterion
v = tv;                                % Update the value function

disp('Iteration');
disp(loop);
disp('Error Term');
disp(crit);
loop = loop + 1;
```

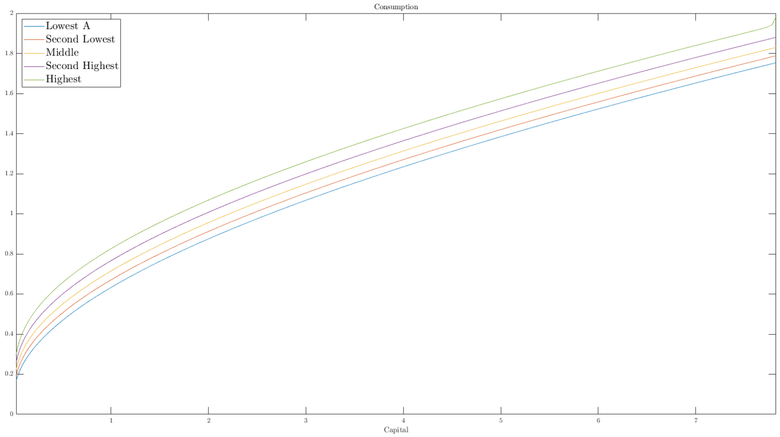
Five Sets of Converging Value Functions for a 5-State Markov Process Problem



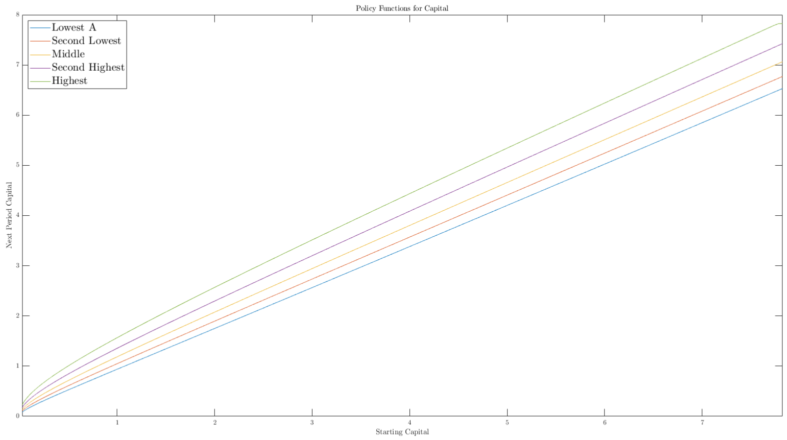
Value Function Solutions for a 5-State Markov Process Problem



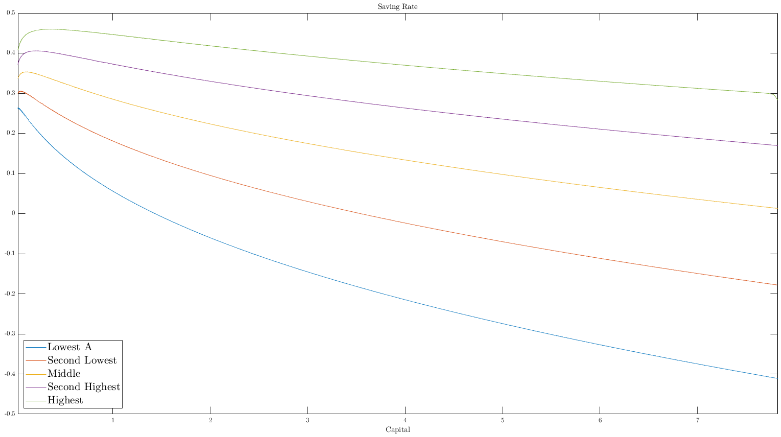
Policy Functions for Consumption for a 5-State Markov Process Problem



Policy Functions for Capital for a 5-State Markov Process Problem



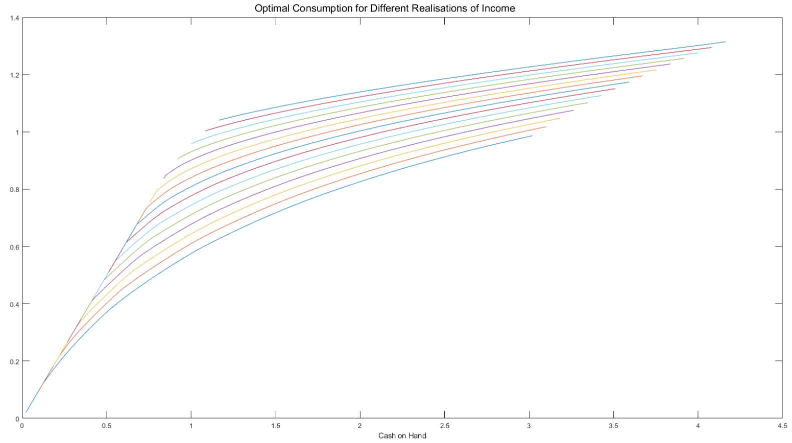
Policy Functions for the Savings Rate for a 5-State Markov Process Problem



Back to Exogenous Income

- Let's go back to the example from the last lecture where income is exogenously determined by a Markov chain but in this case the household lives forever.
- The graph on the next page shows the consumption spending rules as a function of cash-on-hand that comes out of a model with a 50-state Markov chain income process, using the Rouwenhurst method to approximate an AR(1) process.
- This replicates the earlier finding in a finite-horizon model of high MPCs at low levels of cash-on-hand before quickly flattening out.
- This chart more or less replicates a graph from a famous 1991 paper "Saving and Liquidity Constraints" by Angus Deaton which was the first paper to solve an infinite-horizon consumption problem with liquidity constraints. You can see Deaton's graph two pages down.

Consumption and Cash on Hand with a Many-State Markov Process



Angus Deaton's 1991 Graph

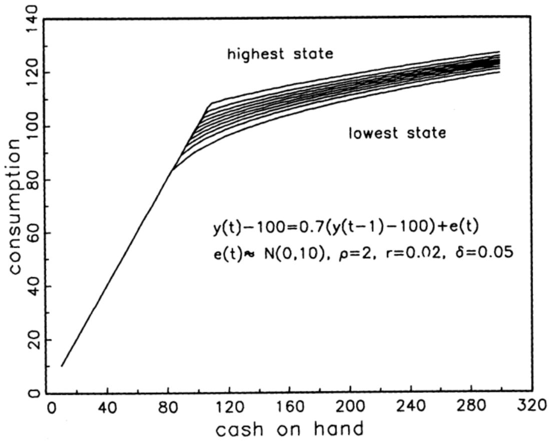


FIGURE 3.—Consumption and cash on hand for AR(1) income process.

Part II

Job Search and Reservation Wages

Accepting or Rejecting a Job Offer

- The following model due to McCall (1970) is a classic illustration of infinite-horizon stochastic dynamic programming.
- An unemployed worker is searching for a job. Once they accept a job, they will have it forever. They are aware there is a distribution of possible wages they can be offered with cumulative distribution function $F(w)$ and probability density function $f(w)$. If they don't accept the job, they get unemployment compensation of b and get another random offer next period. Workers want to maximise the present discounted value (PDV) of their earnings using a discount rate β.
- The optimal policy here takes the form a deciding on a **reservation wage**, R, such that the worker accepts all job offers with wages higher than or equal to R and rejects offers with lower wages.
- This implies a worker who receives a job offer of w formulates a Bellman equation of the form

$$V(w) = \max \left[\frac{w}{1-\beta}, b + \beta \int V(\omega) f(\omega) \, d\omega \right]$$

- Let's explain where this equation comes from.

Explaining the Bellman Equation

- How did we get the Bellman equation?

$$V(w) = \max\left[\frac{w}{1-\beta}, b + \beta \int V(\omega) f(\omega)\, d\omega\right]$$

- The first part $\frac{w}{1-\beta}$ comes from accepting the job offer. Their PDV of wage earnings is going to be

$$w + \beta w + \beta^2 w + ... = \frac{w}{1-\beta}$$

- The second part describes the PDV of earnings when they reject the job. They will get unemployment compensation of b and then their PDV for the rest of time will depend on the uncertain wage offer they get next period.
- The reservation wage is defined to be the wage that makes these two elements of the Bellman equation be equal so the worker is indifferent between accepting or rejecting the offer.

$$\frac{R}{1-\beta} = b + \beta \int V(\omega) f(\omega)\, d\omega$$

A More Intuitive Form for the Reservation Wage

- The worker will refuse all future wage offers that are below R. That means for $w \leq R$

$$V(w) = b + \beta \int V(\omega) f(\omega) \, d\omega = \frac{R}{1-\beta}$$

and they will accept all offers higher than R, so for $w \geq R$

$$V(w) = \frac{w}{1-\beta}$$

- That means we can re-write the reservation wage formula as

$$\frac{R}{1-\beta} = b + \frac{\beta R F(R)}{1-\beta} + \frac{\beta}{1-\beta} \int_{\omega > R} \omega f(\omega) \, d\omega$$

- Adding and subtracting $\frac{\beta R(1-F(R))}{1-\beta}$ on the right-hand side, we get

$$\frac{R}{1-\beta} = b + \frac{\beta R}{1-\beta} + \frac{\beta}{1-\beta} \int_{\omega > R} (\omega - R) f(\omega) \, d\omega$$

A More Intuitive Form for the Reservation Wage

- Re-arranging the last equation we get

$$R - b = \frac{\beta}{1-\beta} \int_{\omega > R} (\omega - R) f(\omega) \, d\omega$$

- The left-hand side is the cost of foregoing the wage of R.
- The right-hand side is the expected benefit of one more search.
- At the reservation wage, these two are equal.
- Note that while this equation is neat, it does not actually give us a simple formula for the reservation wage. The reservation wage appears on both the left hand side and the right hand side, because the integral starts at R.
- One way to figure it out is to guess an initial form for the value function. $V(w) = \frac{w}{1-\beta}$ will be correct for the upper range of wages, so it's a good one to start with. Then iterate on

$$V^n(w) = \max \left[\frac{w}{1-\beta}, b + \beta \int V^{n-1}(\omega) f(\omega) \, d\omega \right]$$

until the V^n functions converge.

Implementing with Matlab

- How do we get Matlab to implement this equation?

$$V^n(w) = \max\left[\frac{w}{1-\beta}, b + \beta \int V^{n-1}(\omega) f(\omega)\, d\omega\right]$$

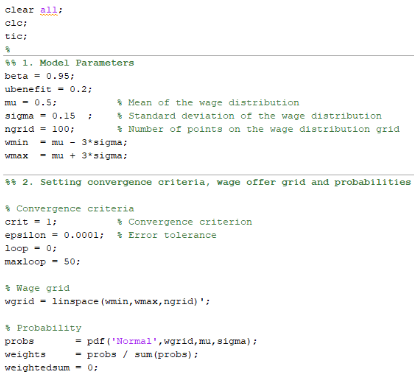
- This contains an integral, so technically you need a closed-form solution that tells you the value of $V^{n-1}(\omega)$ at all possible values of ω. Numerical solution methods will not give you this.
- Instead, what we will do is pick a set of K grid points for different possible wages, w_1, w_2, ..., w_K. Then we will use the probability distribution function values for each of these points p_1, p_2, ..., p_K, sum them up and define a set of relative probability weights

$$\mu_n = \frac{p_n}{\sum_{i=1}^{K} p_i}$$

- We then calculate the value function at each point on the grid and approximate the integrals as

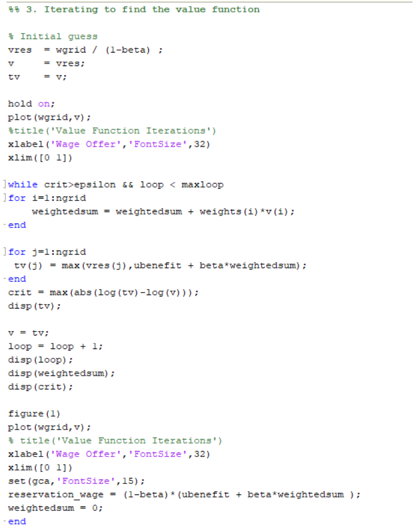
$$V^n(w_n) = \max\left[\frac{w_n}{1-\beta}, b + \beta \sum_{i=1}^{K} \mu_i V^{n-1}(w_i)\right]$$

Reservation Wage Code: Preliminaries



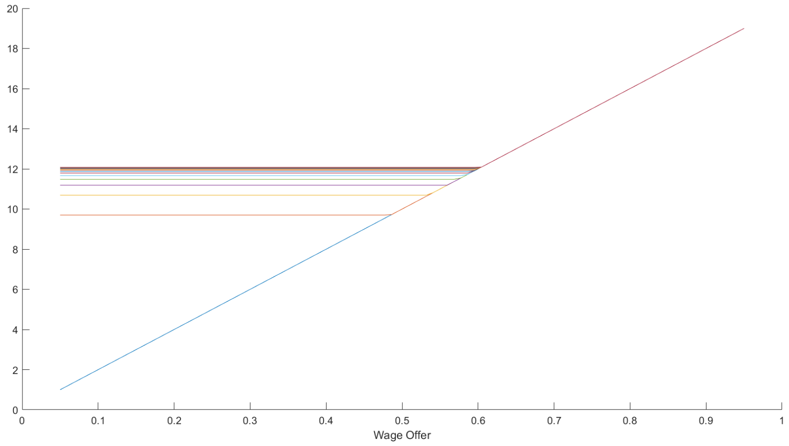
```
clear all;
clc;
tic;
%
%% 1. Model Parameters
beta = 0.95;
ubenefit = 0.2;
mu = 0.5;          % Mean of the wage distribution
sigma = 0.15  ;    % Standard deviation of the wage distribution
ngrid = 100;       % Number of points on the wage distribution grid
wmin  = mu - 3*sigma;
wmax  = mu + 3*sigma;

%% 2. Setting convergence criteria, wage offer grid and probabilities

% Convergence criteria
crit = 1;          % Convergence criterion
epsilon = 0.0001;  % Error tolerance
loop = 0;
maxloop = 50;

% Wage grid
wgrid = linspace(wmin,wmax,ngrid)';

% Probability
probs       = pdf('Normal',wgrid,mu,sigma);
weights     = probs / sum(probs);
weightedsum = 0;
```

Reservation Wage Code: Main Programme

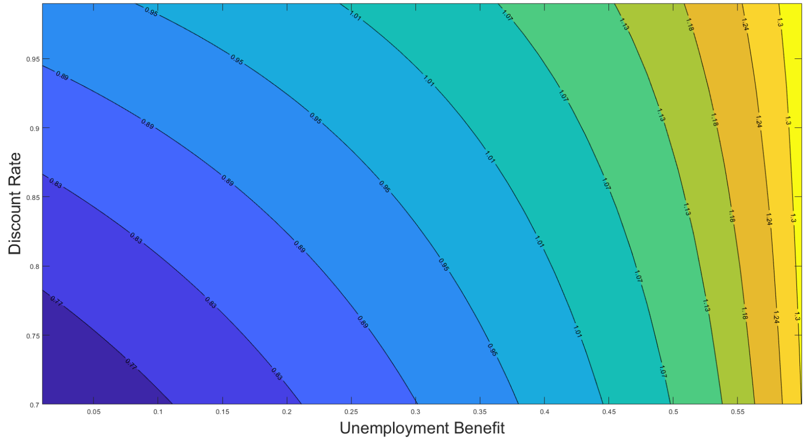


```
%% 3. Iterating to find the value function

% Initial guess
vres  = wgrid / (1-beta) ;
v     = vres;
tv    = v;

hold on;
plot(wgrid,v);
%title('Value Function Iterations')
xlabel('Wage Offer','FontSize',32)
xlim([0 1])

while crit>epsilon && loop < maxloop
for i=1:ngrid
    weightedsum = weightedsum + weights(i)*v(i);
end

for j=1:ngrid
 tv(j) = max(vres(j),ubenefit + beta*weightedsum);
end
crit = max(abs(log(tv)-log(v)));
disp(tv);

v = tv;
loop = loop + 1;
disp(loop);
disp(weightedsum);
disp(crit);

figure(1)
plot(wgrid,v);
% title('Value Function Iterations')
xlabel('Wage Offer','FontSize',32)
xlim([0 1])
set(gca,'FontSize',15);
reservation_wage = (1-beta)*(ubenefit + beta*weightedsum );
weightedsum = 0;
end
```

Iterating to Discover the Value Function



How Reservation Wages Change with β and Unemployment Compensation



Limitations of the Reservation Wage Framework

- The McCall model provides a nice illustration of stochastic dynamic programming and is perhaps a good model of an individual worker deciding on whether to accept a job offer.
- But once you go beyond that to apply it to the labour market as a whole some questions come up. If workers have a reservation wage R, why is there a wage distribution in which lots of employers unnecessarily offer wages higher than R?
- Even if almost all firms decided to offer R, it might be worth accepting an offer just slightly less than R to avoid being unemployed for another period. The logic of this is all firms should just offer the unemployment benefit rate as their wage offer.
- So search and matching theories of the labour market have moved beyond simple reservation wage models to focus more on how wages are determined via bargaining and why there is a nondegenerate distribution of potential wage offers for workers.
- The notes by Daron Acemoglu provided in Brightspace are a comprehensive treatment of this literature.