Homework 3

ECON 8050: Macroeconomics II
Tate Mason

Problem 1: Dynamic Programming

Consider the following model with a disability shock. There are three sources of uncertainty:

- Out-of-pocket medical shock evolving according to transition matrix $\Psi(x_t|x_{t-1})$.
- Productivity evolving according to $T(z_t|z_{t-1})$.
- Disability shock.

The timing of events is as follows: At the beginning of the period, an individual with savings k_t learns their productivity z_t and medical shock x_t. Then they decide whether to work ($l_t = 0$ or $l_t > 0$). If working, labor income is wz_tl_t. Then, they decide about consumption c_t and savings k_{t+1}.

At the end of the period, the disability shock is realized with probability d. Disabled individuals stay permanently disabled, do not work, receive constant benefits DI, and make only consumption/savings decisions. Medical spending for disabled individuals is fully covered by public insurance.

(1) Write down the dynamic programming problem of a non-disabled individual, denoting the value function as V_t.

(2) Write down the dynamic programming problem of a disabled individual, denoting the value function as V_t^d.

(3) Modify the problem assuming disabled individuals can recover with probability f. Recovered individuals draw new productivity realizations from the invariant distribution.

(4) Extend the model to allow non-disabled individuals to falsely claim disability benefits, introducing the value function for falsely disabled V_t^{fd}.

Problem 2: Consumption-Savings Model

A consumer with infinite life maximizes quadratic utility:

$$u(c_t) = -\frac{1}{2}(c_t - \bar{c})^2$$

where future utility is discounted at rate β and borrowing/savings occur at interest rate r with $\beta(1 + r) = 1$.

The consumer's endowment y_t is i.i.d. with values y_H and y_L occurring with probabilities p_H and p_L respectively. The budget constraint is:

$$c_t = a_t(1 + r) + y_t - a_{t+1}.$$

(1) Solve for the consumption and saving functions. Provide intuition on when savings are positive or negative.

(2) Introduce a borrowing constraint $a_{t+1} \geq 0$. Solve the consumer's problem in recursive form numerically using given parameters.

(3) Plot policy functions a_{t+1} and c_t as functions of current assets a_t for cases with and without borrowing constraints.

(4) Simulate the income process and optimal decision rules over $T = 100$ periods. Compare results with and without borrowing constraints.

Solution 1

Part (i)

The Bellman for an able bodied person with probability of becoming disabled d is as follows:

$$V_t(k_t, x_t, z_t) = \max_{c_t, l_t, k_{t+1}} \{u(c_t, l_t) + \beta(1-d) \sum_{x_t} \sum_{z_t} \Psi(x_t|x_{t-1}) \mathrm{T}(z_t|z_{t-1}) V_{t+1}(k_{t+1}, x_{t+1}, z_{t+1})$$
$$+\beta d \sum_{x_t} \Psi(x_t|x_{t-1}) V_{t+1}(k_{t+1}, x_{t+1}, z_{t+1})\}$$

s.t.

$$c_t + k_{t+1} + x_t = wz_t l_t + k_t(1+r)$$

Part (ii)

The Bellman for an individual who is disabled and has no probability of recovery can be represented as follows:

$$V_t^d(k_t, x_t) = \max_{c_t, k_{t+1}} \{u(c_t, 0) + \beta \sum_{x_t} \Psi(x_t|x_{t-1}) V_{t+1}^d(k_{t+1}, x_{t+1})\}$$

s.t.

$$c_t + k_{t+1} = DI + k_t(1+r)$$

Part (iii)

The Bellman equation for an individual who is disabled but has a probability of recovery is as follows:

$$V_t^{df}(k_t, x_t) = \max_{c_t, k_{t+1}} \{u(c_t, l_t) + \beta f \sum_{x_t|x_{t-1}} \sum_{z_t|z_{t-1}} \Psi(x_t|x_{t-1}) \mathrm{T}(z_t|z_{t-1}) V_{t+1}^d(k_{t+1}, x_{t+1}, z_{t+1})$$
$$+\beta(1-f) \sum_{x_t} \Psi(x_t|x_{t-1}) V_{t+1}^d(k_{t+1}, x_{t+1})\}$$

s.t.

$$c_t + k_{t+1} + x_t = DI + k_t(1+r)$$

Part (iv)

Finally, the Bellman for someone who has the option to fake disability is as follows:

$$V_t^{df}(k_t, x_t, z_t) = \max_{c_t, k_{t+1}, l_t} \{u(c_t, l_t) + \beta f \sum_{x_t|x_{t-1}} \sum_{z_t|z_{t-1}} \Psi(x_t|x_{t-1}) \mathrm{T}(z_t|z_{t-1}) V_{t+1}^d(k_{t+1}, x_{t+1}, z_{t+1})$$
$$+\beta(1-f) \sum_{x_t} \Psi(x_t|x_{t-1}) V_{t+1}^d(k_{t+1}, x_{t+1})$$
$$+\beta f \mathbb{1}_{fake=1} \sum_{x_t} (x_t|x_{t-1}) V_{t+1}^{fd}(k_{t+1}, x_{t+1})\}$$

s.t.

$$c_t + k_{t+1} + x_t = wz_tl_t + DI\mathbb{1}_{D=1 or fake=1} + k_t(1+r)$$

Solution 2:

Part (i)

$$\sum_{t=0}^{\infty} \beta^t [-\frac{1}{2}(c_t - \bar{c})^2]$$

s.t.

$$c_t + a_{t+1} = (1+r)a_t + y_t$$

Because the utility is quadratic and $\beta(1+r) = 1$, we know the Euler is as follows:

$$c_t = \mathbb{E}_t[c_{t+1}]$$
$$\therefore \; u'(c_t) = \mathbb{E}_t[u'(c_{t+1})]$$

Plugging in the utility function given above,

$$(c_t - \bar{c}) = \mathbb{E}_t[(c_{t+1} - \bar{c})]$$

Assuming $\bar{c}$ is constant,

$$c_t = \mathbb{E}_t[c_{t+1}] \rightarrow c_t = c_{t+1} = c$$

Plugging this back into the budget constraint will allow us to solve for the intertemporal budget constraint

$$\sum_{t=0}^{\infty} \frac{c}{(1+r)^t} = \sum_{t=0}^{\infty} \frac{y_t}{(1+r)^t} + a_0$$

As given in the question, y_t is i.i.d and thus we can state the following

$$\mathbb{E}_t[y_t] = p_L y_L + p_H y_H$$

Now, after simplifying terms,

$$c = r(a_0 + \frac{\mathbb{E}_t[y_t]}{r})$$
$$c_t = ra_t + \mathbb{E}_t[y_t]$$

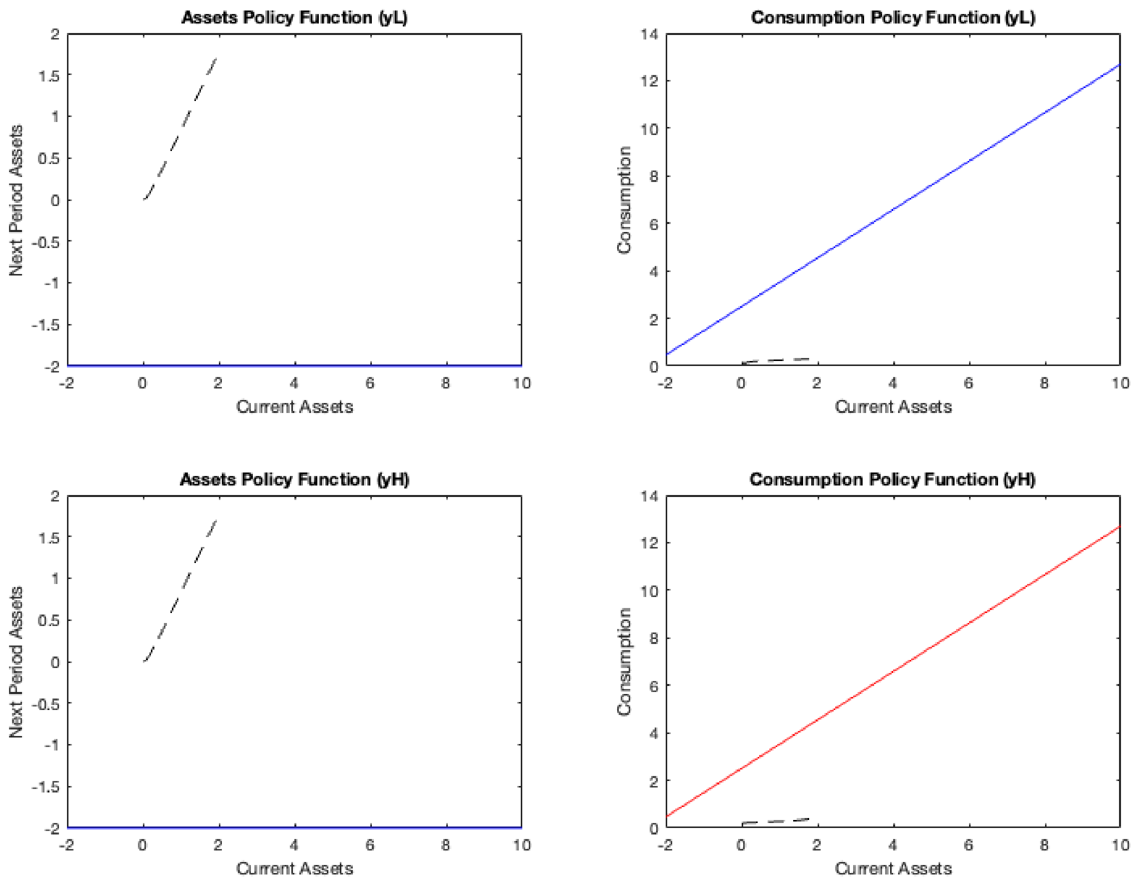
Now, using this, we can find the savings function

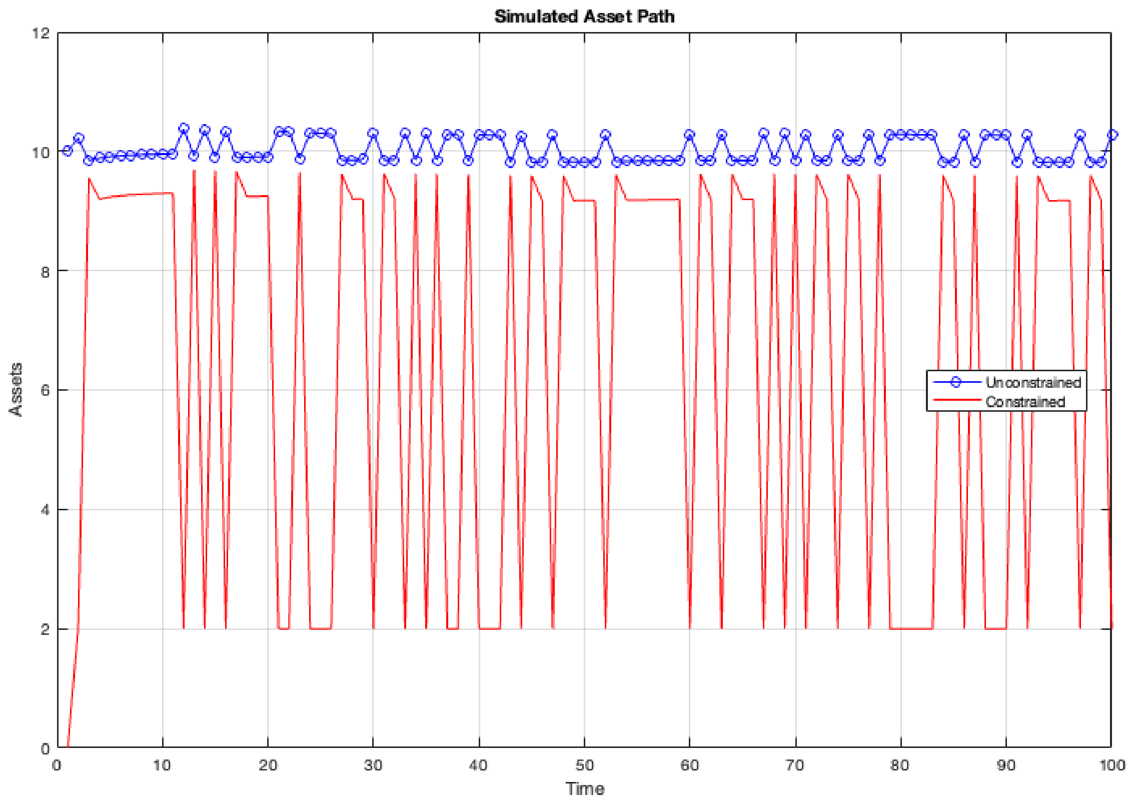
$$
\begin{aligned}
a_{t+1} &= a_t(1+r) + y_t - c_t \\
a_{t+1} &= a_t(1+r) + y_t - ra_t - \mathbb{E}_t[y_t] \\
a_{t+1} &= a_t + y_t - \mathbb{E}_t[y_t]
\end{aligned}
$$

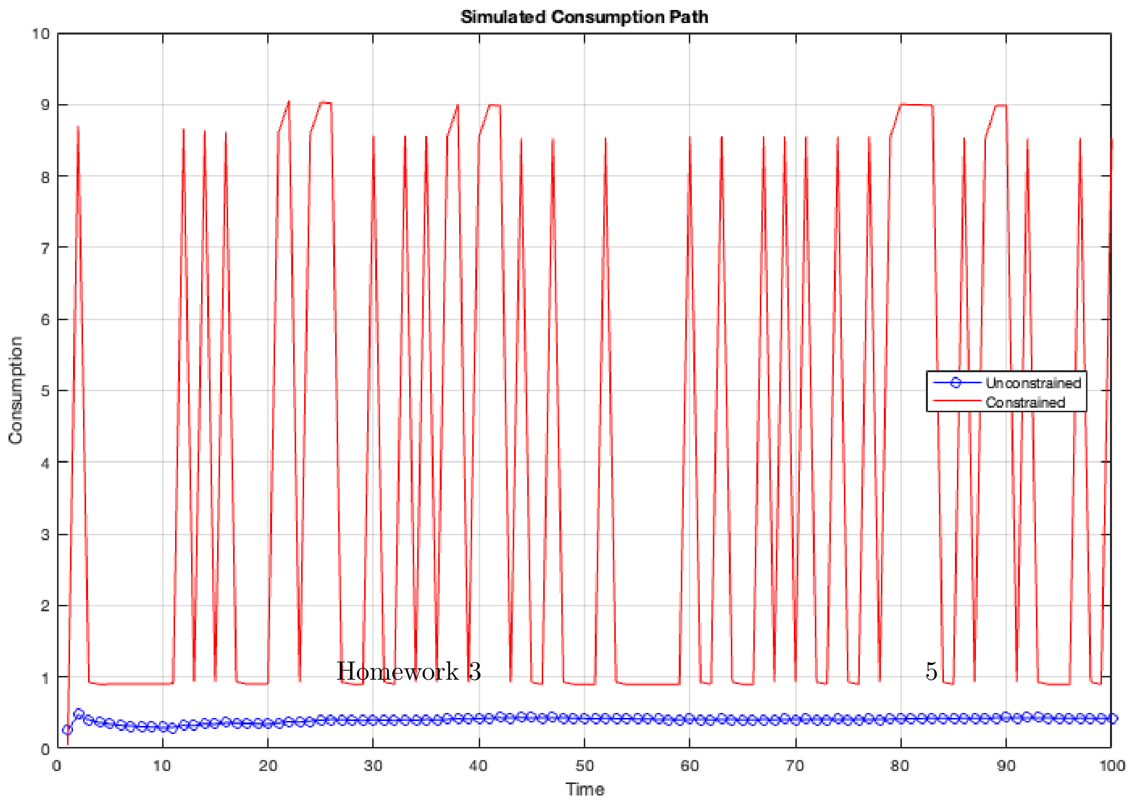
Now, we can see that the agent will consume based on their assets and expected income. This is to be expected and standard. However, their savings function will depend on assets and the difference between actual and expected consumption. If $y_t > \mathbb{E}_t[y_t]$, the agent will save, trying to bolster savings to smooth consumption in periods of lower income. When $y_t < \mathbb{E}_t[y_t]$, the agent will borrow, again attempting to smooth consumption due to their lower income.

Parts (ii-iv)

Outputs from code are as follows:
VFI converged after 1644 iterations, diff = 1.16e-10
Unconstrained VFI converged after 2000 iterations, diff = 1.00e+00
Borrowing constraint was binding in 1.00% of periods.
Here are plots for the associated parts (sorry for poor quality i do not know why this is the case):




Simulated Asset Path
Assets
Time
Unconstrained
Constrained
0
2
4
6
8
10
12
0
10
20
30
40
50
60
70
80
90
100


Simulated Consumption Path
Consumption
Time
Unconstrained
Constrained
0
1
2
3
4
5
6
7
8
9
10
0
10
20
30
40
50
60
70
80
90
100

```
% Solving Consumer's Problem %
clear;
clc;

% Definition of Parameters %
r = 0.02;
beta = 1/(1+r);
cbar = 100;
yL = 0.05;
yH = 0.5;
pL = 0.6;
pH = 0.4;

% Bringing in Agrid %
r=0.02;
yvec=[0.05,0.5];
min=0.0001;
amax=2;
k=20;
gamma=(1+r)^(1/(k-1));
step1=(yvec(1)-min)*(gamma-1)/r;
n=floor(1+log(amax*(gamma-1)/step1+1)/log(gamma));
f=@(x)step1*(gamma.^(x-1)-1)/(gamma-1);
agrid=f(1:n)

% VFI Params %
tol = 1e-10;
maxiter = 2000;
diff = 999.0;
iter = 0;

% Init Matrices %
n = length(agrid);
v0 = zeros(n,2);
v1 = zeros(n,2);
aopt = zeros(n,2);
copt = zeros(n,2);
jopt = zeros(n,2);

% VFI %
while diff > tol && iter<maxiter
  iter = iter + 1;
  diff = 0;
  for i = 1:n
    ai = agrid(i)
    for m = 1:2
      if m == 1
        ym = yL;
      else
        ym = yH;
      end
      res = (1+r)*ai + ym;
      max_val = -Inf;
      opt_index = 1
      for j = 1:n
```

```
        a_next = agrid(j);
        if a_next > res
          break;
        end
        c = res - a_next;
        u = -0.5*(c-cbar)^2;
        EV = beta*(pL*v0(j,1) + pH*v0(j,2));
        val = u+EV;
        if val > max_val
          max_val = val;
          opt_index = j;
        end
      end
    v1(i,m) = max_val;
    jopt(i,m) = opt_index;
    aopt(i,m) = agrid(opt_index);
    copt(i,m) = res-aopt(i,m);
    end
  end
  diff = max(abs(v1-v0));
  v0 = v1;
end
fprint('VFI converged after %d iterations, diff = %2.2e\n', count, diff);

% Unconstrained

% Definition of unconstrained grid
agrid_UC = -2:01:10;
nUC = length(agrid_free);

% initialize params
vUC0 = zeros(nU,2);
vUC1 = zeros(nU,2);
aopt_UC = zeros(nU,2);
copt_UC = zeros(nU,2);
jopt_UC = zeros(nU,2);

% VFI init
tol = 1e-10;
maxiter = 2000;
diff = 1;
iter = 0;

while diff > tol && iter<maxiter
  iter = iter+1;
  diffv = 0;
  for i = 1:nUC
    ai = agrid_UC(i);
    for m = 1:2
      if m == 1
        y_m = yL;
      else
        y_m = yH;
      end
      res = (1+r)*ai + ym;
      max_val = -Inf;
      opt_index = 1
      for j = 1:n
```

```
        a_next = agrid_UC(j);
        if a_next > res
          break;
        end
        cUC = res - a_next;
        u = -0.5*(cUC-cbar)^2;
        EV = beta*(pL*v0(j,1) + pH*v0(j,2));
        val = u+EV;
        if val > max_val
          max_val = val;
          opt_index = j;
        end
      end
    vUC1(i,m) = max_val;
    jopt_UC(i,m) = opt_index;
    aopt_UC(i,m) = agrid_UC(opt_index);
    copt_UC(i,m) = res-aopt_UC(i,m);
    end
  end
  diffUC = max(abs(vUC1-vUC0));
  vUC0 = vUC1;
end
fprint('Unconstrained VFI converged after %d iterations, diff = %2.2e\n',
     count, diff);

figure;
subplot(2,2,1);
plot(agrid_UC, aopt_UC(:,1), 'b-', agrid, aopt(:,1), 'k--');
title('Assets Policy Function (yL)');
xlabel('Current Assets');
ylabel('Next Period Assets');

subplot(2,2,2);
plot(agrid_UC, copt_UC(:,1), 'b-', agrid, copt(:,1), 'k--');
title('Consumption Policy Function (yL)');
xlabel('Current Assets');
ylabel('Consumption');

subplot(2,2,3);
plot(agrid_UC, aopt_UC(:,2), 'b-', agrid, aopt(:,1), 'k--');
title('Assets Policy Function (yH)');
xlabel('Current Assets');
ylabel('Next Period Assets');

subplot(2,2,4);
plot(agrid_UC, copt_UC(:,2), 'r-', agrid, copt(:,2), 'k--');
title('Consumption Policy Function (yH)');
xlabel('Current Assets');
ylabel('Consumption');

% Simulations %

%% Unconstrained
T = 100;
a_sim = zeros(T,1);
c_sim = zeros(T,1);

%% Constrained
```

```
a_con = zeros(T+1, 1);
c_con = zeros(T,1)

a_sim(1) = 0

rng(1);

for t = 1:T
  if rand < pL
    y_sim(t) = yL;
    m = 1;
  else
    y_sim(t) = yH;
    m = 2;
  end
  %% UC
  [~, iU] = min(abs(agrid-a_sim(t)));
  a_sim(t+1) = aopt_UC(idx, m);
  c_sim(t) = copt_UC(idx, m);
  %% C
  a_sim(t+1)=interp1(agrid,aopt(:,(y_sim(t)==yH)+1), a_sim(t),
      'linear','extrap');
  c_sim(t)=interp1(agrid, copt(:,(y_sim(t)==yH)+1), a_sim(t), 'linear',
      'extrap');
end

figure('Name', 'Simulation', 'NumberTitle', 'off');
subplot(2,1,1);
plot(1:T, a_un(2:end), 'b-o', 1:T, a_con(2:end), 'r--o', 'LineWidth',1);
xlabel('Time');ylabel('a_{t+1}');
title('Simulated Assets');
grid on;

subplot(2,1,2);
plot(1:T, c_un, 'b-o', 1:T, c_con, 'r--o','LineWidth',1);
xlabel('Time');ylabel('c_t');
title('Simulated Consumption');
grid on;

% Count of how often constraint binds
bind = sum(abs(a_con(2:end))<1e-10);
pctbind = (100*bind)/T;
fprintf('Borrowing constraint was binding in %.2f%% of periods.\n', pctbind);
```