HW6

Ece Teoman

January 28, 2019

Here, I include the code and the output but the m.files and the diary can be found in the directory.

1

Firm's dynamic optimization problem is as follows:

$$v(y,p) = \max_{x\in[0,y]} (px - 0.2x^{3/2}) + \delta\mathbb{E}[v(y',p')]$$

where $y' = y - x$, $p' = p_0 + \rho p + u$ with $u \sim N(0, 0.01)$ and $y_0 \in [0, 100]$.

State variables are y and p, and the policy variable is x. Simplified version of the optimization problem with the parameters given in the question is as follows:

$$v(y,p) = \max_{y'\in[0,y]} (p(y-y') - 0.2(y-y')^{3/2}) + 0.95\mathbb{E}[v(y',p')]$$

where $p' = 0.5 + 0.5p + u$ with $u \sim N(0, 0.01)$ and $y_0 \in [0, 100]$.

2

Grid that approximates the process for p_t is generated using the Tauchen method:

```
function [prob,grid,invdist]=tauchen(Z,mu,ro,sigma)


% N, number of grid points,
% mu, mean of the error term
% ro, AR(1) parameter
% sigma, std. of the error process


mup=mu/(1-ro);
sigmap=sigma/sqrt(1-ro^2);

epsl=mup-3*sigmap;        % this area (betw eps1 and epsl) is going to capture m
epsh=mup+3*sigmap;        % i.e. prob that an observation fall into this area( 3
eps=linspace(epsl,epsh,Z);
w=(eps(2)-eps(1))/2;   % half of the steps between the grids

if Z==1; prob=1;
else

    p=zeros(Z);
    p(:,1)=normcdf(((eps(1)+w-mu)*ones(Z,1)-ro*eps')/sigma);     %prob that futu
    p(:,Z)=ones(Z,1)-normcdf(((eps(Z)-w-mu)*ones(Z,1)-ro*eps')/sigma);
%the last column of the transition matrix
        for j=2:(Z-1)
            p(:,j)=normcdf(((eps(j)+w-mu)*ones(Z,1)-ro*eps')/sigma)-normcdf(((e
        end
end
```

```
    prob=p;
    grid=eps;



    Trans= prob';
probst = (1/Z)*ones(Z,1); % initial distribution of states
test = 1;

  while test > 10^(-8);
      probst1 = Trans*probst;
      test=max(abs(probst1-probst));
      probst = probst1;
   end
invdist=probst;



end
```

Inputs of the function are provided in main.m.

3

My code for value function iteration is:

```
function val=valfun(x)

 global v0 y0 p0 delta i j gridp gridy prob

if x<0
    r=-999999999999999999;
```

```
else
    r=p0*x - 0.2*(x^1.5);
end


 upty=y0-x;
  for qq=1:21
        vqq=v0(:,qq)
        apprv(qq)=interp1(gridy', vqq, upty, 'linear')
        vqq=zeros(101,1);
  end


val=r+(delta*prob(j,:)*apprv')
```

Unfortunately, it's not properly working because I couldn't debug it. The issue appears to be about matrix size compatibility when approximating the value function around y' after the first round, but I couldn't figure out how to fix it. Consequently, I cannot include the plot of value.

4-5-6

Similar to the previous part, due to not being able to compute the value and the policy function, I cannot include the plots that the questions request. Also, questions 2-4 couldn't be redone without a working code.