ECON 6130

Problem Set 7

Gabe Sekeres

December 8, 2024

1. **Value Function Iteration**

(a) I wrote the `solve_vf` script, and generated the linearized solution matrices. They are:

$$h_x = \begin{bmatrix} 0.9500 & 0 & 0 \\ 28.3163 & 0.9643 & 1.2531 \\ 0.4324 & 0.0007 & 0.8962 \end{bmatrix}$$

$$g_x = \begin{bmatrix} 7.3068 & 0.0413 & 1.4861 \\ 12.4116 & 0.0360 & 0.2914 \\ 28.3163 & -0.0057 & 1.2531 \\ 0.4324 & 0.0007 & 0.8962 \\ 13.1577 & 0.0220 & -0.1167 \\ 0.9537 & 0.0012 & 0.0167 \end{bmatrix}$$

(b) I used `AR1_rouwen` to create the grid for log productivity. I got the transition matrix

$$\theta = \begin{bmatrix} 0.9037 & 0.0927 & 0.0036 & 0.0001 & 0.0000 \\ 0.0232 & 0.9055 & 0.0696 & 0.0018 & 0.0000 \\ 0.0006 & 0.0464 & 0.9061 & 0.0464 & 0.0006 \\ 0.0000 & 0.0018 & 0.0696 & 0.9055 & 0.0232 \\ 0.0000 & 0.0001 & 0.0036 & 0.0927 & 0.9037 \end{bmatrix}$$

and the stationary distribution

$$\bar{\theta} = \begin{bmatrix} 0.0625 & 0.2500 & 0.3750 & 0.2500 & 0.0625 \end{bmatrix}$$

Evaluating the inner product of the transition matrix and the grid around log productivity, I found that $\mathbb{E}[A_t] = 1.0005$. This number is greater than 1 because, though exponentiation and logarithmic transformations are monotonic, they are not linear. That means that the long-run expectation will be slightly greater than 1, as the exponential transformation is convex.

(c) I did this

(d) Did this too

(e) Also this!

(f) I estimated the expected value, and got that

$$\mathbb{E}[V(K_{t+1}, N_t, A_{t+1} \mid A_t)] = -3.654721293050265$$

This is numerically equivalent to Ryan's answer, and I got that the first X is 4, while the second is 3.

(g) I did this part!

(h) Please trust me I did this

(i) I performed the convergence process, using `nfix` $= 1$ for precision, and got convergence in 377 iterations, taking 4,512.69 seconds. I attained a value of

$$V(K_t, N_{t-1}, A_t) = -3.682637761479028$$

Remark. I know that running this with nfix at 25 would make this a lot faster, and Finn got the exact same answer as me in literally $1/25$ of the time. However, when I changed it to 25, I got no convergence in 1,500 iterations. I'm not sure what's happening here, but the output can be replicated by directly substituting 25 in for nfix in my code below.

(j) I plotted the policy functions for capital and labor, and they are displayed in Figure 1. As we can see, the true (non-linear) policy function for capital matches the naive linearized model very closely. However, the labor policy function is clearly a lot more nonlinear, and looks very different.

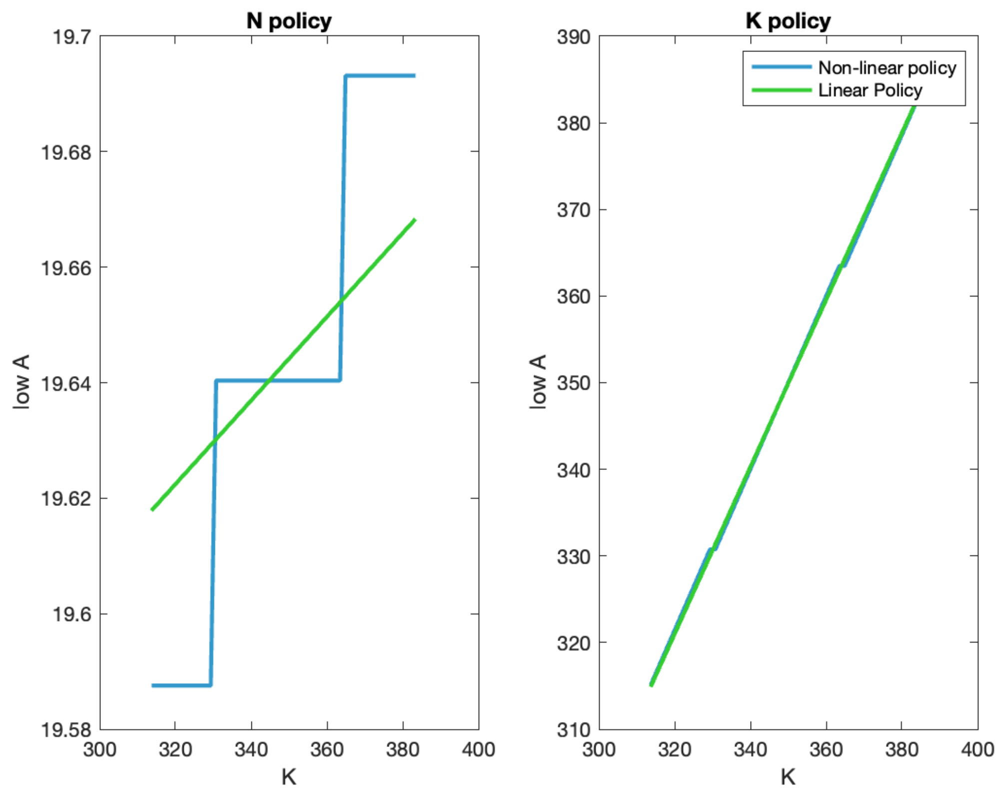

Figure 1: Linear Policy Function and Value Function Policy Function

(k) I simulated the economy for 5,000 periods, and generated the below table from the standard deviations:

Moment	**Linear Model Value**	**Value Function Model Value**
std $\log(Y)$	0.0522	0.0235
std $\log(C)$	0.0391	0.0329
std $\log(I)$	0.0975	0.0958
std $\log(N)$	0.0119	0.0110

2. **Matlab Code:** The code is in three functions. I have `pset7_parameters.m`, `pset7_linear_model.m`, and `pset7_solve_vf.m`. They are:

- `pset7_solve_vf.m`:

```
clear;
format long;
tic;

% Add helper functions
addpath('/Users/gabesekeres/Dropbox/Notes/Cornell_Notes/Fall_2024/
   Macro/Matlab/pset7_helper_functions')

% Load parameters
param = pset7_parameters;

bet = param.bet;
sig = param.sig;
alpha = param.alpha;
deltak = param.deltak;
deltan = param.deltan;
phin = param.phin;
chi = param.chi;
eps = param.eps;
rho = param.rho;
siga = param.siga;

% Solve the linear model
pset7_linear_model;
rehash;
pval  = struct2array(param);

[fy,fx,fyp,fxp,ftest,yxss] = pset7_model_df(pval');
[gx,hx] = gx_hx_alt(fy,fx,fyp,fxp);
eta = [0 ; 1];
disp(gx);
disp(hx);
disp(yxss);

%put parameter value in memory
passign(param);

%steady-state stuff
abar = yxss(a_idx);
kbar = yxss(k_idx);
cbar = yxss(c_idx);
nbar = yxss(n_idx);
vbar = yxss(val_idx);

%Agrid - in logs
na = 5;
[agrid, theta, theta_bar] = AR1_rouwen(na,rho,0,siga);
agrid = exp(agrid);
```

```
disp("Markov transition matrix:");
disp(theta);

disp("Stationary distribution:");
disp(theta_bar);

% Compute the expected value of A_t
E_At = theta_bar *agrid';
disp(['Expected value of A_t: ', num2str(E_At)]);

%Kgrid - in levels
nk    = 50;
kgrid = linspace(.9*kbar,1.1*kbar,nk);

%Hgrid - in levels
nn    = 150;
ngrid = linspace(.8*nbar,1.2*nbar,nn);

%A/K/N combos as initial states
[aagr,kkgr,nngr] = ndgrid(agrid,kgrid,ngrid);
aagr = aagr(:)' ;
kkgr = kkgr(:)';
nngr = nngr(:)';

%K/N combos to choose from
[kkgr2,nngr2] = ndgrid(kgrid,ngrid);
kkgr2 = kkgr2(:)';
nngr2 = nngr2(:)';

%Initial policy functions for K(t+1), N(t)
kinit = kbar + hx(2,:)*[aagr-abar;kkgr-kbar;nngr-nbar];
kinit = reshape(kinit,[na,nk*nn]);

ninit = nbar + gx(n_idx,:)*[aagr-abar;kkgr-kbar;nngr-nbar];
ninit = reshape(ninit,[na,nk*nn]);

vinit = vbar + gx(val_idx,:)*[aagr-abar;kkgr-kbar;nngr-nbar];
vinit = reshape(vinit,[na,nk*nn]);


disp("Initial value function:");
EV_init = theta * vinit;
disp(EV_init(1,1));

% Optimize the value function
idx = zeros(na, nk, nn); crit = 1; jj = 0;

nfix = 1;
```

```
while (crit > 1e-6) && (jj < 1000)
    vinit_old = vinit;
    EVp = theta * vinit;
    vinit = reshape(vinit, na, nk, nn);

    if mod(jj,nfix) == 0
        for aa = 1:na
            for kk = 1:nk
                for nm = 1:nn
                    % State
                    at = agrid(aa);
                    kt = kgrid(kk);
                    nt = ngrid(nm);

                    % Constraints
                    Yt = at .* kt.^alpha .* nngr2 .^ (1 - alpha);
                    vt = ((nngr2 - (1 - deltan) * nt) / chi) .^ (1 /
                        eps);
                    it = kkgr2 - (1 - deltak) * kt;
                    ct = Yt - it - phin * vt;

                    % Compute value function
                    vv = -inf + ones(1,size(EVp,2));
                    idxp = ct>0;
                    vv(idxp) = (ct(idxp) .^ (1 - sig)) / (1 - sig) +
                        bet*EVp(aa,idxp);

                    % Update value function
                    [vinit(aa,kk,nm),idx_tmp] = max(vv);
                    idx(aa,kk,nm) = idx_tmp;
                end
            end
        end
        vinit = reshape(vinit, na, nk*nn);
    else
        evp_k = zeros(na,nk,nn);
        for aa = 1:na
            for kk = 1:nk
                for nm = 1:nn
                    evp_k(aa,kk,nm) = EVp(aa,idx(aa,kk,nm));
                end
            end
        end
        evp_k = reshape(evp_k, na, nk*nn);

        % Constraints
        Yt = aagr .* kkgr.^alpha .* nngr(idx(:)).^(1 - alpha);
        it = kkgr2(idx(:)) - (1 - deltak) * kkgr;
        vt = ((nngr2(idx(:)) - (1 - deltan) * nngr) / chi) .^ (1 / eps
            );
        ct = Yt - it - phin * vt;
```

```
        % Update value function
        vv = (ct.^(1 - sig)) / (1 - sig) + bet*evp_k(:)';

        vinit = reshape(vv, [na,nk*nn]);
    end

    crit = max(max(abs(vinit - vinit_old)));
    vinit_old = vinit;
    disp(['Iteration: ', num2str(jj), ' Crit: ', num2str(crit, '%2.2e'
       )]);
    jj = jj + 1;
end

% Final
disp("Final value function:");
exactvinit = vinit;
disp(exactvinit(1,1,1));

% Plot the policy functions
kpol = reshape(kkgr2(idx(:)),na,nk,nn);
npol = reshape(nngr2(idx(:)),na,nk,nn);

kinitpol = reshape(kinit,[na,nk,nn]);
ninitpol = reshape(ninit,[na,nk,nn]);

% Define colors
calm_blue = [0.2, 0.6, 0.8];
calm_green = [0.2, 0.8, 0.2];

figure;
subplot(1,2,1);
plot(kgrid,npol(3,:,75), 'linewidth',2, 'Color', calm_blue); ylabel('
   low A'); xlabel('K'); title('N policy')
hold on
plot(kgrid,ninitpol(3,:,75),'LineWidth',2,'Color',calm_green);

subplot(1,2,2);
plot(kgrid,kpol(3,:,75), 'linewidth',2, 'Color', calm_blue); ylabel('
   low A'); xlabel('K'); title('K policy')
hold on
plot(kgrid,kinitpol(3,:,75),'LineWidth',2,'Color',calm_green);

legend('Non-linear policy','Linear Policy');

saveas(gcf, '/Users/gabesekeres/Dropbox/Notes/Cornell_Notes/Fall_2024/
   Macro/Matlab/pset7_policy_functions.png');


% Simulate over 5000 periods
mc = dtmc(theta);
```

```
x = simulate(mc,5000);
ks = kgrid(25);
ns = ngrid(75);

npol = reshape(npol,na,nk,nn);
kpol = reshape(kpol,na,nk,nn);


vect= zeros(5000,4);
for u  = 1:5000

    nc = npol(x(u),find(kgrid==ks),find(ngrid==ns));
    kc = kpol(x(u),find(kgrid==ks),find(ngrid==ns));

    y = ks^(alpha)*nc^(1-alpha);
    i = kc-(1-deltak)*ks;
    v = ((nc-(1-deltan)*ns)/chi)^(1/eps);
    c = y-i-phin*v;

    vect(u,:) =[y c i nc];

    ns = nc;
    ks = kc;

end

lvect = log(vect);

% Standard deviations value function model

disp("Standard deviations value function model:");
disp(['Y: ', num2str(std(lvect(:,1)))]);
disp(['C: ', num2str(std(lvect(:,2)))]);
disp(['I: ', num2str(std(lvect(:,3)))]);
disp(['N: ', num2str(std(lvect(:,4)))]);




toc;
```

- pset7_parameters.m:

```
function param = pset7_parameters()
    param.bet = 0.99;
    param.sig = 2;
    param.alpha = 0.3;
    param.deltak = 0.03;
    param.deltan = 0.1;
    param.phin = 0.5;
    param.chi = 1;
```

```
    param.eps = 0.25;
    param.rho = 0.95;
    param.siga = 0.01;
end
```

- pset7_linear_model.m:

```
addpath('/Users/gabesekeres/Dropbox/Notes/Cornell_Notes/Fall_2024/
    Macro/Matlab/pset7_helper_functions')

param = pset7_parameters;

%Declare model variables
declare A K N_m;
X = D; XP = make_prime(X);

declare Yt C I N V VAL;
Y = D; YP = make_prime(Y);

ny = length(Y);
nx = length(X);

%Declare model parameters & values
pnames = fieldnames(param);
declare(pnames{:});
pvec = D;
pnum = struct2array(param);

%Model Equations
f = sym([]);
f(end+1) = 1 - bet * (C_p / C)^(-sig) * (A_p * alpha * (K_p / N_p)^(
    alpha - 1) + 1 - deltak);
f(end+1) = phin / (eps * chi * V^(eps - 1)) - A * (1 - alpha) * (K / N
    )^alpha - bet * (C_p / C)^(-sig) * (phin / (eps * chi * V_p^(eps -
    1))) * (1 - deltan);
f(end+1) = Yt - A * K^alpha * N^(1 - alpha);
f(end+1) = Yt - C - I - phin * V;
f(end+1) = K_p - (1 - deltak) * K - I;
f(end+1) = N - (1 - deltan) * N_m - chi * V^eps;
f(end+1) = log(A_p) - rho * log(A);
f(end+1) = N_m_p - N;
f(end+1) = VAL - (C ^ (1 - sig)) / (1 - sig) - bet*VAL_p;

disp(['neq    :' num2str(length(f))])
disp(['ny + nx:' num2str(ny+nx)])

%Steady state, use closed form expressions for the ss values.
kn = ((1/bet - 1 + deltak) / alpha)^(1 / (alpha - 1));
v = (((eps*chi) / phin) * (1 - alpha) / (1 - bet * (1 - deltan)) * kn
    ^ alpha)^(1 / (1 - eps));
n = chi * v ^ eps / deltan;
k = kn * n;
```

```
y = k^alpha * n^(1-alpha);
i = deltak * k;
c = y - i - phin * v;
a = 1;
val = (1 / (1 - bet)) * (c ^ (1 - sig)) / (1 - sig);

%Y and X vectors with SS values
Yss = [y c i n v val];
Xss = [a k n];

%Log-linear approx (Pure linear if log_var = [])
xlog = [];%1:length(X);
ylog = [];%1:length(Y); ylog(end) = []; %V in negative in SS
log_var = [X(xlog) Y(ylog) XP(xlog) YP(ylog)];

Yss(ylog) = log(Yss(ylog));
Xss(xlog) = log(Xss(xlog));

f = subs(f, log_var, exp(log_var));

% Get the derivative matrices
fx  = subs(jacobian(f,X)  ,[YP,XP,Y,X],[Yss,Xss,Yss,Xss]);
fy  = subs(jacobian(f,Y)  ,[YP,XP,Y,X],[Yss,Xss,Yss,Xss]);
fxp = subs(jacobian(f,XP) ,[YP,XP,Y,X],[Yss,Xss,Yss,Xss]);
fyp = subs(jacobian(f,YP) ,[YP,XP,Y,X],[Yss,Xss,Yss,Xss]);
fv  = subs(f              ,[YP,XP,Y,X],[Yss,Xss,Yss,Xss]);
matlabFunction(fy,fx,fyp,fxp,fv,[Yss,Xss],'vars',{pvec},'file', '/
   Users/gabesekeres/Dropbox/Notes/Cornell_Notes/Fall_2024/Macro/
   Matlab/pset7_model_df.m', 'optimize', false);


make_index([Y,X]);
```