```
% Solving a stochastic neoclassical growth model with elastic labor supply
% ECON 5160, University of Oslo, Spring 2009
% by Espen Henriksen


clear;   % clears all variables from the memory
close;   % closes all figures
tic      % starts the ”stop watch”

% Parameterize the model
alpha   =  .36;
beta    =  .97;
delta   =  .006;
epsilon =  .022;
kappa   =  .975;
psi     = 1.78;
sigma   = 1;

z  = [-epsilon, epsilon];
Xi = (1/beta - (1 - delta))/alpha;
Pi = [kappa   1-kappa
      1-kappa kappa];

dim = 2;    % Number of values the exogenous state variable might take
hg  = 51;   % Number of grid points for the control variable h
kg  = 101;  % Number of grid points for both the control var and state var k

khratio = ((1/beta - 1 + delta)/alpha)^(1/(alpha-1));
chratio = khratio^alpha - delta*khratio;

hstar = 1/(1 + psi*chratio/((1-alpha)*khratio^alpha));
kstar = khratio*hstar;
cstar = chratio*hstar;

clear khratio chratio;   % if you won’t need ’em delete ’em
```

```
kgrid = linspace(.85*kstar,1.15*kstar,kg)';
hgrid = linspace(.8*hstar,1.2*hstar,hg)';

c = zeros(hg,kg,dim,kg);
u = zeros(hg,kg,dim,kg);

% i is a counter for the control variable h
% j is a counter for the control variable k'
% m is a counter for the exog. state variable z
% n is a counter for the endogenous state variable k

warning off; % disable warning for taking log of zero (just annoying)
for i = 1 : hg
    for j = 1 : kg
        for m = 1 : dim
            for n = 1 : kg
                c(i,j,m,n) = exp(z(m))*(kgrid(n)^alpha)*hgrid(i)^(1-alpha) + (1-delta)*kgrid(n) - kgrid(j);
                if c(i,j,m,n) < 0
                    c(i,j,m,n) = 0;
                end
                u(i,j,m,n) = log(c(i,j,m,n)) + psi*log(1-hgrid(i));
            end
        end
    end
end
warning on;   % turn warnings on again
clear c       % free up memory
clear i j m n % clean up

v = zeros(kg,dim);

convcrit = 1E-11;   % chosen convergence criterion
diff = 1;           % arbitrary initial value greate
iter = 0;           % iterations counter
```

```
while diff > convcrit
    diff = 0;
    for m = 1 : dim
        for n = 1 : kg
            objfn(:,:,m,n) = u(:,:,m,n) + beta*(Pi(m,1)*(v(:,1)*ones(1,hg))'+Pi(m,2)*(v(:,2)*ones(1,hg))');
            Tv(n,m) = max(max(objfn(:,:,m,n)));
        end
    end
    diff = norm(v-Tv);
    v = Tv;
    iter = iter + 1;
end


for m = 1 : dim
    for n = 1 : kg
        objfn(:,:,m,n) = u(:,:,m,n) + beta*(Pi(m,1)*(v(:,1)*ones(1,hg))'+Pi(m,2)*(v(:,2)*ones(1,hg))');
        [tmp1,x1] = max(objfn(:,:,m,n),[],1);
        [tmp2,x2] = max(tmp1,[],2);
        kgridrule(m,n) = x2;
        hgridrule(m,n) = x1(x2);
        kdecrule(m,n)  = kgrid(kgridrule(m,n));
        hdecrule(m,n)  = hgrid(hgridrule(m,n));
        cdecrule(m,n)  = exp(z(m))*(kgrid(n)^alpha)*hdecrule(m,n)^(1-alpha) + (1-delta)*kgrid(n) - kdecrule(m,n);
    end
end

% If you won't need 'em -- delete 'em
clear tmp1 tmp2 x1 x2
clear diff convcrit iter m n
clear objfn Tv
clear u

 figure;plot(kgrid,v);
 title('Value␣function')
 figure;
 plot(kgrid,kgrid,kgrid,kdecrule);
 title('Decision␣rules␣for␣capital');
 figure;
 plot(kgrid,cdecrule)
 title('Decision␣rules␣for␣consumption');
 figure;
 plot(kgrid,hdecrule)
 title('Decision␣rules␣for␣labor␣supply');


 keyboard

if mod(kg,2) == 0              % modulus after division,in
    endostate = kg/2;
else
    endostate = (kg-1)/2 + 1;
end
exostate = 1;

kprime = kgrid(endostate);

for ctr = 1 : 10000
    kcurr = kprime;
    draw = rand;
    if exostate == 1
        if draw < Pi(1,1)
            exostate = 1;
        else
            exostate = 2;
        end
    else
        if draw < Pi(2,2)
            exostate = 2;
        else
            exostate = 1;
        end
    end

    kprime = kdecrule(exostate,endostate);
    h(ctr,1) = hdecrule(exostate,endostate);

    k(ctr,1) = kcurr;
    i(ctr,1) = kprime - (1-delta)*kcurr;
    y(ctr,1) = exp(z(exostate))*kcurr^alpha*h(ctr,1)^(1-alpha);
    c(ctr,1) = y(ctr,1) - i(ctr,1);
    r(ctr,1) = exp(z(exostate))*alpha*(kcurr/h(ctr,1))^(alpha-1);
    w(ctr,1) = exp(z(exostate))*(1-alpha)*(kcurr/h(ctr,1))^(alpha);

    endostate = kgridrule(exostate,endostate);
end
clear ctr draw

y = y(1000:10000);
i = i(1000:10000);
h = h(1000:10000);
k = k(1000:10000);
c = c(1000:10000);
r = r(1000:10000);
w = w(1000:10000);

ystd = std(y)/mean(y);
istd = std(i)/mean(i);
hstd = std(h)/mean(h);
kstd = std(k)/mean(k);
cstd = std(c)/mean(c);
rstd = std(r)/mean(r);
wstd = std(w)/mean(w);

ystdy = ystd/ystd;
istdy = istd/ystd;
hstdy = hstd/ystd;
```

```
kstdy = kstd/ystd;
cstdy = cstd/ystd;
rstdy = rstd/ystd;
wstdy = wstd/ystd;

ycorry = corrcoef(y,y);
icorry = corrcoef(y,i);
hcorry = corrcoef(y,h);
kcorry = corrcoef(y,k);
ccorry = corrcoef(y,c);
rcorry = corrcoef(y,r);
wcorry = corrcoef(w,i);

fprintf('␣\n')
fprintf('␣\t␣Mean␣␣␣␣␣␣\t␣St.dev.␣␣␣\t␣Std␣rel␣y␣\t␣Cont.corr␣y␣\n');
fprintf('␣y␣\t␣%1.5f␣␣␣\t␣%1.5f␣␣␣␣␣\t␣%1.5f␣␣␣␣␣\t␣%1.5f␣␣␣␣␣␣␣\n', [mean(y) ystd ystdy ycorry(1,2)])
fprintf('␣c␣\t␣%1.5f␣␣␣\t␣%1.5f␣␣␣␣␣\t␣%1.5f␣␣␣␣␣\t␣%1.5f␣␣␣␣␣␣␣\n', [mean(c) cstd cstdy ccorry(1,2)])
fprintf('␣i␣\t␣%1.5f␣␣␣\t␣%1.5f␣␣␣␣␣\t␣%1.5f␣␣␣␣␣\t␣%1.5f␣␣␣␣␣␣␣\n', [mean(i) istd istdy icorry(1,2)])
fprintf('␣h␣\t␣%1.5f␣␣␣\t␣%1.5f␣␣␣␣␣\t␣%1.5f␣␣␣␣␣\t␣%1.5f␣␣␣␣␣␣␣\n', [mean(h) hstd hstdy hcorry(1,2)])
fprintf('␣k␣\t␣%1.5f␣␣␣\t␣%1.5f␣␣␣␣␣\t␣%1.5f␣␣␣␣␣\t␣%1.5f␣␣␣␣␣␣␣\n', [mean(k) kstd kstdy kcorry(1,2)])
fprintf('␣r␣\t␣%1.5f␣␣␣\t␣%1.5f␣␣␣␣␣\t␣%1.5f␣␣␣␣␣\t␣%1.5f␣␣␣␣␣␣␣\n', [mean(r) rstd rstdy rcorry(1,2)])
fprintf('␣w␣\t␣%1.5f␣␣␣\t␣%1.5f␣␣␣␣␣\t␣%1.5f␣␣␣␣␣\t␣%1.5f␣␣␣␣␣␣␣\n', [mean(w) wstd wstdy wcorry(1,2)])
fprintf('␣\n')
fprintf('␣\n')
fprintf('␣\n')



for ctr = 1 : 9
    tmp = corrcoef(y(ctr:8990+ctr,1),y(5:8995,1));  % because matlab rather stupidly returns the correlation matrix
    corrtable(1,ctr) = tmp(2,1);
    tmp = corrcoef(c(ctr:8990+ctr,1),y(5:8995,1));  % because matlab rather stupidly returns the correlation matrix
    corrtable(2,ctr) = tmp(2,1);
    tmp = corrcoef(i(ctr:8990+ctr,1),y(5:8995,1));  % because matlab rather stupidly returns the correlation matrix
    corrtable(3,ctr) = tmp(2,1);
    tmp = corrcoef(h(ctr:8990+ctr,1),y(5:8995,1));  % because matlab rather stupidly returns the correlation matrix
    corrtable(4,ctr) = tmp(2,1);
end
clear tmp ctr

fprintf('␣␣␣␣␣␣␣␣t-4␣␣␣␣␣␣t-3␣␣␣␣␣␣t-2␣␣␣␣␣␣t-1␣␣␣␣␣␣␣t␣␣␣␣␣␣␣t+1␣␣␣␣␣␣t+2␣␣␣␣␣␣t+3␣␣␣␣␣␣t+4␣␣\n')
fprintf('␣y␣␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣␣␣\n', [corrtable(1,:)])
fprintf('␣c␣␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣␣␣\n', [corrtable(2,:)])
fprintf('␣i␣␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣␣␣\n', [corrtable(3,:)])
fprintf('␣h␣␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣%1.4f␣␣␣␣\n', [corrtable(4,:)])

%for i = 1 : 5000
```