```
# Solving the stochastic Optimal Growth model by Policy Function
Interation
# adapted from Fabrice Collard's Matlab code, http://fabcol.free.fr/
# tested in Julia 0.6.1.1
# this code is part of chapter 4, "Dynamic Programming" from the book:
"Introduction to Quantitative Macroeconomics with Julia"
# Academic Press - Elsevier
# for comments, email at: petre(dot)caraiani(at)gmail(dot)com

sigma   = 1.50;                    # utility parameter
delta   = 0.10;                    # depreciation rate
beta    = 0.95;                    # discount factor
alpha   = 0.30;                    # capital elasticity of output
p       = 0.9;
PI      = [p 1-p;1-p p];
se      = 0.2;
ab      = 0;
am      = exp(ab-se);
as      = exp(ab+se);
A       = [am as];
nba     = 2;

nbk     = 1000;                    # number of data points in the
grid
crit    = 1;                       # convergence criterion
iter    = 1;                       # iteration
tol     = 1e-6;                    # convergence parameter

ks      = ((1-beta*(1-delta))/(alpha*beta))^(1/(alpha-1));

kmin    = 0.2;                     # lower bound on the grid
kmax    = 6;                       # upper bound on the grid
devk    = (kmax-kmin)/(nbk-1);     # implied increment
kgrid   = collect(linspace(kmin,kmax,nbk)); # builds the grid
v       = zeros(nbk,nba);          #  value function
u       = zeros(nbk,nba);          # utility function
u_temp  = zeros(nbk,nba);
c_temp  = zeros(nbk,1);
c       = zeros(nbk,1);
Tv      = zeros(nbk,nba);
Ev      = zeros(nbk,nba);

kp0     = repmat(kgrid,1,nba);     # initial guess on k(t+1)
kp      = zeros(nbk,nba)
dr      = zeros(nbk,nba);          # decision rule (will contain
indices)

#Main Loop
while crit>tol
for i=1:nbk;
```

```
   for j=1:nba;
         c_temp   = A[j]*kgrid[i]^alpha+(1-delta)*kgrid[i]-kgrid;
         neg      = find(x -> x <=0.0,c_temp);
         c_temp[neg]    = NaN;
         u_temp[:,j]    = (c_temp.^(1-sigma)-1)/(1-sigma);

         u_temp[neg,j] = -1e30;
         Ev[:,j]      = v*PI[j,:];

    end
        (Tv[i,:],dr[i,:]) = findmax(u_temp+beta* Ev,1);
        dr[i,2]=dr[i,2]-nbk;
end;


#decision rules
for i=1:nbk
   for j=1:nba

   index=trunc(Int, dr[i,j]);
   kp[i,j]= kgrid[index];

   end;
end;

Q  = spzeros(nbk*nba,nbk*nba);

for j=1:nba;

    c = A[j]*kgrid.^alpha+(1-delta)*kgrid-kp[:,j];

#update the value
    u[:,j]= (c.^(1-sigma)-1)/(1-sigma);

    Q0  = spzeros(nbk,nbk);

    for i=1:nbk;
        index=trunc(Int, dr[i,j]);

        Q0[i,index] = 1;
    end;

Q[(j-1)*nbk+1:j*nbk,:]= kron(PI[j,:]',Q0);
end;

AA   = (speye(nbk*nba)-beta*Q);
BB   = vec(u);
Tvv  = \(AA, BB);
crit = maximum(abs.(kp-kp0));
```

```
vv   = reshape(Tvv,nbk,nba);
v    = copy(vv);
kp0  = copy(kp);
iter = iter+1;

end;

using Plots
plotly() # Choose the Plotly.jl backend for web interactivity
#plot(k',c',linewidth=1,label="Consumption",title="Consumption vs
capital stock")
plot(kp,v,linewidth=1,label="Value Function vs Capital Stock",
title="Value Function")
```