Contents


```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MATLAB Code: HJB_ramsey_implicit_upwind
%
% Author: Muhammad (Follows closely Kiyea and Benn Moll's versions)
% Date: Nov 8, 2024
%
% Description:
% This MATLAB script implements implicit method to solve the HJB equation
% Hugget Model using upwind scheme.
%
% Reference:
% HJB_NGM_implicit.m by Benjamin Moll
% ramsey_implicit.m by Pontus Rendahl
%
% Notes:
% - CRRA utility function: U(c) = (c^(1-gamma))/(1-gamma)
% - Production function: f(k) = A*k^alpha
% - Relative risk aversion coefficient (gamma): 2
% - Discount rate (rho): 0.03
% - Depreciation rate (delta): 0.025
% - Elasticity of output with respect to capital (alpha): 1/3
% - Total fator productivity (A): 1
% - Delta = 1000 (Can be arbitrarily large in implicit method)
% - Try with rho = delta = 0.05
%
% Code Structure:
% 1. DEFINE PARAMETERS
% 2. INITIALIZE GRID POINTS
% 3. PRE-ITERATION INITIALIZATION
% 4. VALUE FUNCTION ITERATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;
close all;
clc;

clear all;
close all;
clc;
```

1. DEFINE PARAMETERS

```
p = define_parameters_Hugget();
```

2. INITIALIZE GRID POINTS

```
% log(k_min) = log(kss)-p.klim
a_min = p.a_min;
a_max = p.a_max;        % This is what Bell Moll uses

a = linspace(a_min, a_max, p.I)';        % Grid for wealth
da = (a_max - a_min)/(p.I-1);       % Grid size

% Every period we are going to have two wealth levels for employed and unemployed
aa = [a,a];
zz = ones(p.I,1)*p.z;       % Income grid
Aswitch = [-speye(p.I)*p.la(1),speye(p.I)*p.la(1);speye(p.I)*p.la(2),-speye(p.I)*p.la(2)]; % Transition matrix, same as our definitions in slides,
```

3. PRE-ITERATION INITIALIZATION

```
% 3-1. Construct the forward and backward differential operator
% Df such that Df*V=dVf and Db such that Db*V=dVb

    Df = zeros(p.I, p.I);
    for i = 1:p.I-1
        Df(i,i) = -1/da; Df(i,i+1) = 1/da;      % Forward differencing
    end
    Df = sparse(Df);

    Db = zeros(p.I, p.I);
    for i = 2:p.I
        Db(i,i-1) = -1/da; Db(i,i) = 1/da;    % Backward differencing
    end
    Db = sparse(Db);

% 3-2. Guess an initial value of the value function (I evalulate utility at the income paths, follows Benn Moll!)

    v0(:,1) = p.u(p.z(1)+p.r.*a)/p.rho;
```

```
    v0(:,2) = p.u(p.z(2)+p.r.*a)/p.rho;

    v = v0;

% 3-3. Pre-allocate arrays for solutions

    dVf = zeros(p.I,2);
    dVb = zeros(p.I,2);
    c = zeros(p.I,2);
```

4. VALUE FUNCTION ITERATION

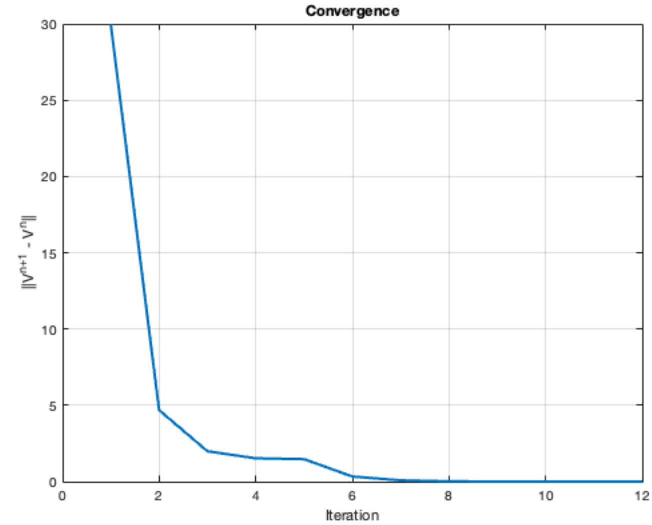
```
tic;

for n = 1:p.maxit
        V= v;
    % 4-1. Compute the derivative of the value function
        dVf(:,1) = Df*V(:,1);
        dVb(:,1) = Db*V(:,1);
        dVf(:,2) = Df*V(:,2);
        dVb(:,2) = Db*V(:,2);

    % BOUNDARY CONDITIONS
        dVf(end,:) = p.mu(p.z+p.r.*a_max); % k<=k_max is enforced which helps stability of the algorithm
        dVb(1,:)   = p.mu(p.z+p.r.*a_min); % k>=k_min is enforced which helps stability of the algorithm

    % 4-2. Compute the optimal consumption and savings with forward differences
        cf = p.inv_mu(dVf);
        cb = p.inv_mu(dVb);

        sf = zz +p.r.*aa  - cf;       % Savings
        sb = zz +p.r.*aa  - cb;

    % UPWIND SCHEME
        If = sf>0;      % If savings is positive, positive drift
        Ib = sb<0;      % If savings is negative, negative drift
        I0 = 1-If-Ib;   % If savings is zero, no drift
        dV0 = p.mu(zz+p.r.*a);

        dV_upwind = dVf.*If + dVb.*Ib + dV0.*I0;

        c = p.inv_mu(dV_upwind);
        u = p.u(c);                     % Benn Moll tracks utility as well, I am also going to keep it for now

    % 4-4. Update the value function: V^(n+1) = [(rho+1/Delta)*I - SD]^(-1)[u(c) + 1/Delta*V^n]

    %CONSTRUCT MATRIX
    X = - min(sb,0)/da;
    Y = - max(sf,0)/da + min(sb,0)/da;
    Z = max(sf,0)/da;

    A1=spdiags(Y(:,1),0,p.I,p.I)+spdiags(X(2:p.I,1),-1,p.I,p.I)+spdiags([0;Z(1:p.I-1,1)],1,p.I,p.I);
    A2=spdiags(Y(:,2),0,p.I,p.I)+spdiags(X(2:p.I,2),-1,p.I,p.I)+spdiags([0;Z(1:p.I-1,2)],1,p.I,p.I);
    A = [A1,sparse(p.I,p.I);sparse(p.I,p.I),A2] + Aswitch;

    % @Muhammad, all clear except definition of A1,A2, and A. Definition of Aswitch is also very clear.

    B = (p.rho + 1/p.Delta)*speye(2*p.I) - A;

    u_stacked = [u(:,1);u(:,2)];
    V_stacked = [V(:,1);V(:,2)];

    b = u_stacked + V_stacked/p.Delta;
    V_stacked = B\b; %SOLVE SYSTEM OF EQUATIONS

    V = [V_stacked(1:p.I),V_stacked(p.I+1:2*p.I)];

    Vchange = V - v;
    v = V;

    dist(n) = max(max(abs(Vchange)));
    if dist(n)<p.tol
        disp('Value Function Converged, Iteration = ')
        disp(n)
        break
    end
end

toc;
```

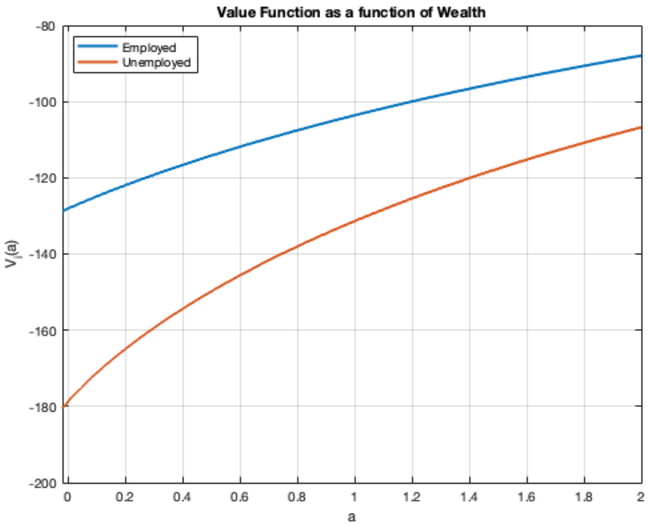
```
Value Function Converged, Iteration =
    12

Elapsed time is 0.024695 seconds.
```

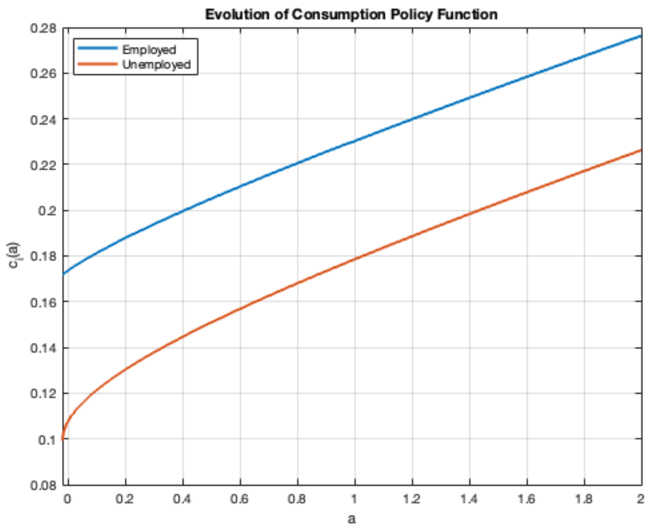
5. Graphs

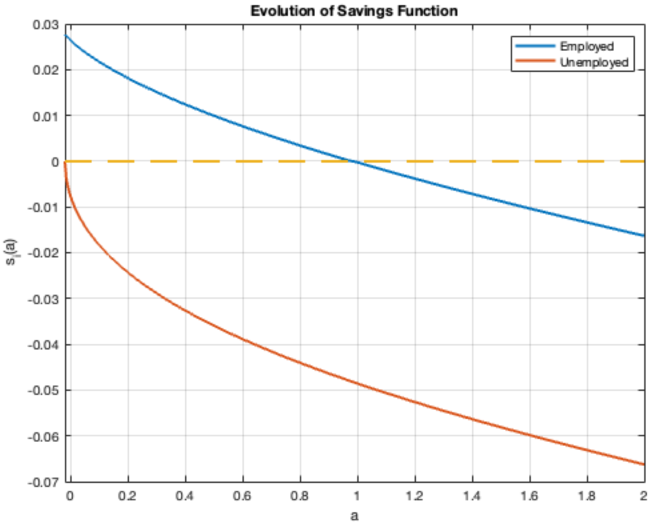
```
figure
set(gca,'FontSize',14)
```

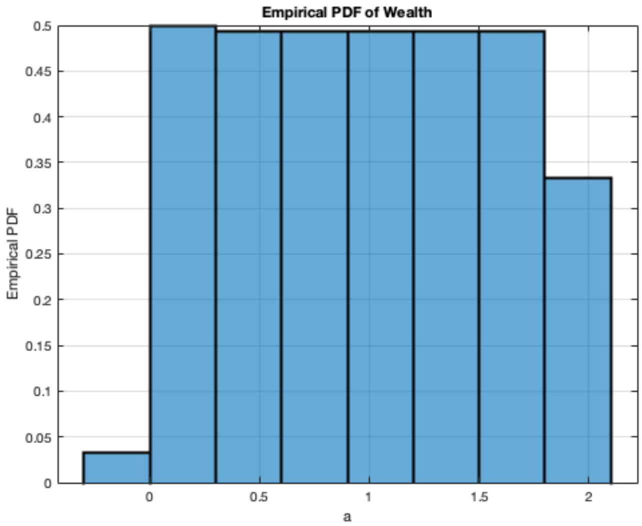
```
plot(dist,'LineWidth',2)
grid
xlabel('Iteration')
ylabel('||V^{n+1} - V^n||')
title('Convergence')


% Verr = c.^(1-s)/(1-s) + dV_Upwind.*(zz + r.*aa - c) + ones(I,1)*la.*(V_switch - V) - rho.*V;
%
% set(gca,'FontSize',14)
% plot(a,Verr,'LineWidth',2)
% grid
% xlabel('k')
% ylabel('Error in HJB Equation')
% xlim([amin amax])

adot = zz + p.r.*aa - c;

% Plot of Value function
figure
set(gca,'FontSize',12)
plot(a,V,'LineWidth',2)
grid
xlabel('a')
ylabel('V_i(a)')
xlim([p.a_min p.a_max])
legend('Employed','Unemployed', 'Location', 'northwest')
title('Value Function as a function of Wealth')

% Plot of Consumption function
figure
set(gca,'FontSize',14)
plot(a,c,'LineWidth',2)
grid
xlabel('a')
ylabel('c_i(a)')
xlim([p.a_min p.a_max])
legend('Employed','Unemployed', 'Location', 'northwest')
title('Evolution of Consumption Policy Function')

% Plot of Savings function
figure
set(gca,'FontSize',14)
plot(a,adot,a,zeros(1,p.I),'--','LineWidth',2)
grid
xlabel('a')
ylabel('s_i(a)')
xlim([p.a_min p.a_max])
legend('Employed','Unemployed', 'Location', 'northeast')
title('Evolution of Savings Function')

% Plot distribution of wealth (pdf)
% Empirical PDF of wealth
figure
set(gca,'FontSize',14)
histogram(a, 'Normalization', 'pdf', 'LineWidth', 2)
grid
xlabel('a')
ylabel('Empirical PDF')
title('Empirical PDF of Wealth')
% Plot fitted kernel density estimate for wealth distribution
figure
set(gca,'FontSize',14)
[f_employed, xi_employed] = ksdensity(a, 'Weights', V(:,1));
[f_unemployed, xi_unemployed] = ksdensity(a, 'Weights', V(:,2));
plot(xi_employed, f_employed, 'LineWidth', 2)
hold on
plot(xi_unemployed, f_unemployed, 'LineWidth', 2)
grid
xlabel('a')
ylabel('Density')
legend('Employed', 'Unemployed', 'Location', 'northwest')
title('Kernel Density Estimate of Wealth Distribution')
% also plot cdfs
figure
set(gca,'FontSize',14)
[f_employed, xi_employed] = ksdensity(a, 'Weights', V(:,1), 'Function', 'cdf');
[f_unemployed, xi_unemployed] = ksdensity(a, 'Weights', V(:,2), 'Function', 'cdf');
plot(xi_employed, f_employed, 'LineWidth', 2)
hold on
plot(xi_unemployed, f_unemployed, 'LineWidth', 2)
grid
xlabel('a')
ylabel('CDF')
legend('Employed', 'Unemployed', 'Location', 'northwest')
title('Empirical CDF of Wealth Distribution')

% In general, employed people always consume more across the wealth
% distribution. Those employed with low wealth, save more and accumulate
% wealth while very wealthy people dissave even if they are employed or
% unemployed. At all wealth levels in this set up, employed people have
% more contintion value then unemployed people.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

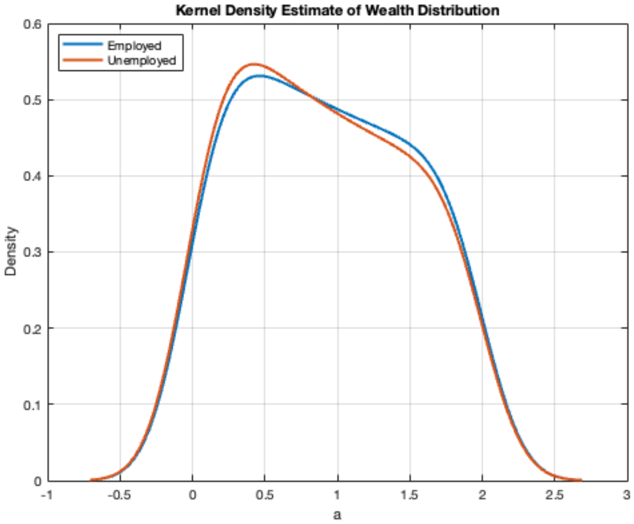

Convergence
$\|V^{n+1} - V^n\|$
Iteration
0
2
4
6
8
10
12
30
25
20
15
10
5

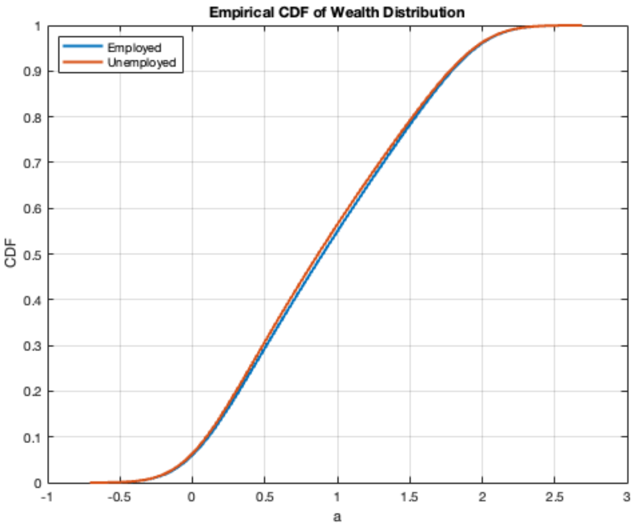

Value Function as a function of Wealth
Employed
Unemployed
$V_i(a)$
a
-80
-100
-120
-140
-160
-180
-200
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


Evolution of Consumption Policy Function
Employed
Unemployed
$c_i(a)$
a
0.28
0.26
0.24
0.22
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


Evolution of Savings Function
Employed
Unemployed
$s_i(a)$
a
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
-0.06
-0.07
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


Empirical PDF of Wealth
Empirical PDF
a
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
0.5
1
1.5
2





Published with MATLAB® R2024b