Coding Exercise 3 Solution: Solving the HJB Equation of the Huggett Model

ECON 202A

November 21, 2025

1. Plot of the Optimal Consumption

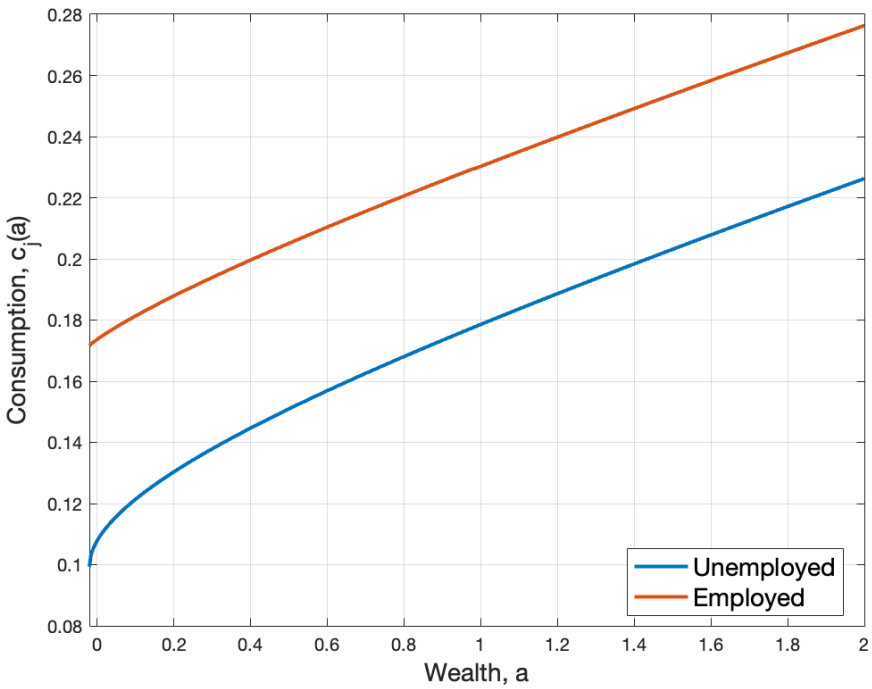

Figure 1: Optimal Consumption Policy for Employed and Unemployed States

For individuals close to the borrowing constraint, consumption is significantly lower, especially for the unemployed. Since they lack sufficient assets and cannot borrow to smooth consumption, they are forced to cut back on consumption more than their employed counterparts. This reduced consumption is a direct consequence of the lack of access to credit. In the Huggett model, with incomplete markets, borrowing constraints prevent individuals from smoothing consumption effectively during periods of income loss.

2. Plot of the Optimal Savings

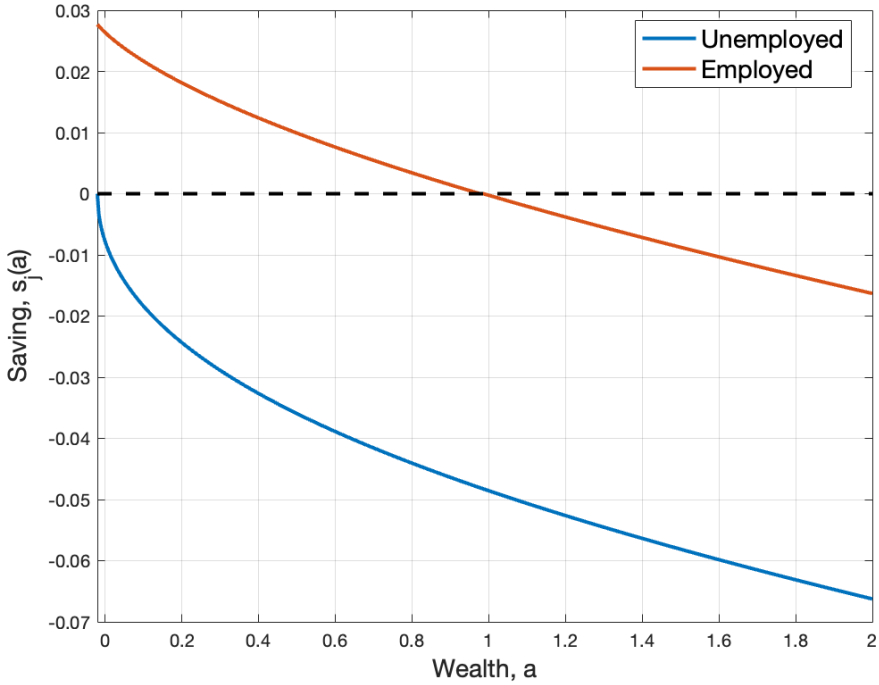

Figure 2: Optimal Savings Policy for Employed and Unemployed States

Employed individuals near the borrowing constraint tend to save a higher proportion of their income compared to those with higher asset levels. This is because they are aware of the risk of potential income shocks, like unemployment, and the difficulty they would face if they were to hit the borrowing constraint. This precautionary savings behavior is particularly pronounced for individuals with low assets, as they are more vulnerable to the adverse effects of income shocks and have limited means to maintain their consumption levels in the absence of sufficient assets.

Unemployed individuals are dissaving. Without income, these individuals must draw down their limited assets to fund consumption, even if it brings them closer to the borrowing constraint. Unemployed individuals near the constraint face a tight consumption limit, since they cannot borrow. This dissaving behavior highlights the difficulties faced by individuals with low asset levels and no income, as they are forced to erode their financial buffer just to meet basic consumption needs.

3. Plot of the Value Function

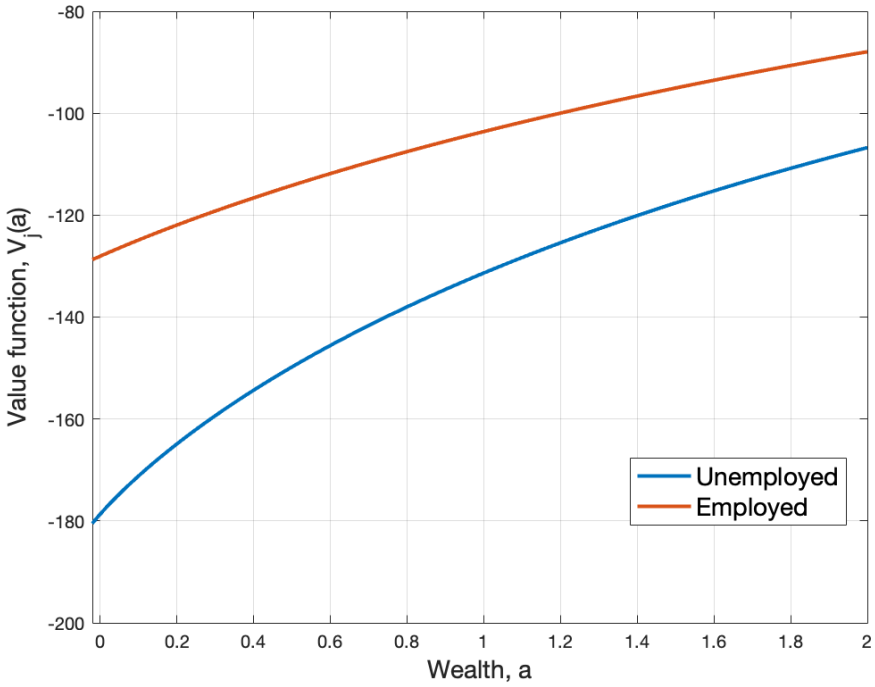

Figure 3: Value Function for Employed and Unemployed States

For individuals with low or near-zero assets (close to the borrowing constraint), the value function drops significantly, particularly for unemployed individuals. This is because they are more exposed to the risk of income shocks without having adequate assets to buffer their consumption. The borrowing constraint prevents these individuals from maintaining their welfare during times of unemployment. Without the option to borrow, they cannot smooth consumption effectively, leading to a sharp decrease in their value function and overall welfare.

4. Discussion of Insights and Implications

Precautionary savings and consumption smoothing behavior impact aggregate demand. During recessions, when income risk increases or unemployment rises, individuals may increase precautionary savings and cut back on consumption. This reduced consumption further depresses aggregate demand, exacerbating the downturn.

In economies with many borrowing-constrained individuals, the effectiveness of traditional monetary policy (e.g., lowering interest rates) may be limited. Lowering interest rates might not stimulate additional borrowing or spending for individuals who are already close to their borrowing limits. This suggests that fiscal policy may play a more significant role in stabilizing demand when borrowing constraints are widespread.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MATLAB Code: HJB_Huggett
%
% Author: Kiyea Jin
% Date: Nov 2, 2024
%
% Description:
% This MATLAB script implements the implicit method to solve a
% Huggett Model using upwind scheme.
%
% Reference: HJB_stateconstraint_implicit.m by Benjamin Moll
%
% Notes:
% - CRRA utility function: U(c) = (c^(1-sigma))/(1-sigma)
% - Elasticity of intertemporal substitution (sigma): 2
% - Interest rate (r) : 0.03
% - Discount rate (rho): 0.05
% - Income: z = [z_u, z_e] = [0.1, 0.2];
% - Lambda: la = [la_u, la_e] = [0.02, 0.03];
% - Discrete grid of asset levels (a): -0.02 to 2
% - Borrowing constraint: a>=-0.02
% - Delta = 1000; (Can be arbitrarily large in implicit method)
%
% Code Structure:
% 1. DEFINE PARAMETERS
% 2. INITIALIZE GRID POINTS
% 3. PRE-ITERATION INITIALIZATION
% 4. VALUE FUNCTION ITERATION
% 5. GRAPHS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;
close all;
clc;

%% 1. DEFINE PARAMETERS

p = define_parameters();

%% 2. INITIALIZE GRID POINTS

a = linspace(p.amin, p.amax, p.I)';
da = (p.amax-p.amin)/(p.I-1);

aa = [a, a]; % I*2 matrix

%% 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;
    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;
    end
    Db = sparse(Db);

% 3-2. Construct A_switch matrix

    A_switch = [speye(p.I).*(-p.lambda(1)), speye(p.I).*p.lambda(1);
                speye(p.I).*p.lambda(2), speye(p.I).*(-p.lambda(2))];

%A_switch = zeros(2*I, 2*I);
%for i=1:I
%    A_switch(i,i) = -lambda(1);
%    A_switch(i,i+I) = lambda(1);
%    A_switch(i+I,i) = lambda(2);
%    A_switch(i+I,i+I) = -lambda(2);
%end

% 3-3. Guess an initial value of the value function
    zz = ones(p.I, 1).*p.zz; % I*2 matrix

    % The value function of "staying put"
    v0 = p.u(zz + p.r.*aa)./p.rho;
    V = v0;

%% 4. VALUE FUNCTION ITERATION

for n=1:p.maxit

    % 4-1. Compute the derivative of the value function
    dVf = Df*V;
    dVb = Db*V;

    % 4-2. Boundary conditions
    dVb(1,:) = p.mu(zz(1,:) + p.r.*aa(1,:)); % a>=a_min is enforced (borrowing constra
    dVf(end,:) = p.mu(zz(end,:) + p.r.*aa(end,:)); % a<=a_max is enforced which helps

    I_concave = dVb > dVf; % indicator whether value function is concave (problems ari

    % 4-3. Compute the optimal consumption
    cf = p.inv_mu(dVf);
    cb = p.inv_mu(dVb);

    % 4-4. Compute the optimal savings
    sf = zz + p.r.*aa - cf;
    sb = zz + p.r.*aa - cb;

    % 4-5. Upwind scheme
    If = sf>0;
```

```
    Ib = sb<0;
    I0 = 1-If-Ib;
    dV0 = p.mu(zz + p.r.*aa); % If sf<=0<=sb, set s=0

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

    c = p.inv_mu(dV_upwind);

    % 4-6. Update value function:
    % Vj^(n+1) = [(rho + 1/Delta)*I - (Sj^n*Dj^n+A_switch)]^(-1)*[u(cj^n) + 1/Delta*Vj

    V_stacked = V(:); % 2I*1 matrix
    c_stacked = c(:); % 2I*1 matrix

    % A = SD
    SD_u = spdiags(If(:,1).*sf(:,1), 0, p.I, p.I)*Df + spdiags(Ib(:,1).*sb(:,1), 0, p.
    SD_e = spdiags(If(:,2).*sf(:,2), 0, p.I, p.I)*Df + spdiags(Ib(:,2).*sb(:,2), 0, p.
    SD = [SD_u, sparse(p.I, p.I);
          sparse(p.I, p.I), SD_e]; % 2I*2I matrix

    % P = A + A_switch
    P = SD + A_switch;

    % B = [(rho + 1/Delta)*I - P]
    B = (p.rho + 1/p.Delta)*speye(2*p.I) - P;

    % b = u(c) + 1/Delta*V
    b = p.u(c_stacked) + (1/p.Delta)*V_stacked;

    % V = B\b;
    V_update = B\b; % 2I*1 matrix
    V_change = V_update - V_stacked;
    V = reshape(V_update, p.I, 2); % I*2 matrix

    % 3-6. Convergence criterion
    dist(n) = max(abs(V_change));
    if dist(n)<p.tol
       disp('Value function converged. Iteration = ')
       disp(n)
       break
    end
end

toc;

%% 5. GRAPHS

set(gca,'FontSize',14)
plot(dist,'LineWidth',2)
grid
xlabel('Iteration')
ylabel('||V^{n+1} - V^n||')

% Verr = c.^(1-s)/(1-s) + dV_Upwind.*(zz + r.*aa - c) + ones(I,1)*la.*(V_switch - V) -
```

```
%
% set(gca,'FontSize',14)
% plot(a,Verr,'LineWidth',2)
% grid
% xlabel('k')
% ylabel('Error in HJB Equation')
% xlim([amin amax])

% 5-1. Optimal consumption

set(gca, 'FontSize', 18)
plot(a, c, 'LineWidth', 2)
grid
xlabel('Wealth, a','FontSize', 14)
ylabel('Consumption, c_j(a)','FontSize', 14)
xlim([p.amin p.amax])
legend('Unemployed', 'Employed', 'Location', 'best', 'FontSize', 14)

% 5-2. Optimal savings

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

set(gca, 'FontSize', 18)
plot(a, adot, a, zeros(1,p.I), '--k', 'LineWidth', 2)
grid
xlabel('Wealth, a', 'FontSize', 14)
ylabel('Saving, s_j(a)', 'FontSize', 14)
xlim([p.amin p.amax])
legend('Unemployed', 'Employed', 'Location', 'best', 'FontSize', 14)

% 5-3. Value function

set(gca, 'FontSize', 18)
plot(a, V, 'LineWidth', 2)
grid
xlabel('Wealth, a', 'FontSize', 14)
ylabel('Value function, V_j(a)', 'FontSize', 14)
xlim([p.amin p.amax])
legend('Unemployed', 'Employed', 'Location', 'best', 'FontSize', 14)
```

```
function p = define_parameters()

% This function defines the parameters needed for the HJB_Huggett.m script

%% Economic Parameters

    % Discount rate
    p.rho = 0.05;

    % Relative risk aversion coefficient
    p.sigma = 2;
```

```
    % Interest rate
    p.r = 0.03;

    % Income
    p.z_u = 0.1;
    p.z_e = 0.2;
    p.zz = [p.z_u, p.z_e];

    % Transition rates
    p.lambda_u = 0.02;
    p.lambda_e = 0.03;
    p.lambda = [p.lambda_u, p.lambda_e];

%% Economic Functions

    % Utility function
        % if sigma == 1
        % p.u = @(c) log(c);
        % else
        % p.u = @(c) (c.^(1-sigma))./(1-sigma);
        % end
    p.u = @(c) (c.^(1-p.sigma))./(1-p.sigma);

    % Marginal utility function
    p.mu = @(c) c.^(-p.sigma);

    % Inverse of marginal utility
    % Note: FOC: mu(c) = dv(a) -> c = inv_mu(dv)
    p.inv_mu = @(dv) dv.^(-1/p.sigma);

%% Grid Paramters

    % The lower bound of the state space (borrowing constraint)
    p.amin = -0.02;

    % The upper bound of the state space
    p.amax = 2;

    % The number of grid points
    p.I = 500;

%% Tuning Parameters

    % The maximum number of iteration
    p.maxit = 100;

    % Convergence criterion
    p.tol = 1e-8;

    % Step size (Delta)
    p.Delta = 1000;

end
```