IMF, Problem Set 2

1068576

4 December 2023

Q1.

The household's problem is:

$$\max_{\{c_t, b_{t+1}\}_{t=1}^{T}} u(c_t) + \beta\mathbb{E}[u(c_{t+1})] \tag{1}$$

subject to:

$$c_t = y_t - b_t + q(b_{t+1})b_{t+1} \tag{2}$$

$$c_T = \max_{D\in\{0,1\}} \{(1-D)[y_T - b_T] + D[y_t - \phi(y_T)]\} \tag{3}$$

where equation (3) is the terminal condition at time period T. This problem can be rewritten recursively as:

$$v^c(b_t) = max_{b_{t+1}} u(y_t - b_t + q(b_{t+1}b_{t+1}) + \beta max\{\mathbb{E}V^b_{t+1}, \mathbb{E}V^c(b_{t+1})\} \tag{4}$$

when the household has access to the international debt market and as:

$$v^b_t = u(y_t - \phi(y_t)) + \sum_{j=1}^{T-t} \beta^j [\pi u(y_H - \hat{\phi}) + (1-\pi)u(y_L)] \tag{5}$$

after the household loses access to the debt market. y_t follows the following distribution:

$$y_t = \begin{cases} y_H, & \text{w.p. } \pi \\ y_L, & \text{w.p. } 1-\pi \end{cases} \tag{6}$$

and the cost of default is defined as:

$$\phi(y_t) = \begin{cases} \hat{\phi}, & \text{if } y_t = y_H \\ 0, & \text{if } y_t = y_L \end{cases} \tag{7}$$

Finally, the price of debt is given by the function:

$$q(b_t) = \beta^* \begin{cases} 1, & \text{if } b_t =\leq 0 \\ (1 - \mathbb{E}[\pi_{def}])^{\sigma}, & \text{if } b_t \in (0, \hat{\phi}) \\ 0, & \text{if } b_t \geq \hat{\phi} \end{cases} \tag{8}$$

where we assume a risk-neutral lender, such that $\sigma = 1$. β^* is the quarterly risk-free rate derived from the subjective discount factor, and $\mathbb{E}[\pi_{def}]$ is the expected probability of default, which is solved numerically. The household's default decision follows:

$$D(b_t) = argmax_{D \in \{0,1\}} \{(1 - D)V^c(b_t) + DV_t^b\}, \quad t < T \tag{9}$$

where we assume no possibility of re-entry into the debt market, such that if the household defaults, it receives utility given by the value function (5) until T.

Q2.

We solve for the model above using a value function iteration algorithm (see Appendix). The parameters and constants used to calibrate the model are presented in table (1).

Table 1: Calibration values

Description	Notation	Calibration value
risk aversion	γ	2.0
subjective discount factor	β	0.97
prob. of good state	π	0.5
endowment in good state	y_H	1.1
endowment in bad state	y_L	0.90
time horizon	T	30

The policy functions for default and borrowing, as well as the equilibrium price of debt are given in figure 1.

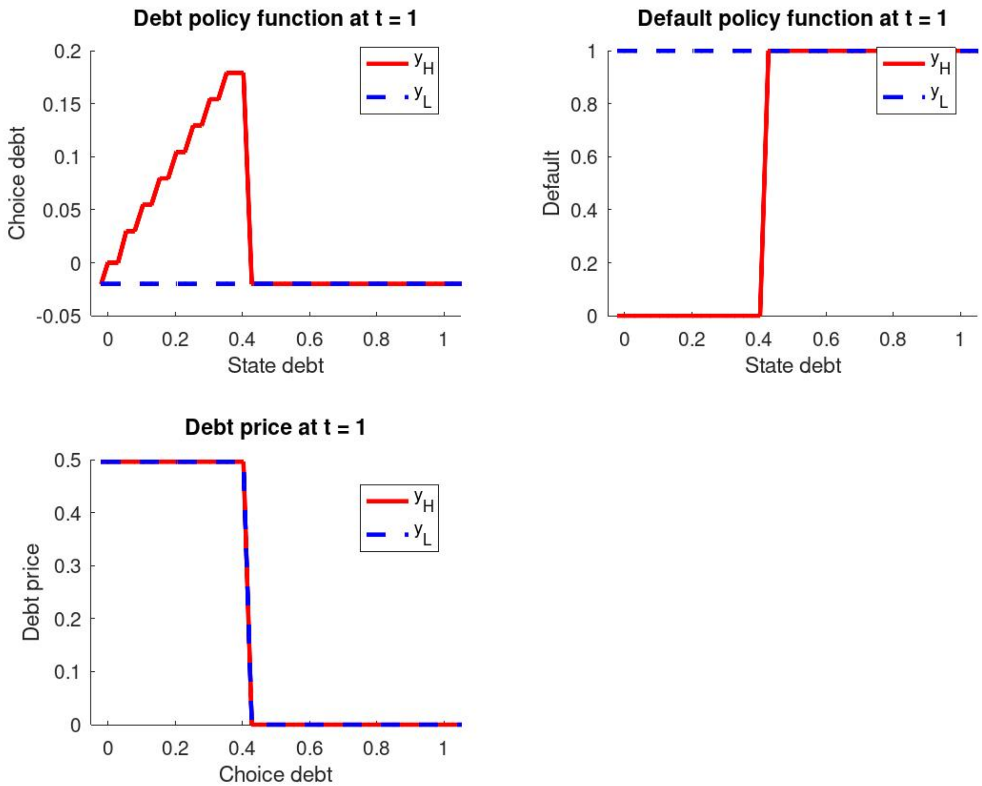

Figure 1: Policy functions and debt price

From the top-right panel of figure (1), we observe that the household defaults at all debt levels when endowment is low to benefit from the low cost of default. Consequently, the household is excluded from the debt market, such that choice debt equals zero.

When endowment is high on the other hand, the household borrows more when the state debt level is higher, to smoothe consumption. Beyond a debt level of approximately 0.4, the household decides to default because the cost of repaying the debt accumulated from the previous period outweighs the benefits of repaying.

The price of debt reflects the current default policy function under rational expectation. The risk-neutral lender bases her belief of the likelihood of default tomorrow on today's default policy function of the borrower.

Q3.

Figure (2) shows the value function when the endowment is high, under the same calibration as in Q2. but with varying time horizons. We observe that the difference between the value functions in the first and second periods becomes smaller as the model's terminal period is extended. This could be because under a long time horizon, the cost of defaulting early is high due to the cumulative cost of exclusion. Therefore, the household decides not to default in both the first and second period, and the small difference between the value functions only reflects the higher risk premium the household in the second period has to pay to the risk-neutral lender to compensate for the higher perceived risk of default.
For shorter time horizons by contrast, it is optimal for the household to default because the cost of future exclusion is low. However, defaulting in the second period is significantly less costly relative to defaulting in the first period due to the short time horizon, which explains the large difference between the value functions.

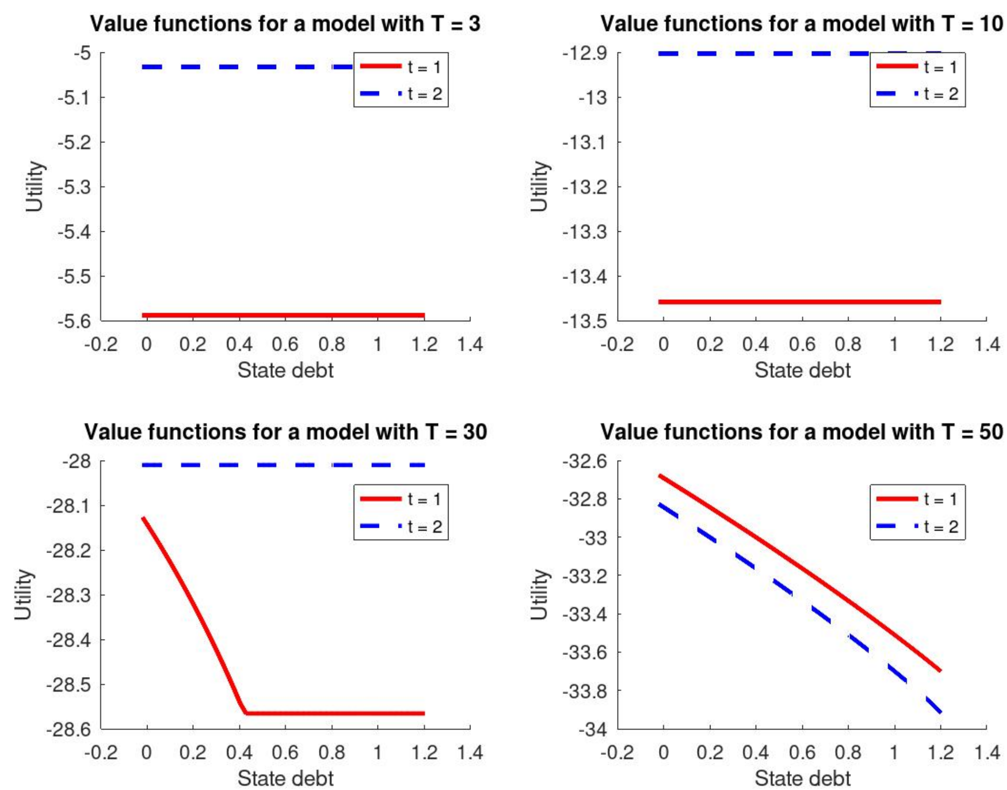

Figure 2: Value function at t = 1 and t = 2 for various time horizons

Q4.

Figures (3) to (5) show the default policy function evaluated at various parameter values. All parameters other than the parameter of interest are calibrated according to table (1).

Unsurprisingly, figure (3) shows that the household defaults at all levels of state debt when the cost of default is low, and does not default when the cost of default is high. At an intermediary cost of default, the household defaults whenever the state debt is large enough such that the benefit of not repaying outweighs the cost of default.

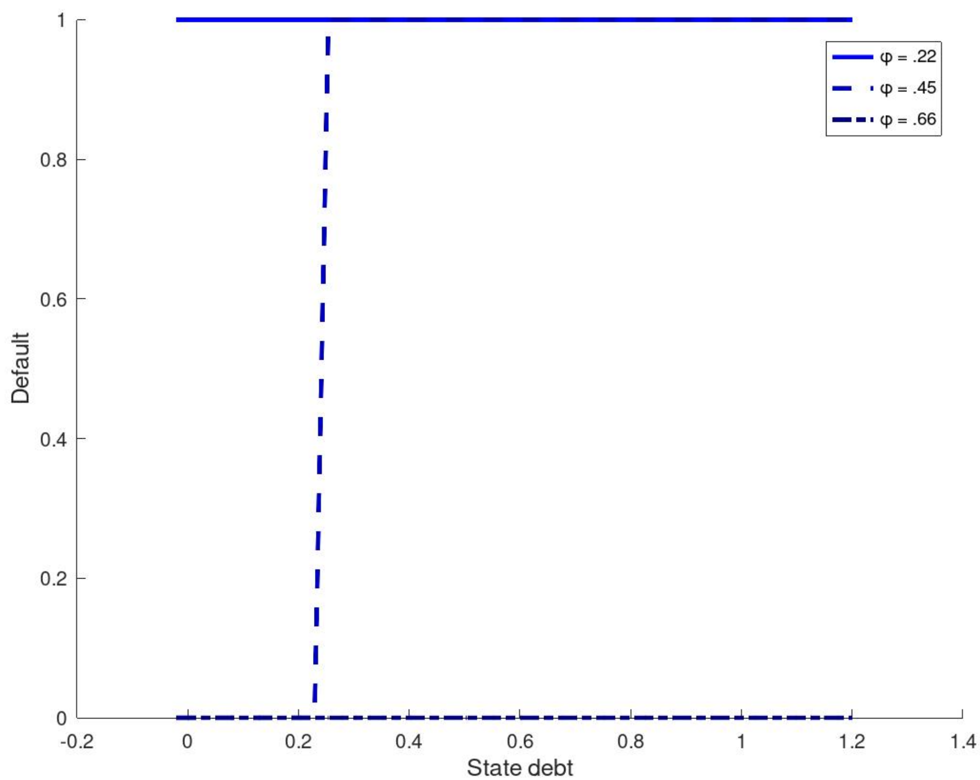

Figure 3: Default policy function given various default costs

From figure (4), we observe that a more risk-averse household defaults at a lower level of state debt. This result is puzzling given that a higher level of risk aversion implies a higher preference for consumption smoothing, which should imply that the household has an incentive to retain access to the debt market.

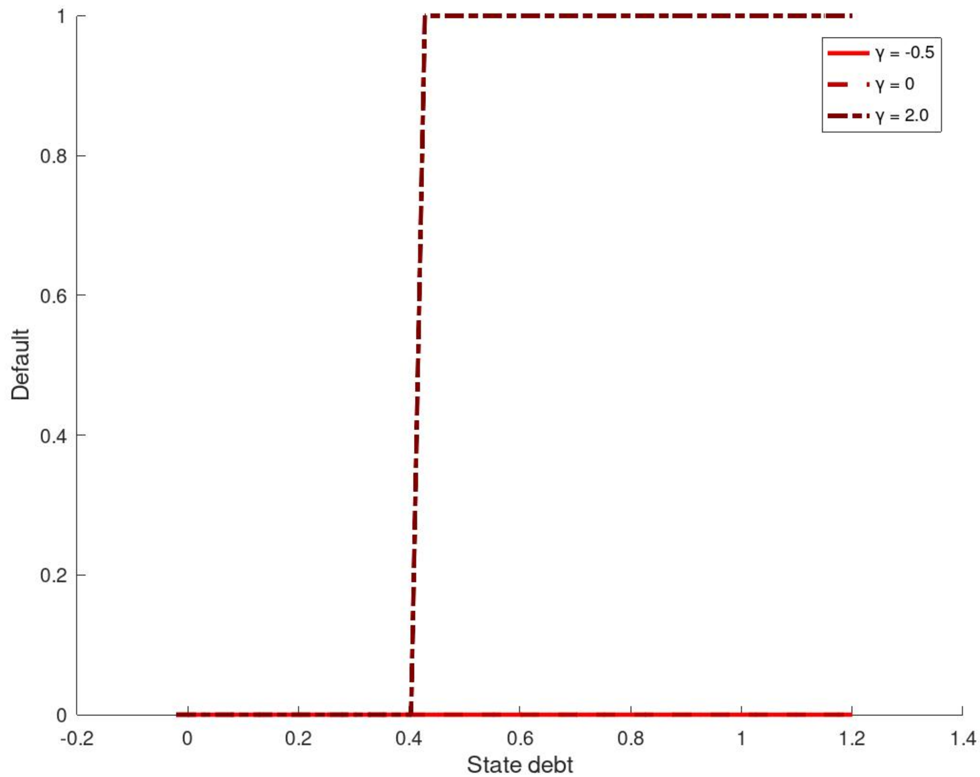

Figure 4: Default policy function given various borrower risk aversions

Figure (5) shows the default policy function for various levels of lender risk aversion. From equation (8), we can set $\sigma < 1$ to model a risk-loving agent and $\sigma > 1$ to model a risk-neutral agent.
From the figure, we immediately observe that a more risk-averse lender implies that the borrower defaults at a lower level of state debt. From figure (6), we can infer that a higher level of lender risk aversion leads to a lower debt price since the lender will require a higher risk premium, making borrowing more costly to the household. Therefore, the cost of being excluded from the debt market is lower for the household if the lender is risk-neutral, which leads to a lower threshold for default.

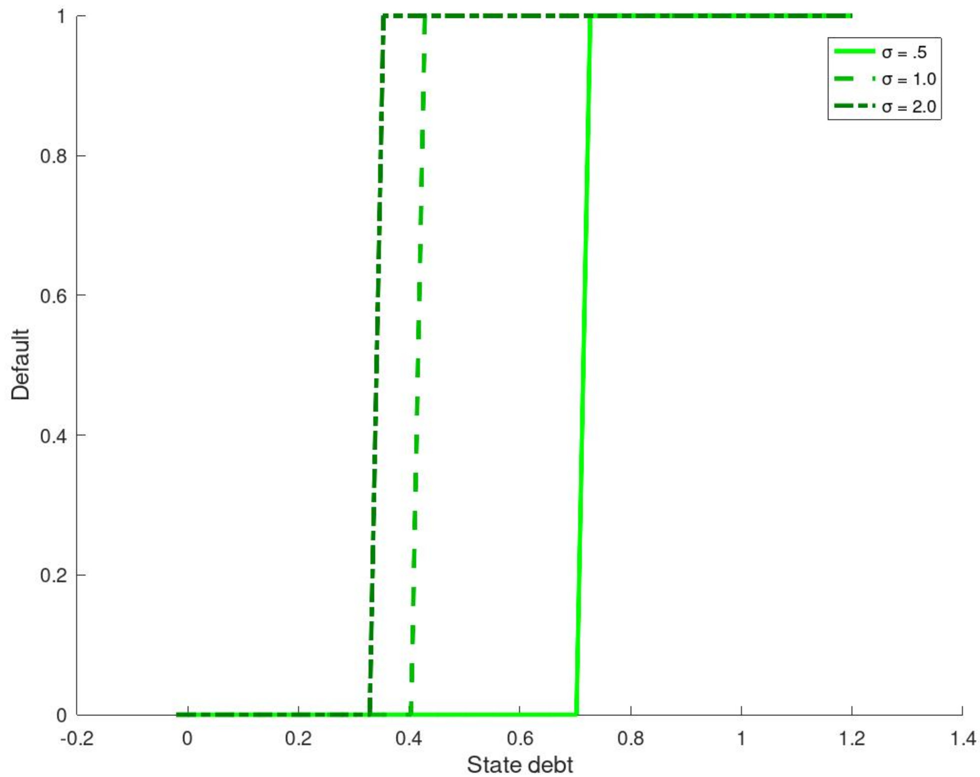

Figure 5: Default policy function given various lender risk aversions

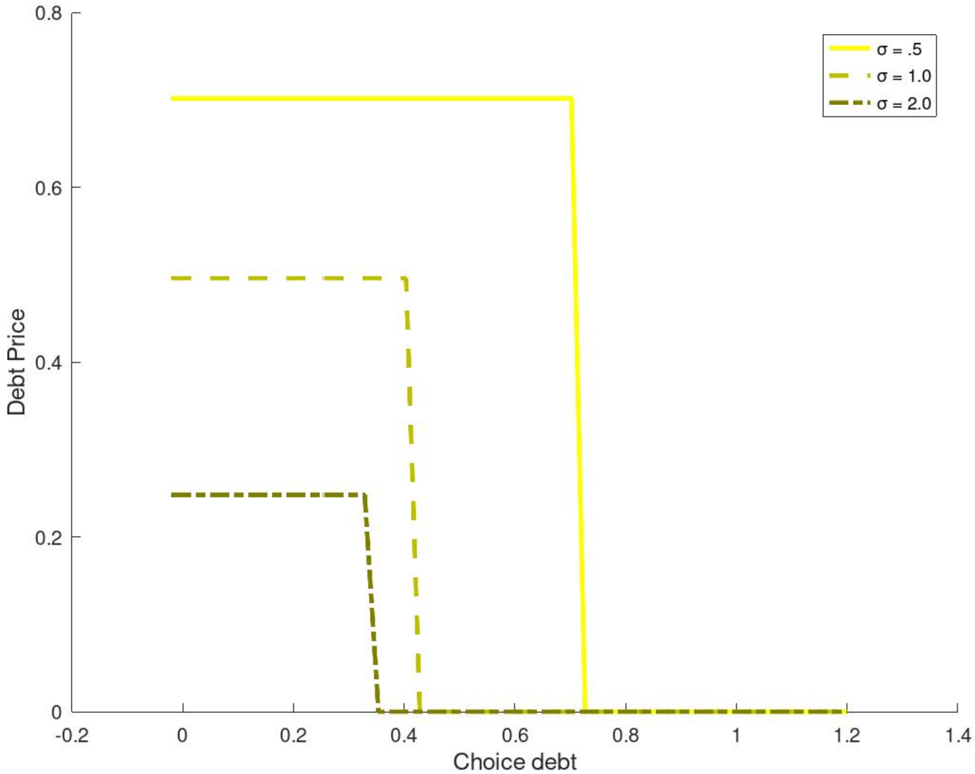

Figure 6: Price of debt given various lender risk aversions

Appendix A: Code

Main file

```
%--------------------------------------------------------------------------
%
% Title: International Macro-Finance Problem Set 2, main file
% Author: Rinto Fujimoto
% Date: 25/11/2023
% Description: Sovereign default model with T periods
%
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
% 0. Housekeeping (close all graphic windows)
%--------------------------------------------------------------------------

close all;
clear all;

cd '/home/rinto/Desktop/International Macro/PS2'

%--------------------------------------------------------------------------
% 1. Defining parameters, grid and variables
%--------------------------------------------------------------------------

% parameters:
par.gamma = 2 % household's RRA preference
par.beta = .97; % subjective discount factor
par.pi = .5; % probability of being in the good state


% constants:
cons.phi_hat = .5; % cost of default in the good state
cons.yh = 1.1; % endowment in good state
cons.yl = .9; % endowment in bad state
cons.rf = par.beta^(-1) - 1; % risk-free rate

% number of time periods:
t = 30;

% define convergence criterion
maxit_q = 1000; % max iteration to solve for q
maxit_v = 1000; % max iteration to solve for v
err_tol_q = 1e-10; % error tolerance
err_tol_v = 1e-10;
lambda_q = .5; % dampening parameter
lambda_v = .5;


% transition matrix
Pr = [par.pi, 1 - par.pi;
      par.pi, 1 - par.pi];
```

```
% derived values:
qrf = (1 + repmat(cons.rf, [2, 1])).^-(1/4); % quarterly risk-free rate


%--------------------------------------------------------------------------
% 2. Defining grid
%--------------------------------------------------------------------------

n = 50; % grid points for b

b_min = -.02; % min value for b
b_max = 1.2; % max value for b

b_vec = linspace(b_min, b_max, n); % grid for debt, 1 x n vector
y_vec = [cons.yh cons.yl].'; % grid for endowment, 2 x 1 vector

% Ensure presence of 0 on b-vector
[~,i_b_zero] = min(abs(b_vec));
% i_b_zero captures the index of the cell containing 0
b_vec(i_b_zero) = 0;

% grid:
grid.b_state3 = repmat(b_vec, [2, 1, n]); % state debt, 2 x n x n
grid.b_choice2 = repmat(b_vec, [2, 1]); % choice debt, 2 x n
grid.y_state3 = repmat(y_vec, [1, n, n]); % state output, 2 x n x n

grid.borr_choice2 = zeros(2, n, t); % grid which will contain debt value
grid.borr_choice3 = zeros(2, n, n, t);
% grid.borr_choice2 augmented by 1 dimension to account for choice debt



%--------------------------------------------------------------------------
% 3. Setting Initial Guesses
%--------------------------------------------------------------------------

guess.v = zeros(2, n, t); % value function
guess.v_new = zeros(2, n, t);

guess.i_b = zeros(2, n, t); % debt choice index (to be extracted from the grid)

guess.q = qrf(1, 1) * ones(2, n, t); % price of debt, 2 x n x t matrix
guess.def = zeros(2, n, t); % default choice


%--------------------------------------------------------------------------
% 4. Defining functions
%--------------------------------------------------------------------------


function u = util(c, gamma)
% this is the utility function (CRRA)
  u = c.^(1 - gamma) / (1 - gamma);
end



%--------------------------------------------------------------------------
```

```
% 5. Other variables
%-----------------------------------------------------------------------------

% storage value
store.v = zeros(2, n, t);
store.i_b = zeros(2, n, t);
store.q = qrf(1, 1) * ones(2, n, t);
store.def = zeros(2, n, t);

% policy functions
policy.b = zeros(2, n, t);
policy.def = zeros(2, n, t);

% others
con_choice = zeros(2, n, n, t); % consumption choice for non-default
util_choice = zeros(2, n, n, t); % utility choice for non-default
borr_maximand = zeros(2, n, n, t); % to be maximised over choice debt
e_def = zeros(2, n, t); % expected default prob.
e_v = zeros(2, n); % continuation value
e_v3 = zeros(2, n, n); % continuation value, augmented
e_v_def = zeros(2, n); % continuation value after default

% sum of discount factors (i.e. (1 - beta^(T-t)) / (1 - beta))
discount = zeros(1, t);
for i = 1:t
  discount(:, i) = par.beta * ((1 - par.beta^(i)) / (1 - par.beta)); % scalar
end
discount = flip(discount, 2);

% consumption in default
con_def = [cons.yh - cons.phi_hat, cons.yl].'; % 2 x 1
con_def3 = repmat(con_def, [1, n, n]);


% Expected value of exclusion state
e_v_def =  discount.' * Pr(1, :) * util(con_def, par.gamma); % t x 1

% evaluate value function in default state
v_def = repmat(util(con_def, par.gamma), [1, n, t]) + permute(repmat(e_v_def, [1, n, 2]), [3, 2, 1]);
% 2 x n x t



%-----------------------------------------------------------------------------
% 5. Value function iteration
%-----------------------------------------------------------------------------


err_q = 7;
err_v = 7;
iter_q = 1;
iter_v = 1;



while err_q > err_tol_q && iter_q < maxit_q

```

```
while err_v > err_tol_v && iter_v < maxit_v

  for i = 1:t

    % Expected continuation value
    e_v = Pr * guess.v(:, :, i);
    e_v3 = permute(repmat(e_v(:, :), [1, 1, n]), [1, 3, 2]); % 2 x n x n matrix
    e_v_def = Pr * v_def(:, :, i);

    optimal_choice = max(e_v3(:, :, i), e_v_def);

    % Resources from borrowing
    grid.borr_choice2(:, :, i) = guess.q(:, :, i) .* grid.b_choice2; % 2 x n matrix
    grid.borr_choice3(:, :, :, i) = repmat(grid.borr_choice2(:, :, i), [1, 1, n]); % 2 x n x n
    grid.borr_choice3(:, :, :, i) = permute(grid.borr_choice3(:, :, :, i), [1, 3, 2]);

    if i < t
      % Consumption implied by choice and state b and guess q
      con_choice(:, :, :, i) = grid.y_state3 - grid.b_state3 + grid.borr_choice3(:, :, :, i); % 2 x n x n matrix
    else
      % Terminal condition
      con_choice(:, :, :, i) = max(grid.y_state3 - grid.b_state3, con_def3);
    end

    con_choice(con_choice < eps) = eps; % rule out negative consumption

    % Period-utility implied by state and choice b
    util_choice(:, :, :, i) = util(con_choice(:, :, :, i), par.gamma); % 2 x 100 x 100

    % Formulate maximand in borrowing choice
    borr_maximand(:, :, :, i) = util_choice(:, :, :, i) + par.beta * e_v3(:, :, :); % 2 x 100 x 100

    % Store old choice for borrowing
    store.i_b(:, :, i) = guess.i_b(:, :, i); % 2 x n

    % Maximise over b' and update value function
    [guess.v_new(:, :, i), guess.i_b(:, :, i)] = max(borr_maximand(:, :, :, i), [], 3); % 2 x n and 2 x n

    % Ensures that HH cannot borrow once defaulted
    guess.i_b(:, :, i) = guess.def(:, :, i) .* ones(2, n) * i_b_zero + (1 - guess.def(:, :, i)) .* guess.i_b(:, :, i);

  end

  %Store old default policy function
  store.def = guess.def;

  % Evaluate policy function for default (indicator function)
  guess.def = v_def > guess.v_new;
  % compare two 2 x n x t matrices

```

```
    % Evaluate value function including discrete choice for default
    guess.v_new = max(v_def, guess.v_new);

    % Store old value-function guesses
    store.v = guess.v;

    % Update value-function guesses
    guess.v = lambda_v * guess.v_new + (1 - lambda_v) * store.v;

    % Evaluate change in v and compare to error tolerance
    err_v = max(abs(guess.v(:) - store.v(:)));

    iter_v
    err_v

    iter_v = iter_v + 1;

end

  % expected probability of default
  for i = 1:t
    e_def(:, :, i) = Pr * guess.def(:, :, i);
  end


  % Store old sovereign debt price
  store.q = guess.q;

  guess.q = qrf(1, 1) * (1 - e_def);

  % Evaluate change in q_g
  err_q = max(abs(guess.q(:) - store.q(:)));

  % Update sovereign debt price
  guess.q = lambda_q * guess.q + (1 - lambda_q) * store.q;

  iter_q
  err_q

  iter_q = iter_q + 1;

  % Reset counter and diff for inner-v loop
  err_v = 7;
  iter_v = 1;

end


policy.b = repmat(grid.b_choice2, [1, 1, t]);
policy.b = policy.b(store.i_b);

policy.def = store.def;

%---------------------------------------------------------------------------
% 6.Plot the value function and policy functions
%---------------------------------------------------------------------------


```

```
% Policy functions and debt price at t = 1
figure
  subplot(2, 2, 1);
    hold on;
    plot(b_vec, policy.b(1, :, 1), 'r-', 'LineWidth', 2.5);
    plot(b_vec, policy.b(2, :, 1), 'b--', 'LineWidth', 2.5);
    xlim([-.05 1.05]);
    xlabel('State debt'), ylabel('Choice debt');
    title('Debt policy function at t = 1');
    legend('y_{H}', 'y_{L}', 'Location');
    hold off;
  subplot(2, 2, 2);
    hold on;
    plot(b_vec, policy.def(1, :, 1), 'r-', 'LineWidth', 2.5);
    plot(b_vec, policy.def(2, :, 1), 'b--', 'LineWidth', 2.5);
    xlim([-.05 1.05]);
    xlabel('State debt'), ylabel('Default');
    title('Default policy function at t = 1');
    legend('y_{H}', 'y_{L}', 'Location');
    hold off;
  subplot(2, 2, 3);
    hold on;
    plot(b_vec, store.q(1, :, 1), 'r-', 'LineWidth', 2.5);
    plot(b_vec, store.q(2, :, 1), 'b--', 'LineWidth', 2.5);
    xlim([-.05 1.05]);
    xlabel('Choice debt'), ylabel('Debt price');
    title('Debt price at t = 1');
    legend('y_{H}', 'y_{L}', 'Location');
    hold off;

saveas(gcf, 'Policy_functions_Q2.jpg');



figure;
  hold on;
  % Value function for high output at t = 1 (solid red line)
  plot(b_vec, store.v(1, :, 1), 'r', 'LineWidth', 2.5);
  % Value function for high output at t = 2 (solid blue line)
  plot(b_vec, store.v(1, :, 2), 'Color', 'b', 'LineWidth', 2.5);
  % Value function for low output at t = 1 (dashed red line)
  plot(b_vec, store.v(2, :, 1), 'Color', 'r', '--', 'LineWidth', 2.5);
  % Value function for low output at t = 2 (dashed blue line)
  plot(b_vec, store.v(2, :, 2), 'Color', 'b', '--', 'LineWidth', 2.5);
  xlabel('Debt levels', 'FontSize', 12);
  ylabel('Value function', 'FontSize', 12);
  title('Value function by time period and endowment levels', 'FontSize', 14);
  % Adding a legend
  legend('High endowment at t=1', 'High endowment at t=2', 'Low endowment at t=1', 'Low endowment at t=2', 'Location', 'best');
  hold off;

saveas(gcf, 'Value_functions_Q2.jpg');
```

Q3-Q4 Main file

```


%--------------------------------------------------------------------------
%
% Title: International Macro-Finance Problem Set 2, Q4
% Author: Rinto Fujimoto
% Date: 25/11/2023
% Description: Sovereign default model with T periods
%
%--------------------------------------------------------------------------


%--------------------------------------------------------------------------
% 0. Housekeeping
%--------------------------------------------------------------------------

close all;
clear all;


%--------------------------------------------------------------------------
% 1. Defining functions
%--------------------------------------------------------------------------

function u = util(c, gamma)
% this is the utility function (CRRA)
  u = c.^(1 - gamma) / (1 - gamma);
end



function [value, default, debt_price, b_vec] = model(time, gamma, sigma, phi_hat)
  % This function performs value function iteration for different
  % parameters.
  % Inputs: time horizon, borrower's risk aversion, lender's risk aversion, cost of default
  % Outputs: value function, default policy function, debt price, grid for b
  % Note: sigma appears in the debt pricing equation


  par.gamma = gamma;
  par.beta = .97;
  par.pi = .5;

  cons.phi_hat = phi_hat;
  cons.yh = 1.1;
  cons.yl = .9;
  cons.rf = par.beta^(-1) - 1;

  t = time;

  maxit_q = 1000;
  maxit_v = 1000;
  err_tol_q = 1e-10;
  err_tol_v = 1e-10;
  lambda_v = .5;
```

```
lambda_q = .5;

Pr = [par.pi, 1 - par.pi;
      par.pi, 1 - par.pi];

qrf = (1 + repmat(cons.rf, [2, 1])).^-(1/4);

n = 50;

b_min = -.02;
b_max = 1.2;

b_vec = linspace(b_min, b_max, n);
y_vec = [cons.yh cons.yl].';

[~,i_b_zero] = min(abs(b_vec));
b_vec(i_b_zero) = 0;

grid.b_state3 = repmat(b_vec, [2, 1, n]);
grid.b_choice2 = repmat(b_vec, [2, 1]);
grid.y_state3 = repmat(y_vec, [1, n, n]);

grid.borr_choice2 = zeros(2, n, t);
grid.borr_choice3 = zeros(2, n, n, t);

guess.v = zeros(2, n, t);
guess.v_new = zeros(2, n, t);

guess.i_b = zeros(2, n, t);

guess.q = qrf(1, 1) * ones(2, n, t);
guess.def = zeros(2, n, t);

store.v = zeros(2, n, t);
store.i_b = zeros(2, n, t);
store.q = qrf(1, 1) * ones(2, n, t);
store.def = zeros(2, n, t);

policy.b = zeros(2, n, t);
policy.def = zeros(2, n, t);

con_choice = zeros(2, n, n, t);
util_choice = zeros(2, n, n, t);
borr_maximand = zeros(2, n, n, t);
e_def = zeros(2, n, t);
e_v = zeros(2, n);
e_v3 = zeros(2, n, n);
e_v_def = zeros(2, n);

discount = zeros(1, t);
for i = 1:t
  discount(:, i) = par.beta * ((1 - par.beta^(i)) / (1 - par.beta));
end
discount = flip(discount, 2);

con_def = [cons.yh - cons.phi_hat, cons.yl].';
con_def3 = repmat(con_def, [1, n, n]);

```

```
e_v_def =  discount.' * Pr(1, :) * util(con_def, par.gamma);

v_def = repmat(util(con_def, par.gamma), [1, n, t]) + permute(repmat(e_v_def, [1, n, 2]), [3, 2, 1]);

err_q = 7;
err_v = 7;
iter_q = 1;
iter_v = 1;



while err_q > err_tol_q && iter_q < maxit_q

  while err_v > err_tol_v && iter_v < maxit_v

    for i = 1:t

      e_v = Pr * guess.v(:, :, i);
      e_v3 = permute(repmat(e_v(:, :), [1, 1, n]), [1, 3, 2]);
      e_v_def = Pr * v_def(:, :, i);

      optimal_choice = max(e_v3(:, :, i), e_v_def);

      grid.borr_choice2(:, :, i) = guess.q(:, :, i) .* grid.b_choice2;
      grid.borr_choice3(:, :, :, i) = repmat(grid.borr_choice2(:, :, i), [1, 1, n]);
      grid.borr_choice3(:, :, :, i) = permute(grid.borr_choice3(:, :, :, i), [1, 3, 2]);

      if i < t
        con_choice(:, :, :, i) = grid.y_state3 - grid.b_state3 + grid.borr_choice3(:, :, :, i);
      else
        con_choice(:, :, :, i) = max(grid.y_state3 - grid.b_state3, con_def3);
      end

      con_choice(con_choice < eps) = eps;

      util_choice(:, :, :, i) = util(con_choice(:, :, :, i), par.gamma);

      borr_maximand(:, :, :, i) = util_choice(:, :, :, i) + par.beta * e_v3(:, :, :);

      store.i_b(:, :, i) = guess.i_b(:, :, i);

      [guess.v_new(:, :, i), guess.i_b(:, :, i)] = max(borr_maximand(:, :, :, i), [], 3); % 2 x n and 2 x n

      guess.i_b(:, :, i) = guess.def(:, :, i) .* ones(2, n) * i_b_zero + (1 - guess.def(:, :, i)) .* guess.i_b(:, :, i);

    end

    store.def = guess.def;

    guess.def = v_def > guess.v_new;

```

```
      guess.v_new = max(v_def, guess.v_new);

      store.v = guess.v;

      guess.v = lambda_v * guess.v_new + (1 - lambda_v) * store.v;

      err_v = max(abs(guess.v(:) - store.v(:)));

      iter_v
      err_v

      iter_v = iter_v + 1;

  end

    for i = 1:t
      e_def(:, :, i) = Pr * guess.def(:, :, i);
    end

    store.q = guess.q;

    % sigma is the lender's risk aversion
    guess.q = qrf(1, 1) * (1 - e_def).^sigma;

    err_q = max(abs(guess.q(:) - store.q(:)));

    guess.q = lambda_q * guess.q + (1 - lambda_q) * store.q;

    iter_q
    err_q

    iter_q = iter_q + 1;

    err_v = 7;
    iter_v = 1;

  end


  policy.b = repmat(grid.b_choice2, [1, 1, t]);
  policy.b = policy.b(store.i_b);
  policy.def = store.def;

  default = policy.def;
  debt_price = store.q;
  value = store.v;

end



%----------------------------------------------------------------
% 2. Running functions with different parameters
%----------------------------------------------------------------

[value1, ~, ~, b_vec] = model(20, 2, 1.0, .5);
value2 = model(30, 2, 1.0, .5);
value3 = model(40, 2, 1.0, .5);
```

```
value4 = model(50, 2, 1.0, .5);

[~, default1] = model(30, -.5, 1.0, .5);
[~, default2] = model(30, 0, 1.0, .5);
[~, default3] = model(30, 2.0, 1.0, .5);

[~, default4, debt_price1] = model(30, 2, .5, .5);
[~, default5, debt_price2] = model(30, 2, 1.0, .5);
[~, default6, debt_price3] = model(30, 2, 2.0, .5);

[~, default7] = model(30, 2, 1.0, .22);
[~, default8] = model(30, 2, 1.0, .45);
[~, default9] = model(30, 2, 1.0, .66);



%--------------------------------------------------------------------------
% 3.Plot the value function and policy functions
%--------------------------------------------------------------------------


% Value function at t=1 and t=2 for various time horizons
figure
  subplot(2, 2, 1);
    hold on;
    plot(b_vec, value1(1, :, 1), 'r-', 'LineWidth', 2.5);
    plot(b_vec, value1(2, :, 1), 'b--', 'LineWidth', 2.5);
    xlabel('State debt'), ylabel('Utility');
    title('Value functions for a model with T = 3');
    legend('t = 1', 't = 2');
    hold off;
  subplot(2, 2, 2);
    hold on;
    plot(b_vec, value2(1, :, 1), 'r-', 'LineWidth', 2.5);
    plot(b_vec, value2(2, :, 1), 'b--', 'LineWidth', 2.5);
    xlabel('State debt'), ylabel('Utility');
    title('Value functions for a model with T = 10');
    legend('t = 1', 't = 2');
    hold off;
  subplot(2, 2, 3);
    hold on;
    plot(b_vec, value3(1, :, 1), 'r-', 'LineWidth', 2.5);
    plot(b_vec, value3(2, :, 1), 'b--', 'LineWidth', 2.5);
    xlabel('State debt'), ylabel('Utility');
    title('Value functions for a model with T = 30');
    legend('t = 1', 't = 2');
    hold off;
  subplot(2, 2, 4);
    hold on;
    plot(b_vec, value4(1, :, 1), 'r-', 'LineWidth', 2.5);
    plot(b_vec, value4(2, :, 1), 'b--', 'LineWidth', 2.5);
    xlabel('State debt'), ylabel('Utility');
    title('Value functions for a model with T = 50');
    legend('t = 1', 't = 2');
    hold off;

    saveas(gcf, 'Value_functions_Q3.jpg');

```

```
figure;
  hold on;
  % Default policy function for risk-loving borrower (darker red)
  plot(b_vec, default1(1, :, 1), 'Color', [1 0 0], 'LineWidth', 2.5);
  % Default policy function for risk-neutral borrower
  plot(b_vec, default2(1, :, 1), 'Color', [.75 0 0], '--', 'LineWidth', 2.5);
  % Default policy function for risk-averse borrower (lighter red)
  plot(b_vec, default3(1, :, 1), 'Color', [.5 0 0], '-.', 'LineWidth', 2.5);
  xlabel('State debt', 'FontSize', 12);
  ylabel('Default', 'FontSize', 12)
  % Adding a legend
  legend('\gamma = -0.5', '\gamma = 0', '\gamma = 2.0', 'Location', 'best');
  hold off;

saveas(gcf, 'Default_risk_borrower_Q4.jpg');



figure;
  hold on;
  % Default policy function for risk-loving lender (darker green)
  plot(b_vec, default4(1, :, 1), 'Color', [0 1 0], 'LineWidth', 2.5);
  % Default policy function for risk-neutral lender
  plot(b_vec, default5(1, :, 1), 'Color', [0 .75 0], '--', 'LineWidth', 2.5);
  % Default policy function for risk-averse lender (lighter green)
  plot(b_vec, default6(1, :, 1), 'Color', [0 .5 0], '-.', 'LineWidth', 2.5);
  xlabel('State debt', 'FontSize', 12);
  ylabel('Default', 'FontSize', 12);
  % Adding a legend
  legend('\sigma = .5', '\sigma = 1.0', '\sigma = 2.0', 'Location', 'best');
  hold off;

saveas(gcf, 'Default_risk_lender_Q4.jpg');


figure;
  hold on;
  % Debt price for risk-loving lender (darker yellow)
  plot(b_vec, debt_price1(1, :, 1), 'Color', [1 1 0], 'LineWidth', 2.5);
  % Debt price for risk-neutral lender
  plot(b_vec, debt_price2(1, :, 1), 'Color', [.75 .75 0], '--', 'LineWidth', 2.5);
  % Debt price for risk-averse lender (lighter yellow)
  plot(b_vec, debt_price3(1, :, 1), 'Color', [.5 .5 0], '-.', 'LineWidth', 2.5);
  xlabel('Choice debt', 'FontSize', 12);
  ylabel('Debt Price', 'FontSize', 12);
  % Adding a legend
  legend('\sigma = .5', '\sigma = 1.0', '\sigma = 2.0', 'Location', 'best');
  hold off;

saveas(gcf, 'Debt_price_risk_lender_Q4.jpg');



figure;
  hold on;
  % Default policy function for low default cost (darker blue)
  plot(b_vec, default7(1, :, 1), 'Color', [0 0 1], 'LineWidth', 2.5);
```

```
  % Default policy function for medium default cost
  plot(b_vec, default8(1, :, 1), 'Color', [0 0 .75], '--', 'LineWidth', 2.5);
  % Default policy function for high default cost (lighter blue)
  plot(b_vec, default9(1, :, 1), 'Color', [0 0 .5], '-.', 'LineWidth', 2.5);
  xlabel('State debt', 'FontSize', 12);
  ylabel('Default', 'FontSize', 12);
  % Adding a legend
  legend('\phi = .22', '\phi = .45', '\phi = .66', 'Location', 'best');
  hold off;

saveas(gcf, 'Default_cost_Q4.jpg');
```