

IMF, Problem Set 2

1068576

4 December 2023

Q1.

The household's problem is:

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

subject to:

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

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

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

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

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

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

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

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

and the cost of default is defined as:

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

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

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

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:

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

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.

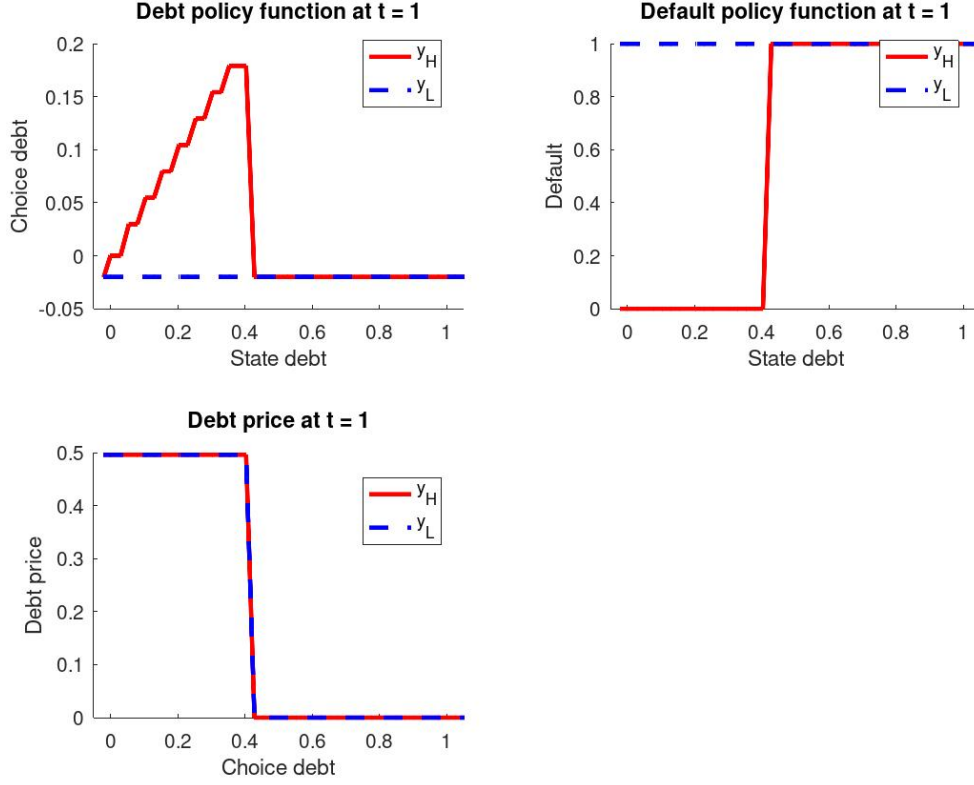


Figure 1: Policy functions and debt price

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.

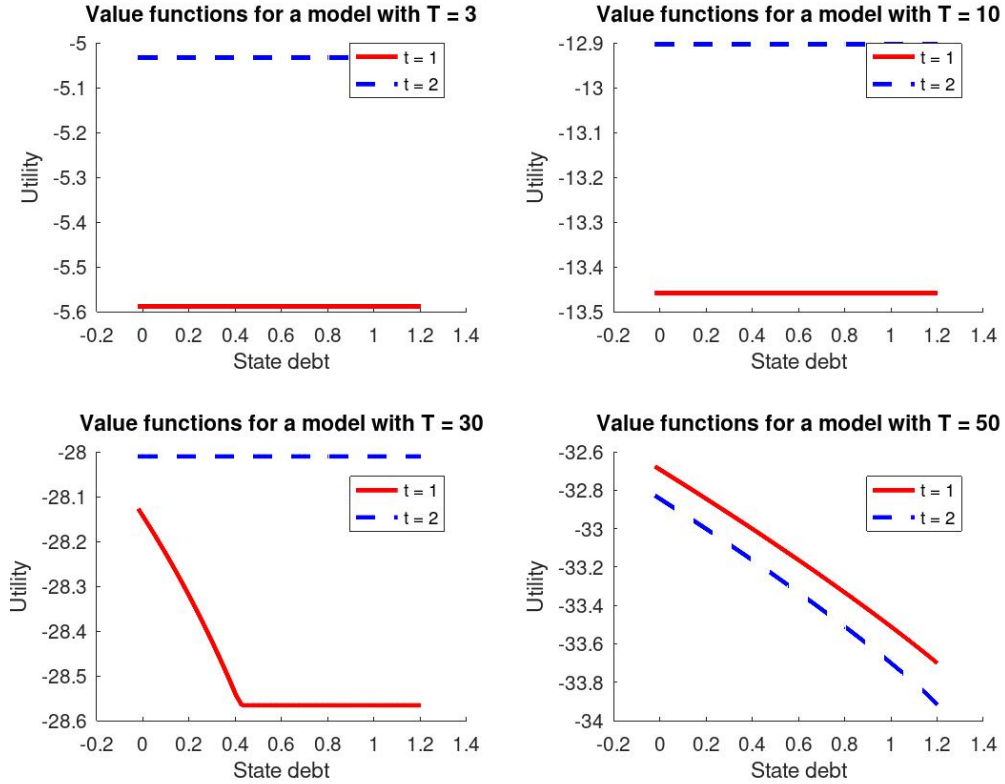


Figure 2: Value function at $t = 1$ and $t = 2$ for various time horizons

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.

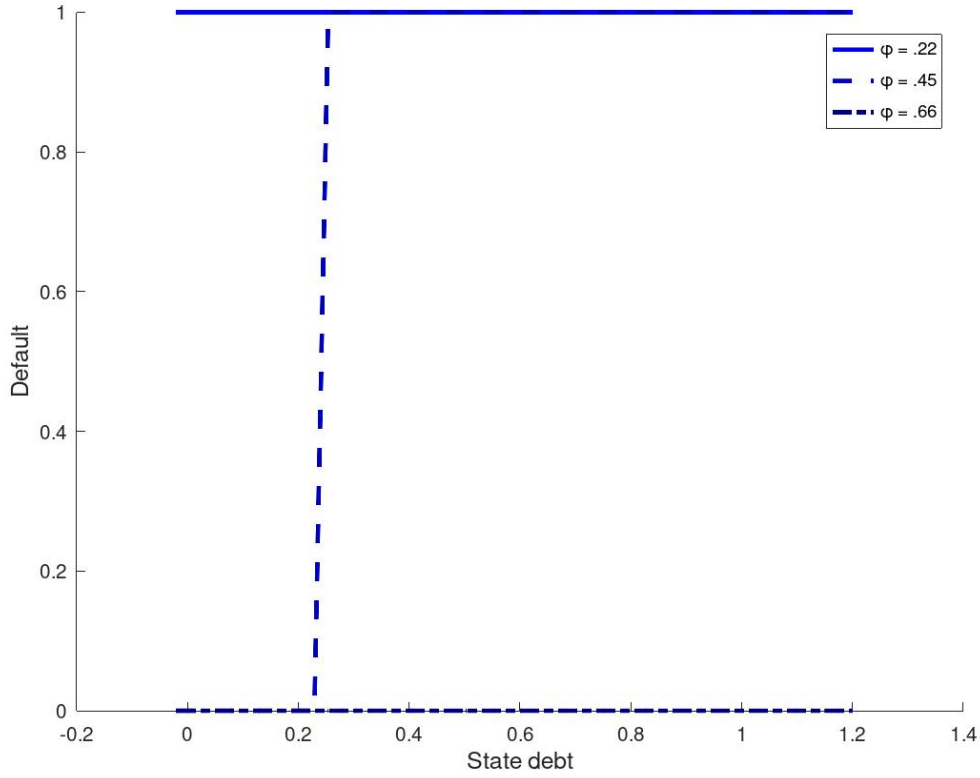


Figure 3: Default policy function given various default costs

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.

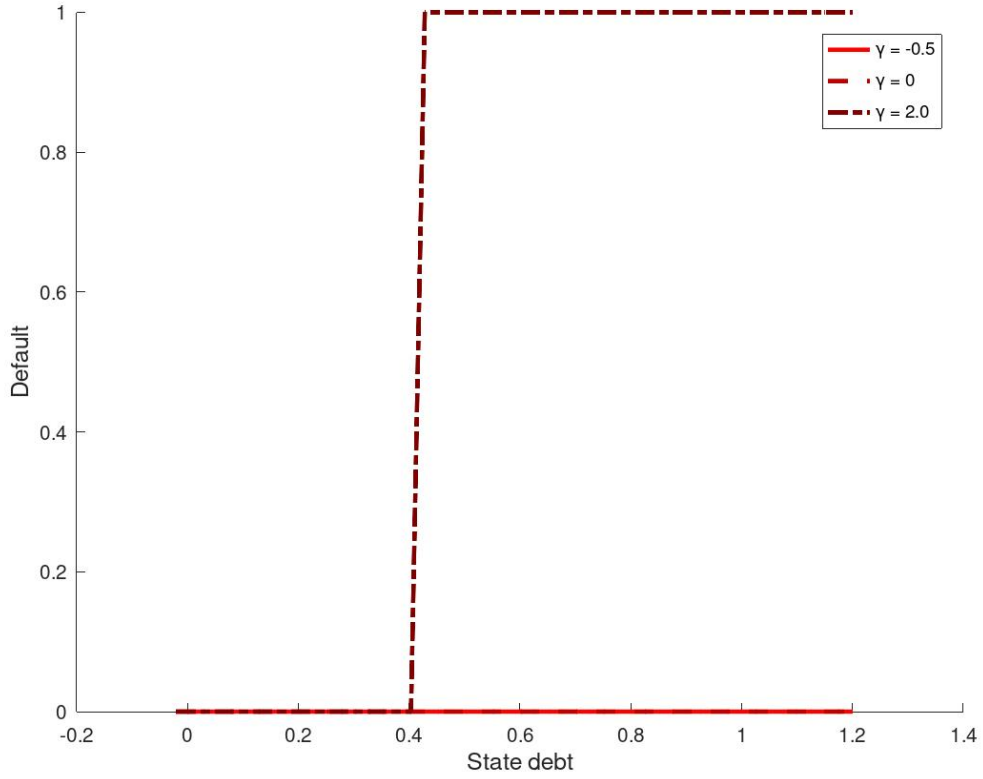


Figure 4: Default policy function given various borrower risk aversions

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.

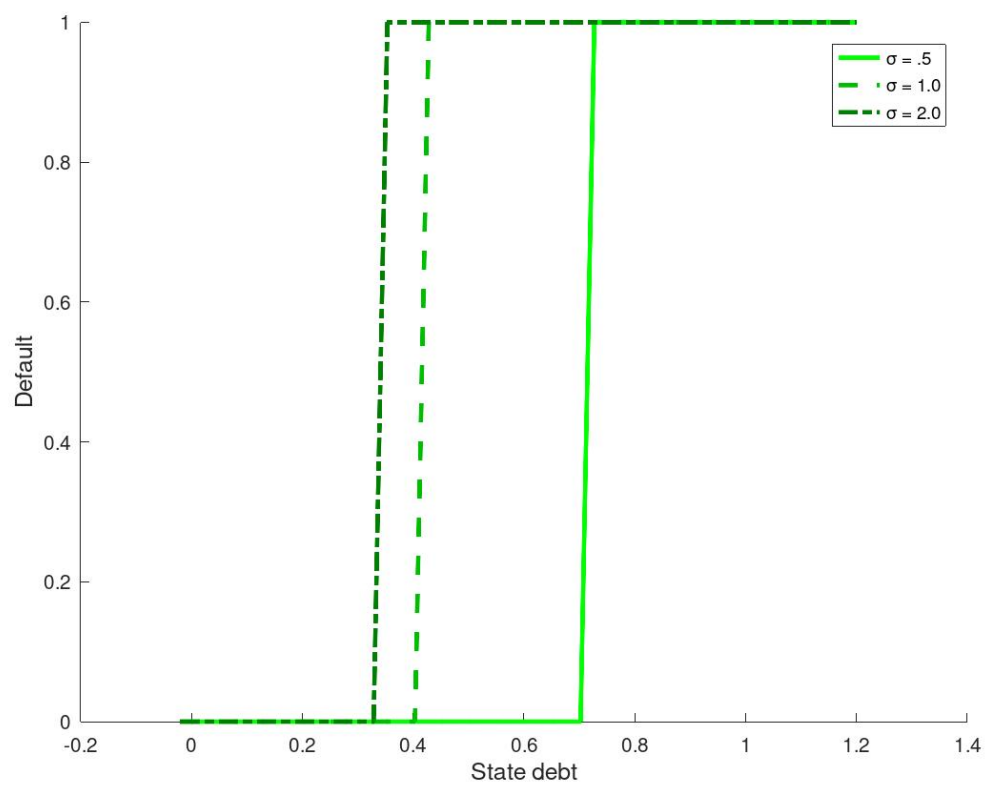


Figure 5: Default policy function given various lender risk aversions

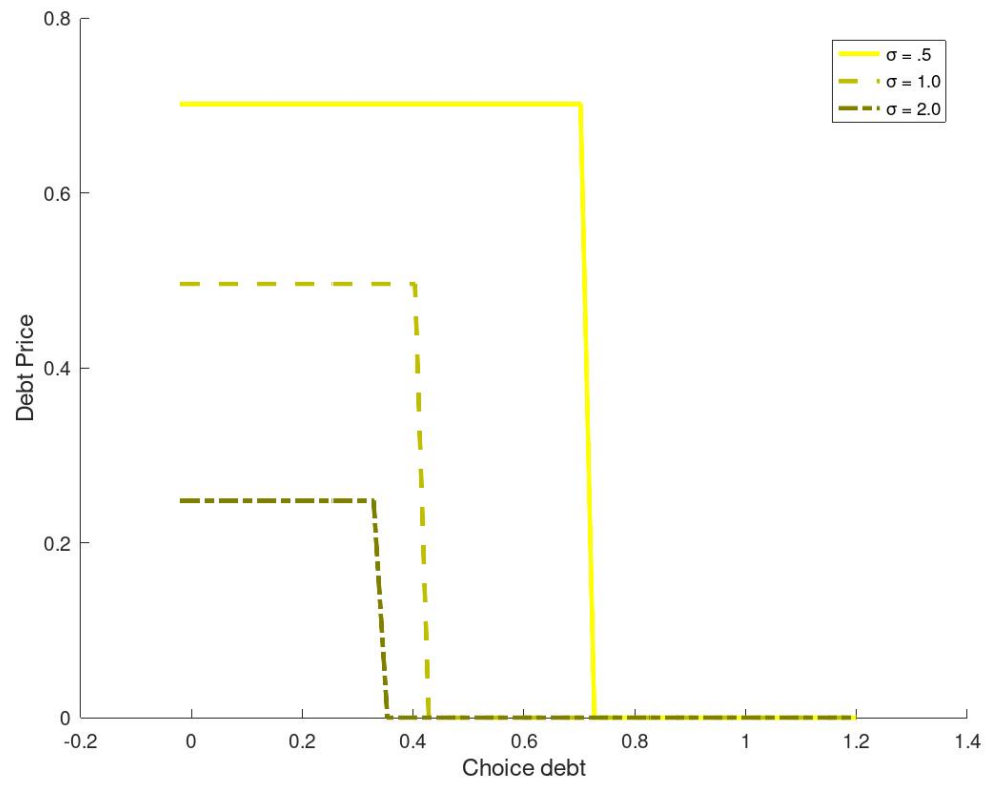


Figure 6: Price of debt given various lender risk aversions

Appendix A: Code

Main file

```
1
2
3 %-----
4 %
5 % Title: International Macro-Finance Problem Set 2, main file
6 % Author: Rinto Fujimoto
7 % Date: 25/11/2023
8 % Description: Sovereign default model with T periods
9 %
10 %-----
11
12 %-----
13 % 0. Housekeeping (close all graphic windows)
14 %-----
15
16 close all;
17 clear all;
18
19 cd '/home/rinto/Desktop/International Macro/PS2'
20
21 %-----
22 % 1. Defining parameters, grid and variables
23 %-----
24
25 % parameters:
26 par.gamma = 2 % household's RRA preference
27 par.beta = .97; % subjective discount factor
28 par.pi = .5; % probability of being in the good state
29
30
31 % constants:
32 cons.phi_hat = .5; % cost of default in the good state
33 cons.yh = 1.1; % endowment in good state
34 cons.yl = .9; % endowment in bad state
35 cons.rf = par.beta^(-1) - 1; % risk-free rate
36
37 % number of time periods:
38 t = 30;
39
40 % define convergence criterion
41 maxit_q = 1000; % max iteration to solve for q
42 maxit_v = 1000; % max iteration to solve for v
43 err_tol_q = 1e-10; % error tolerance
44 err_tol_v = 1e-10;
45 lambda_q = .5; % dampening parameter
46 lambda_v = .5;
47
48
49 % transition matrix
50 Pr = [par.pi, 1 - par.pi;
51       par.pi, 1 - par.pi];
52
53
```

```

54 % derived values:
55 qrf = (1 + repmat(cons.rf, [2, 1])).^(1/4); % quarterly risk-free rate
56
57
58 %-----
59 % 2. Defining grid
60 %-----
61
62 n = 50; % grid points for b
63
64 b_min = -.02; % min value for b
65 b_max = 1.2; % max value for b
66
67 b_vec = linspace(b_min, b_max, n); % grid for debt, 1 x n vector
68 y_vec = [cons.yh cons.yl].'; % grid for endowment, 2 x 1 vector
69
70 % Ensure presence of 0 on b-vector
71 [~, i_b_zero] = min(abs(b_vec));
72 % i_b_zero captures the index of the cell containing 0
73 b_vec(i_b_zero) = 0;
74
75 % grid:
76 grid.b_state3 = repmat(b_vec, [2, 1, n]); % state debt, 2 x n x n
77 grid.b_choice2 = repmat(b_vec, [2, 1]); % choice debt, 2 x n
78 grid.y_state3 = repmat(y_vec, [1, n, n]); % state output, 2 x n x n
79
80 grid.borr_choice2 = zeros(2, n, t); % grid which will contain debt value
81 grid.borr_choice3 = zeros(2, n, n, t);
82 % grid.borr_choice2 augmented by 1 dimension to account for choice debt
83
84
85
86 %-----
87 % 3. Setting Initial Guesses
88 %-----
89
90 guess.v = zeros(2, n, t); % value function
91 guess.v_new = zeros(2, n, t);
92
93 guess.i_b = zeros(2, n, t); % debt choice index (to be extracted from the grid)
94
95 guess.q = qrf(1, 1) * ones(2, n, t); % price of debt, 2 x n x t matrix
96 guess.def = zeros(2, n, t); % default choice
97
98
99 %-----
100 % 4. Defining functions
101 %-----
102
103
104 function u = util(c, gamma)
105 % this is the utility function (CRRA)
106 u = c.^(1 - gamma) / (1 - gamma);
107 end
108
109
110
111 %-----

```

```

112 % 5. Other variables
113 %-----
114
115 % storage value
116 store.v = zeros(2, n, t);
117 store.i_b = zeros(2, n, t);
118 store.q = qrf(1, 1) * ones(2, n, t);
119 store.def = zeros(2, n, t);
120
121 % policy functions
122 policy.b = zeros(2, n, t);
123 policy.def = zeros(2, n, t);
124
125 % others
126 con_choice = zeros(2, n, n, t); % consumption choice for non-default
127 util_choice = zeros(2, n, n, t); % utility choice for non-default
128 borr_maximand = zeros(2, n, n, t); % to be maximised over choice debt
129 e_def = zeros(2, n, t); % expected default prob.
130 e_v = zeros(2, n); % continuation value
131 e_v3 = zeros(2, n, n); % continuation value, augmented
132 e_v_def = zeros(2, n); % continuation value after default
133
134 % sum of discount factors (i.e.  $(1 - \beta^{T-t}) / (1 - \beta)$ )
135 discount = zeros(1, t);
136 for i = 1:t
137     discount(:, i) = par.beta * ((1 - par.beta^(i)) / (1 - par.beta)); % scalar
138 end
139 discount = flip(discount, 2);
140
141 % consumption in default
142 con_def = [cons.yh - cons.phi_hat, cons.yl].'; % 2 x 1
143 con_def3 = repmat(con_def, [1, n, n]);
144
145
146 % Expected value of exclusion state
147 e_v_def = discount.' * Pr(1, :) * util(con_def, par.gamma); % t x 1
148
149 % evaluate value function in default state
150 v_def = repmat(util(con_def, par.gamma), [1, n, t]) + permute(repmat(e_v_def, [1, n,
151     2]), [3, 2, 1]);
152 % 2 x n x t
153
154
155 %-----
156 % 5. Value function iteration
157 %-----
158
159
160 err_q = 7;
161 err_v = 7;
162 iter_q = 1;
163 iter_v = 1;
164
165
166
167 while err_q > err_tol_q && iter_q < maxit_q
168

```

```

169 while err_v > err_tol_v && iter_v < maxit_v
170
171     for i = 1:t
172
173         % Expected continuation value
174         e_v = Pr * guess.v(:, :, i);
175         e_v3 = permute(repmat(e_v(:, :), [1, 1, n]), [1, 3, 2]); % 2 x n x n matrix
176         e_v_def = Pr * v_def(:, :, i);
177
178         optimal_choice = max(e_v3(:, :, i), e_v_def);
179
180         % Resources from borrowing
181         grid.borr_choice2(:, :, i) = guess.q(:, :, i) .* grid.b_choice2; % 2 x n
matrix
182         grid.borr_choice3(:, :, :, i) = repmat(grid.borr_choice2(:, :, i), [1, 1, n]);
% 2 x n x n
183         grid.borr_choice3(:, :, :, i) = permute(grid.borr_choice3(:, :, :, i), [1, 3,
2]);
184
185         if i < t
186             % Consumption implied by choice and state b and guess q
187             con_choice(:, :, :, i) = grid.y_state3 - grid.b_state3 + grid.borr_choice3
(:, :, :, i); % 2 x n x n matrix
188         else
189             % Terminal condition
190             con_choice(:, :, :, i) = max(grid.y_state3 - grid.b_state3, con_def3);
191         end
192
193         con_choice(con_choice < eps) = eps; % rule out negative consumption
194
195         % Period-utility implied by state and choice b
196         util_choice(:, :, :, i) = util(con_choice(:, :, :, i), par.gamma); % 2 x 100 x
100
197
198         % Formulate maximand in borrowing choice
199         borr_maximand(:, :, :, i) = util_choice(:, :, :, i) + par.beta * e_v3(:, :, :);
% 2 x 100 x 100
200
201         % Store old choice for borrowing
202         store.i_b(:, :, i) = guess.i_b(:, :, i); % 2 x n
203
204         % Maximise over b' and update value function
205         [guess.v_new(:, :, i), guess.i_b(:, :, i)] = max(borr_maximand(:, :, :, i),
[], 3); % 2 x n and 2 x n
206
207         % Ensures that HH cannot borrow once defaulted
208         guess.i_b(:, :, i) = guess.def(:, :, i) .* ones(2, n) * i_b_zero + (1 - guess.
def(:, :, i)) .* guess.i_b(:, :, i);
209
210     end
211
212     %Store old default policy function
213     store.def = guess.def;
214
215     % Evaluate policy function for default (indicator function)
216     guess.def = v_def > guess.v_new;
217     % compare two 2 x n x t matrices
218

```

```

219 % Evaluate value function including discrete choice for default
220 guess.v_new = max(v_def, guess.v_new);
221
222 % Store old value-function guesses
223 store.v = guess.v;
224
225 % Update value-function guesses
226 guess.v = lambda_v * guess.v_new + (1 - lambda_v) * store.v;
227
228 % Evaluate change in v and compare to error tolerance
229 err_v = max(abs(guess.v(:) - store.v(:)));
230
231 iter_v
232 err_v
233
234 iter_v = iter_v + 1;
235
236 end
237
238 % expected probability of default
239 for i = 1:t
240     e_def(:, :, i) = Pr * guess.def(:, :, i);
241 end
242
243 % Store old sovereign debt price
244 store.q = guess.q;
245
246 guess.q = qrf(1, 1) * (1 - e_def);
247
248 % Evaluate change in q-g
249 err_q = max(abs(guess.q(:) - store.q(:)));
250
251 % Update sovereign debt price
252 guess.q = lambda_q * guess.q + (1 - lambda_q) * store.q;
253
254 iter_q
255 err_q
256
257 iter_q = iter_q + 1;
258
259 % Reset counter and diff for inner-v loop
260 err_v = 7;
261 iter_v = 1;
262
263 end
264
265
266 policy.b = repmat(grid.b_choice2, [1, 1, t]);
267 policy.b = policy.b(store.i_b);
268
269 policy.def = store.def;
270
271
272 %-----
273 % 6. Plot the value function and policy functions
274 %-----
275
276

```

```

277 % Policy functions and debt price at t = 1
278 figure
279 subplot(2, 2, 1);
280 hold on;
281 plot(b_vec, policy.b(1, :, 1), 'r-', 'LineWidth', 2.5);
282 plot(b_vec, policy.b(2, :, 1), 'b--', 'LineWidth', 2.5);
283 xlim([-0.05 1.05]);
284 xlabel('State debt'), ylabel('Choice debt');
285 title('Debt policy function at t = 1');
286 legend('y_{H}', 'y_{L}', 'Location');
287 hold off;
288 subplot(2, 2, 2);
289 hold on;
290 plot(b_vec, policy.def(1, :, 1), 'r-', 'LineWidth', 2.5);
291 plot(b_vec, policy.def(2, :, 1), 'b--', 'LineWidth', 2.5);
292 xlim([-0.05 1.05]);
293 xlabel('State debt'), ylabel('Default');
294 title('Default policy function at t = 1');
295 legend('y_{H}', 'y_{L}', 'Location');
296 hold off;
297 subplot(2, 2, 3);
298 hold on;
299 plot(b_vec, store.q(1, :, 1), 'r-', 'LineWidth', 2.5);
300 plot(b_vec, store.q(2, :, 1), 'b--', 'LineWidth', 2.5);
301 xlim([-0.05 1.05]);
302 xlabel('Choice debt'), ylabel('Debt price');
303 title('Debt price at t = 1');
304 legend('y_{H}', 'y_{L}', 'Location');
305 hold off;
306
307 saveas(gcf, 'Policy_functions_Q2.jpg');
308
309
310
311 figure;
312 hold on;
313 % Value function for high output at t = 1 (solid red line)
314 plot(b_vec, store.v(1, :, 1), 'r', 'LineWidth', 2.5);
315 % Value function for high output at t = 2 (solid blue line)
316 plot(b_vec, store.v(1, :, 2), 'Color', 'b', 'LineWidth', 2.5);
317 % Value function for low output at t = 1 (dashed red line)
318 plot(b_vec, store.v(2, :, 1), 'Color', 'r', 'LineDash', [4 4], 'LineWidth', 2.5);
319 % Value function for low output at t = 2 (dashed blue line)
320 plot(b_vec, store.v(2, :, 2), 'Color', 'b', 'LineDash', [4 4], 'LineWidth', 2.5);
321 xlabel('Debt levels', 'FontSize', 12);
322 ylabel('Value function', 'FontSize', 12);
323 title('Value function by time period and endowment levels', 'FontSize', 14);
324 % Adding a legend
325 legend('High endowment at t=1', 'High endowment at t=2', 'Low endowment at t=1', 'Low endowment at t=2', 'Location', 'best');
326 hold off;
327
328 saveas(gcf, 'Value_functions_Q2.jpg');

```

Q3-Q4 Main file

```
1
2
3 %-----
4 %
5 % Title: International Macro–Finance Problem Set 2, Q4
6 % Author: Rinto Fujimoto
7 % Date: 25/11/2023
8 % Description: Sovereign default model with T periods
9 %
10 %-----
11
12
13 %-----
14 % 0. Housekeeping
15 %-----
16
17 close all;
18 clear all;
19
20
21 %-----
22 % 1. Defining functions
23 %-----
24
25 function u = util(c, gamma)
26 % this is the utility function (CRRA)
27     u = c.^(1 - gamma) / (1 - gamma);
28 end
29
30
31
32 function [value, default, debt_price, b_vec] = model(time, gamma, sigma, phi_hat)
33 % This function performs value function iteration for different
34 % parameters.
35 % Inputs: time horizon, borrower's risk aversion, lender's risk aversion, cost of
36 % default
37 % Outputs: value function, default policy function, debt price, grid for b
38 % Note: sigma appears in the debt pricing equation
39
40 par.gamma = gamma;
41 par.beta = .97;
42 par.pi = .5;
43
44 cons.phi_hat = phi_hat;
45 cons.yh = 1.1;
46 cons.yl = .9;
47 cons.rf = par.beta^(-1) - 1;
48
49 t = time;
50
51 maxit_q = 1000;
52 maxit_v = 1000;
53 err_tol_q = 1e-10;
54 err_tol_v = 1e-10;
55 lambda_v = .5;
```



```

56 lambda_q = .5;
57
58 Pr = [par.pi, 1 - par.pi;
59       par.pi, 1 - par.pi];
60
61 qrf = (1 + repmat(cons.rf, [2, 1])).^(1/4);
62
63 n = 50;
64
65 b_min = -.02;
66 b_max = 1.2;
67
68 b_vec = linspace(b_min, b_max, n);
69 y_vec = [cons.yh cons.yl].';
70
71 [~, i_b_zero] = min(abs(b_vec));
72 b_vec(i_b_zero) = 0;
73
74 grid.b_state3 = repmat(b_vec, [2, 1, n]);
75 grid.b_choice2 = repmat(b_vec, [2, 1]);
76 grid.y_state3 = repmat(y_vec, [1, n, n]);
77
78 grid.borr_choice2 = zeros(2, n, t);
79 grid.borr_choice3 = zeros(2, n, n, t);
80
81 guess.v = zeros(2, n, t);
82 guess.v_new = zeros(2, n, t);
83
84 guess.i_b = zeros(2, n, t);
85
86 guess.q = qrf(1, 1) * ones(2, n, t);
87 guess.def = zeros(2, n, t);
88
89 store.v = zeros(2, n, t);
90 store.i_b = zeros(2, n, t);
91 store.q = qrf(1, 1) * ones(2, n, t);
92 store.def = zeros(2, n, t);
93
94 policy.b = zeros(2, n, t);
95 policy.def = zeros(2, n, t);
96
97 con_choice = zeros(2, n, n, t);
98 util_choice = zeros(2, n, n, t);
99 borr_maximand = zeros(2, n, n, t);
100 e_def = zeros(2, n, t);
101 e_v = zeros(2, n);
102 e_v3 = zeros(2, n, n);
103 e_v_def = zeros(2, n);
104
105 discount = zeros(1, t);
106 for i = 1:t
107     discount(:, i) = par.beta * ((1 - par.beta^(i)) / (1 - par.beta));
108 end
109 discount = flip(discount, 2);
110
111 con_def = [cons.yh - cons.phi_hat, cons.yl].';
112 con_def3 = repmat(con_def, [1, n, n]);
113

```

```

114 e_v_def = discount.' * Pr(1, :) * util(con_def, par.gamma);
115
116 v_def = repmat(util(con_def, par.gamma), [1, n, t]) + permute(repmat(e_v_def, [1,
117     n, 2]), [3, 2, 1]);
118
119 err_q = 7;
120 err_v = 7;
121 iter_q = 1;
122 iter_v = 1;
123
124
125 while err_q > err_tol_q && iter_q < maxit_q
126
127     while err_v > err_tol_v && iter_v < maxit_v
128
129         for i = 1:t
130
131             e_v = Pr * guess.v(:, :, i);
132             e_v3 = permute(repmat(e_v(:, :), [1, 1, n]), [1, 3, 2]);
133             e_v_def = Pr * v_def(:, :, i);
134
135             optimal_choice = max(e_v3(:, :, i), e_v_def);
136
137             grid.borr_choice2(:, :, i) = guess.q(:, :, i) .* grid.b_choice2;
138             grid.borr_choice3(:, :, :, i) = repmat(grid.borr_choice2(:, :, i), [1, 1, n
139 ]);
140             grid.borr_choice3(:, :, :, i) = permute(grid.borr_choice3(:, :, :, i), [1,
141 3, 2]);
142
143             if i < t
144                 con_choice(:, :, :, i) = grid.y_state3 - grid.b_state3 + grid.borr_choice3
145                 (:, :, :, i);
146             else
147                 con_choice(:, :, :, i) = max(grid.y_state3 - grid.b_state3, con_def3);
148             end
149
150             con_choice(con_choice < eps) = eps;
151
152             util_choice(:, :, :, i) = util(con_choice(:, :, :, i), par.gamma);
153
154             borr_maximand(:, :, :, i) = util_choice(:, :, :, i) + par.beta * e_v3(:, :,
155 :);
156
157             store.i_b(:, :, i) = guess.i_b(:, :, i);
158
159             [guess.v_new(:, :, i), guess.i_b(:, :, i)] = max(borr_maximand(:, :, :, i),
160 [], 3); % 2 x n and 2 x n
161
162             guess.i_b(:, :, i) = guess.def(:, :, i) .* ones(2, n) * i_b_zero + (1 -
163 guess.def(:, :, i)) .* guess.i_b(:, :, i);
164
165         end
166
167         store.def = guess.def;
168
169         guess.def = v_def > guess.v_new;

```

```

165     guess.v_new = max(v_def, guess.v_new);
166
167     store.v = guess.v;
168
169     guess.v = lambda_v * guess.v_new + (1 - lambda_v) * store.v;
170
171     err_v = max(abs(guess.v(:) - store.v(:)));
172
173     iter_v
174     err_v
175
176     iter_v = iter_v + 1;
177
178 end
179
180 for i = 1:t
181     e_def(:, :, i) = Pr * guess.def(:, :, i);
182 end
183
184 store.q = guess.q;
185
186 % sigma is the lender's risk aversion
187 guess.q = qrf(1, 1) * (1 - e_def).^sigma;
188
189 err_q = max(abs(guess.q(:) - store.q(:)));
190
191 guess.q = lambda_q * guess.q + (1 - lambda_q) * store.q;
192
193 iter_q
194 err_q
195
196 iter_q = iter_q + 1;
197
198 err_v = 7;
199 iter_v = 1;
200
201 end
202
203
204 policy.b = repmat(grid.b_choice2, [1, 1, t]);
205 policy.b = policy.b(store.i_b);
206 policy.def = store.def;
207
208 default = policy.def;
209 debt_price = store.q;
210 value = store.v;
211
212 end
213
214
215
216 %
217 % 2. Running functions with different parameters
218 %
219
220 [value1, ~, ~, b_vec] = model(20, 2, 1.0, .5);
221 value2 = model(30, 2, 1.0, .5);
222 value3 = model(40, 2, 1.0, .5);

```

```

223 value4 = model(50, 2, 1.0, .5);
224
225 [~, default1] = model(30, -.5, 1.0, .5);
226 [~, default2] = model(30, 0, 1.0, .5);
227 [~, default3] = model(30, 2.0, 1.0, .5);
228
229 [~, default4, debt_price1] = model(30, 2, .5, .5);
230 [~, default5, debt_price2] = model(30, 2, 1.0, .5);
231 [~, default6, debt_price3] = model(30, 2, 2.0, .5);
232
233 [~, default7] = model(30, 2, 1.0, .22);
234 [~, default8] = model(30, 2, 1.0, .45);
235 [~, default9] = model(30, 2, 1.0, .66);
236
237
238
239 %
240 % 3. Plot the value function and policy functions
241 %
242
243
244 % Value function at t=1 and t=2 for various time horizons
245 figure
246     subplot(2, 2, 1);
247     hold on;
248     plot(b_vec, value1(1, :, 1), 'r-', 'LineWidth', 2.5);
249     plot(b_vec, value1(2, :, 1), 'b--', 'LineWidth', 2.5);
250     xlabel('State debt'), ylabel('Utility');
251     title('Value functions for a model with T = 3');
252     legend('t = 1', 't = 2');
253     hold off;
254     subplot(2, 2, 2);
255     hold on;
256     plot(b_vec, value2(1, :, 1), 'r-', 'LineWidth', 2.5);
257     plot(b_vec, value2(2, :, 1), 'b--', 'LineWidth', 2.5);
258     xlabel('State debt'), ylabel('Utility');
259     title('Value functions for a model with T = 10');
260     legend('t = 1', 't = 2');
261     hold off;
262     subplot(2, 2, 3);
263     hold on;
264     plot(b_vec, value3(1, :, 1), 'r-', 'LineWidth', 2.5);
265     plot(b_vec, value3(2, :, 1), 'b--', 'LineWidth', 2.5);
266     xlabel('State debt'), ylabel('Utility');
267     title('Value functions for a model with T = 30');
268     legend('t = 1', 't = 2');
269     hold off;
270     subplot(2, 2, 4);
271     hold on;
272     plot(b_vec, value4(1, :, 1), 'r-', 'LineWidth', 2.5);
273     plot(b_vec, value4(2, :, 1), 'b--', 'LineWidth', 2.5);
274     xlabel('State debt'), ylabel('Utility');
275     title('Value functions for a model with T = 50');
276     legend('t = 1', 't = 2');
277     hold off;
278
279     saveas(gcf, 'Value-functions-Q3.jpg');
280

```

```

281
282 figure;
283 hold on;
284 % Default policy function for risk-loving borrower (darker red)
285 plot(b_vec, default1(1, :, 1), 'Color', [1 0 0], 'LineWidth', 2.5);
286 % Default policy function for risk-neutral borrower
287 plot(b_vec, default2(1, :, 1), 'Color', [.75 0 0], '--', 'LineWidth', 2.5);
288 % Default policy function for risk-averse borrower (lighter red)
289 plot(b_vec, default3(1, :, 1), 'Color', [.5 0 0], '-.', 'LineWidth', 2.5);
290 xlabel('State debt', 'FontSize', 12);
291 ylabel('Default', 'FontSize', 12)
292 % Adding a legend
293 legend('\gamma = -0.5', '\gamma = 0', '\gamma = 2.0', 'Location', 'best');
294 hold off;
295
296 saveas(gcf, 'Default_risk_borrower-Q4.jpg');
297
298
299
300 figure;
301 hold on;
302 % Default policy function for risk-loving lender (darker green)
303 plot(b_vec, default4(1, :, 1), 'Color', [0 1 0], 'LineWidth', 2.5);
304 % Default policy function for risk-neutral lender
305 plot(b_vec, default5(1, :, 1), 'Color', [0 .75 0], '--', 'LineWidth', 2.5);
306 % Default policy function for risk-averse lender (lighter green)
307 plot(b_vec, default6(1, :, 1), 'Color', [0 .5 0], '-.', 'LineWidth', 2.5);
308 xlabel('State debt', 'FontSize', 12);
309 ylabel('Default', 'FontSize', 12);
310 % Adding a legend
311 legend('\sigma = .5', '\sigma = 1.0', '\sigma = 2.0', 'Location', 'best');
312 hold off;
313
314 saveas(gcf, 'Default_risk_lender-Q4.jpg');
315
316
317 figure;
318 hold on;
319 % Debt price for risk-loving lender (darker yellow)
320 plot(b_vec, debt_price1(1, :, 1), 'Color', [1 1 0], 'LineWidth', 2.5);
321 % Debt price for risk-neutral lender
322 plot(b_vec, debt_price2(1, :, 1), 'Color', [.75 .75 0], '--', 'LineWidth', 2.5);
323 % Debt price for risk-averse lender (lighter yellow)
324 plot(b_vec, debt_price3(1, :, 1), 'Color', [.5 .5 0], '-.', 'LineWidth', 2.5);
325 xlabel('Choice debt', 'FontSize', 12);
326 ylabel('Debt Price', 'FontSize', 12);
327 % Adding a legend
328 legend('\sigma = .5', '\sigma = 1.0', '\sigma = 2.0', 'Location', 'best');
329 hold off;
330
331 saveas(gcf, 'Debt_price_risk_lender-Q4.jpg');
332
333
334
335 figure;
336 hold on;
337 % Default policy function for low default cost (darker blue)
338 plot(b_vec, default7(1, :, 1), 'Color', [0 0 1], 'LineWidth', 2.5);

```

```

339 % Default policy function for medium default cost
340 plot(b_vec, default8(1, :, 1), 'Color', [0 0 .75], '--', 'LineWidth', 2.5);
341 % Default policy function for high default cost (lighter blue)
342 plot(b_vec, default9(1, :, 1), 'Color', [0 0 .5], '-.', 'LineWidth', 2.5);
343 xlabel('State debt', 'FontSize', 12);
344 ylabel('Default', 'FontSize', 12);
345 % Adding a legend
346 legend('\phi = .22', '\phi = .45', '\phi = .66', 'Location', 'best');
347 hold off;
348
349 saveas(gcf, 'Default_cost-Q4.jpg');

```