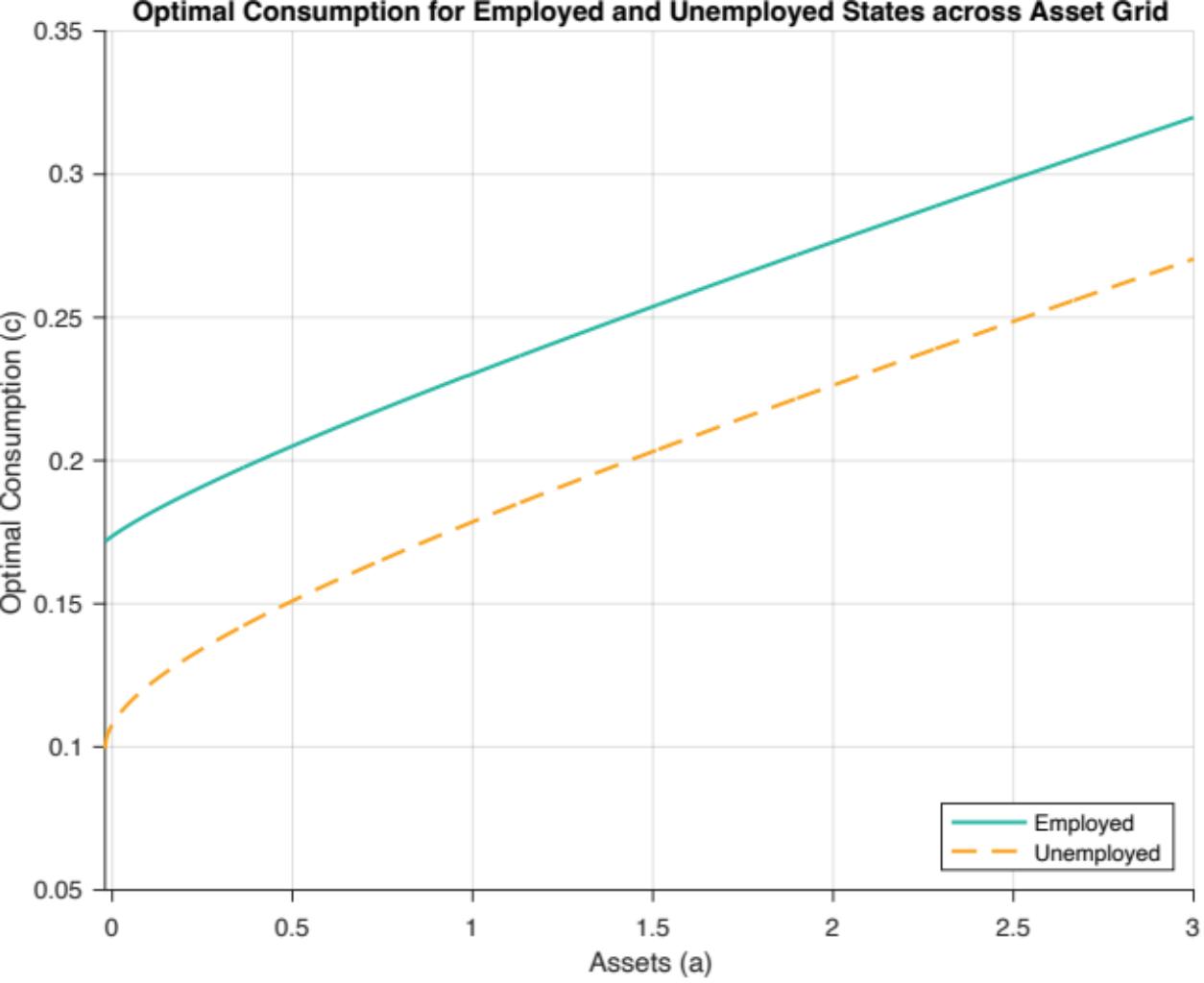


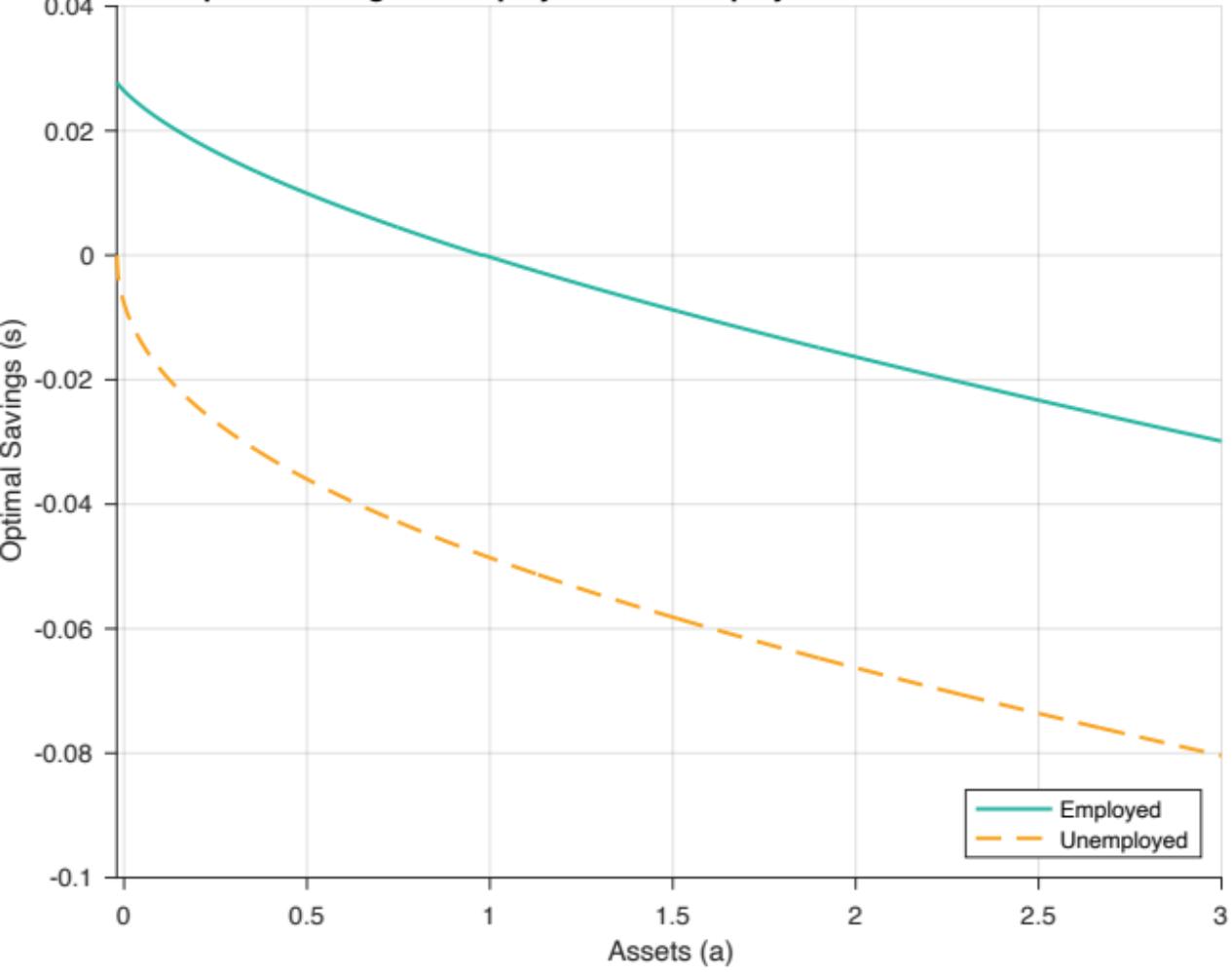
ECON 202A; problem set 3 coding

1. We see that the gap in consumption between the employed & unemployed states decreases as assets increase. This difference reflects that those in the unemployed state have a higher marginal utility from increased assets due to their income constraint. In regard to levels, employed individuals consume more at all asset levels, reflecting their higher income (& thus ability) to consume more.
2. We see the gap in savings between the employed & unemployed states increases as assets increase. This difference reflects the agent's preferences to consumption smooth, as unemployed individuals factor in future employment, a state in which they can save more. The converse is true for employed individuals. This consumption smoothing behavior also explains why savings decrease in assets.
3. We see the gap in the value function between employed & unemployed states decreases as assets increase. This reflects the ability of unemployed individuals having a higher marginal value from increased assets, since they are credit constrained.

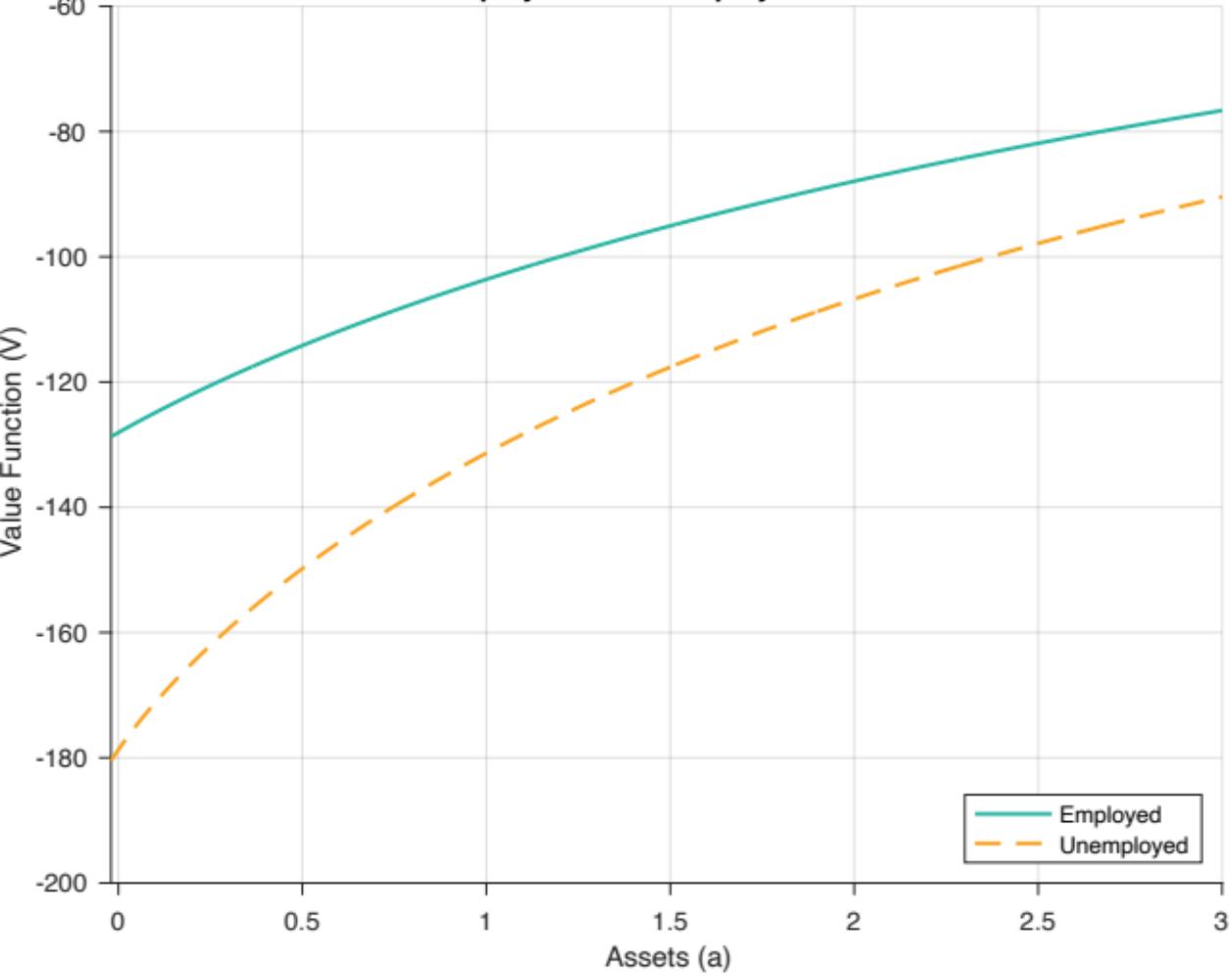
4. The Huggett model suggests that agents have a strong desire to smooth consumption, as seen from part 2. Additionally, it suggests that at most asset levels, agents should not save even if they are employed (not save at all if they are unemployed). Overall, the model predicts a low aggregate savings rate in the economy. Downstream, we would expect low capital investment.



Optimal Savings for Employed and Unemployed States across Asset Grid



Value function for Employed and Unemployed States across Asset Grid



```

problem_sets/problem_set_3/problem_set_3_code.m
1 %
2 -----
3 Purpose: Coding part of problem set 3
4 Created: Nico Rotundo 2024-11-13
5
6 %}
7
8 %
9 -----
10 Define parameters
11
12 %}
13 clear all;
14
15 % Relative risk aversion coefficient
16 sigma = 2;
17
18 % Income
19 z_e = 0.2;
20 z_u = 0.1;
21 z = [z_e, z_u];
22
23 % Transition rates
24 lambda_e = 0.03;
25 lambda_u = 0.02;
26 lambda = [lambda_e, lambda_u];
27
28 % Interest rate
29 r = 0.03;
30
31 % Discount rate
32 rho = 0.05;
33 %
34 %
35 -----
36 Define asset grid
37
38 %
39 % Borrowing constraint minimum
40 a_min = -0.02;
41
42 % Borrowing constraint maximum
43 a_max = 3;
44
45 % Number of grid points
46 num_points = 1000;
47
48 % Define grid (slide 40)
49 a_grid = linspace(a_min, a_max, num_points)';
50
51 % Step size
52 delta_a = (a_max - a_min)/(num_points-1);
53
54 % Define double length a and z vectors
55 a_grid_expanded = [a_grid, a_grid];
56 z_matrix_expanded = ones(num_points, 1) * z;
57
58 % Max iterations
59 max_iterations = 500;
60
61 %
62 %
63 Define utility function and its derivative
64
65 %
66 %
67 % Utility function (CRRA), handling vector inputs
68 U = @(c) c.^ (1 - sigma)) / (1 - sigma);
69
70 % Derivative of utility, handling vector inputs
71 U_prime = @(c) c.^(-sigma);
72
73 % Inverse of derivative of utility
74 U_prime_inv = @(Vp) (Vp).^( -1 / sigma);
75
76 %
77 %
78 % Initialize value functions, consumption policies, and block matrix with transition probabilities
79
80 %
81 %
82 % Combined consumption for employed and unemployed states
83 % Column 1 for employed, Row 2 for unemployed
84 c = zeros(num_points, 2);
85
86
87 % Initialize arrays for forward finite difference approximation for V'_i,j
88 % Column 1 for employed, Row 2 for unemployed
89 V_prime_forward = zeros(num_points,2);
90
91 % Initialize arrays for forward finite difference approximation for V'_i,j
92 % Column 1 for employed, Row 2 for unemployed
93 V_prime_backward = zeros(num_points,2);
94
95 % Construct block matrix with transition probabilities (slide 41)
96 A = [-speye(num_points)*lambda(1), speye(num_points)*lambda(1);
97         speye(num_points)*lambda(2), -speye(num_points)*lambda(2)
98     ];
99
100 %
101 %
102 Compute optimal savings using both the forward and backward difference approximations, and use the approximation for
103 V'_{i, j} from the slides
104
105 %
106 %
107 % % Initial guess (slides 39 and 40)
108
109 % Employed column
110 V_0(:,1) = U(z(1) + r * a_grid)/rho;
111
112 % Unemployed column
113 V_0(:,2) = U(z(2) + r * a_grid)/rho;
114
115 % Set initial approximation of value function
116 v_approximation = V_0;
117
118 % Iterate from 1 to max_iterations
119 for i = 1:max_iterations
120
121     % Set value function equal to current approximation of value function
122     V = v_approximation;
123
124     % Iterate over each employment state (1 = unemployed, 2 = employed)
125     for j = 1:2
126
127         % Value function finite difference approximations (slide 32)
128
129         % Forward difference for state j
130         V_prime_forward(1:num_points-1, j) = (V(2:num_points, j) - V(1:num_points-1, j)) / delta_a;
131
132         % State constraint a <= a_max (slide 38)
133         V_prime_forward(num_points, j) = U_prime(z(j) + r * a_max);
134
135         % Backward difference for state j
136         V_prime_backward(2:num_points, j) = (V(2:num_points, j) - V(1:num_points-1, j)) / delta_a;
137
138         % Borrowing constraint (slide 38)
139         V_prime_backward(1, j) = U_prime(z(j) + r * a_min);
140
141         % Consumption finite difference approximations for state j (hw page 1)
142         c_forward(:, j) = U_prime_inv(V_prime_forward(:, j));
143         c_backward(:, j) = U_prime_inv(V_prime_backward(:, j));
144         c_0(:, j) = z(j) + r * a_grid;
145
146         % Savings finite difference approximations for state j (slide 32)
147         savings_forward(:, j) = z(j) + r * a_grid - c_forward(:, j);
148         savings_backward(:, j) = z(j) + r * a_grid - c_backward(:, j);
149
150         % Derivative of value function at steady state for state j
151         V_prime_bar(:, j) = U_prime(c_0(:, j));
152
153         % Upwind scheme to select V'_i,j for state j (slide 32)
154         V_prime_upwind(:, j) = V_prime_forward(:, j).* (savings_forward(:, j) > 0) + ...
155             V_prime_backward(:, j).* (savings_backward(:, j) < 0) + ...
156             V_prime_bar(:, j).* (1 - (savings_forward(:, j) > 0) - (savings_backward(:, j) < 0));
157
158         % Optimal consumption for state j
159         c(:, j) = U_prime_inv(V_prime_upwind(:, j));
160
161         % Maximized utility for state j
162         u_optimal(:, j) = U(c(:, j));
163
164     end
165
166     % Construct matrices for each state (slide 36)
167     x_matrix = -min(savings_backward, 0) / delta_a;
168     y_matrix = -max(savings_forward, 0) / delta_a + min(savings_backward, 0) / delta_a;
169     z_matrix = max(savings_forward, 0) / delta_a;
170
171     % Construct S^n D^n matrix (slide 45)
172
173     % Employed
174     S_D_employed = spdiags(y_matrix(:, 1), 0, num_points, num_points) + ...
175         spdiags(x_matrix(2:num_points, 1), -1, num_points, num_points) + ...
176         spdiags([0; z_matrix(1:num_points-1, 1)], 1, num_points, num_points);
177
178     % Unemployed
179     S_D_unemployed = spdiags(y_matrix(:, 2), 0, num_points, num_points) + ...
180         spdiags(x_matrix(2:num_points, 2), -1, num_points, num_points) + ...
181         spdiags([0; z_matrix(1:num_points-1, 2)], 1, num_points, num_points);
182
183     % N-step transition matrix (slide 45)
184     P = [S_D_employed, sparse(num_points, num_points); sparse(num_points, num_points), S_D_unemployed] + A;
185
186     % Construct discretized Bellman operator
187     B = (rho + 1/num_points) * speye(2 * num_points) - P;
188
189     % Stack optimal utility and value function matrices for both states
190     u_optimal_stacked = [u_optimal(:, 1); u_optimal(:, 2)];
191     V_stacked = [V(:, 1); V(:, 2)];
192
193     % Solve system of equations
194     b = u_optimal_stacked + V_stacked / num_points;
195     V_stacked = B \ b;
196
197     % Reshape V_stacked back to its original structure
198     V = [V_stacked(1:num_points), V_stacked(num_points+1:2*num_points)];
199
200     % Update approximation for the next iteration
201     v_approximation = V;
202
203 end
204
205 % Optimal savings (hw page 1)
206 s_optimal = z_matrix_expanded + r .* a_grid_expanded - c;
207
208 % Question 1: Plot optimal consumption
209 figure;
210 plot(a_grid, c(:, 1), 'b-', 'LineWidth', 1.5, 'DisplayName', 'Employed', 'Color', [41/255, 182/255, 164/255]);
211 hold on;
212
213 plot(a_grid, c(:, 2), 'r--', 'LineWidth', 1.5, 'DisplayName', 'Unemployed', 'Color', [250/255, 165/255, 35/255]);
214 xlabel('Assets (a)');
215 xlim([a_min a_max]);
216
217 ylabel('Optimal Consumption (c)');
218
219 legend('Employed', 'Unemployed', 'Location', 'southeast');
220
221 title('Optimal Consumption for Employed and Unemployed States across Asset Grid');
222
223 set(gca, 'TickDir', 'out');
224 box off;
225
226 grid on;
227 hold off;
228
229 % Export graph
230 exportgraphics(gcf, '/Users/nicorotundo/Documents/GitHub/DynamicProgramming2024/problem_sets/problem_set_3/output/question_1.pdf', 'ContentType', 'vector');
231
232 % Question 2: Plot optimal savings
233 figure;
234 plot(a_grid, s_optimal(:, 1), 'b-', 'LineWidth', 1.5, 'DisplayName', 'Employed', 'Color', [41/255, 182/255, 164/255]);
235 hold on;
236
237 plot(a_grid, s_optimal(:, 2), 'r--', 'LineWidth', 1.5, 'DisplayName', 'Unemployed', 'Color', [250/255, 165/255, 35/255]);
238 xlabel('Assets (a)');
239 xlim([a_min a_max]);
240
241 ylabel('Optimal Savings (s)');
242
243 legend('Employed', 'Unemployed', 'Location', 'southeast');
244
245 title('Optimal Savings for Employed and Unemployed States across Asset Grid');
246
247 set(gca, 'TickDir', 'out');
248 box off;
249
250 grid on;
251 hold off;
252
253 % Export graph
254 exportgraphics(gcf, '/Users/nicorotundo/Documents/GitHub/DynamicProgramming2024/problem_sets/problem_set_3/output/question_2.pdf', 'ContentType', 'vector');
255
256
257 % Question 3: Plot value function
258 figure;
259 plot(a_grid, V(:, 1), 'b-', 'LineWidth', 1.5, 'DisplayName', 'Employed', 'Color', [41/255, 182/255, 164/255]);
260 hold on;
261
262 plot(a_grid, V(:, 2), 'r--', 'LineWidth', 1.5, 'DisplayName', 'Unemployed', 'Color', [250/255, 165/255, 35/255]);
263 xlabel('Assets (a)');
264 xlim([a_min a_max]);
265
266 ylabel('Value Function (V)');
267
268 legend('Employed', 'Unemployed', 'Location', 'southeast');
269
270 title('Value function for Employed and Unemployed States across Asset Grid');
271
272 set(gca, 'TickDir', 'out');
273 box off;
274
275 grid on;
276 hold off;
277
278 % Export graph
279 exportgraphics(gcf, '/Users/nicorotundo/Documents/GitHub/DynamicProgramming2024/problem_sets/problem_set_3/output/question_3.pdf', 'ContentType', 'vector');
280
281
282
283
284
285
286

```