

problem\_sets/problem\_set\_5/problem\_set\_5\_question\_1\_2.m

```
1 %{\n2\n3 Purpose: Coding part of problem set 5\n4 Created: Nice Retundo 2024-12\n5\n6 %}\n7\n8 %% Define parameters and initialize grids\n9\n10 %{\n11\n12 Define parameters\n13\n14 %}\n15\n16 clear all;\n17\n18 % Relative risk aversion coefficient\n19 sigma = 2;\n20\n21 % Productivity\n22 z_u = 1;\n23 z_e = 2;\n24 z = [z_u, z_e];\n25\n26 % Transition rates\n27 lambda_u = 1/3;\n28 lambda_e = 1/3;\n29 lambda = [lambda_u, lambda_e];\n30\n31 % Discount rate\n32 rho = 0.05;\n33\n34 % Capital depreciation rate\n35 delta = 0.05;\n36\n37 % Capital share in production\n38 alpha = 1/3;\n39\n40 % TFP\n41 A = .1;\n42\n43 % Interest rate for partial equilibrium part of the problem set\n44 r = 0.035;\n45\n46 %{\n47\n48 Define capital grid\n49\n50 %}\n51\n52 % Borrowing constraint minimum\n53 k_min = 0;\n54\n55 % Borrowing constraint maximum\n56 k_max = 20;\n57\n58 % Number of grid points\n59 num_points = 1000;\n60\n61 k_grid = linspace(k_min, k_max, num_points)';\n62 dk = (k_max-k_min)/(num_points-1);\n63\n64 kk = [k_grid, k_grid]; % Ix2 matrix\n65\n66 %{\n67\n68 Define utility function and its derivative\n69\n70 %}\n71\n72 % Utility function (CRRA), handling vector inputs\n73 U = @(c) (c.^(1 - sigma)) / (1 - sigma);\n74\n75 % Derivative of utility, handling vector inputs\n76 U_prime = @(c) c.^(1-sigma);\n77\n78 % Inverse of derivative of utility\n79 U_prime_inv = @(Vp) (Vp).^(1 / sigma);\n80\n81 %{\n82\n83 Firm production technology and FOCs\n84\n85 %}\n86\n87 % Production technology Y = A K^alpha L^(1-alpha)\n88 Y = @(K,L) K.^alpha*L.^(1-alpha);\n89\n90 % F.O.C.: \nw_foc = @(K,L) (1-alpha)*A*(K/L).^alpha;\n91 r_foc = @(K,L) alpha*A*(L/K).^(1-alpha);\n92 K_partial = @(L,r) L*(alpha*A/(r+delta)).^(1/(1-alpha));\n93\n94 %{\n95\n96 Tuning parameters (general)\n97\n98 %}\n99\n100 % Step size: can be arbitrarily large in implicit method\n101 Delta = 1000;\n102\n103 % The maximum number of value function iterations\n104 max_iterations_vf = 100;\n105\n106 % Tolerance for value function iterations\n107 tolerance = 10^(-6);\n108\n109 %{\n110\n111 Tuning parameters (interest rate iteration)\n112\n113 %}\n114\n115 % The maximum number of interest rate iterations\n116 num_iterations_r = 1000;\n117\n118 % Tolerance for interest rate iterations\n119 tolerance_S = 10^(-5);\n120\n121 %% Initialize matrices before iterating\n122\n123 %{\n124\n125 Define differential operator\n126\n127 %}\n128\n129 % Forward; satisfies Df=V*dVf\n130 Df = zeros(num_points, num_points);\n131 for i = 1:num_points-1\n132     Df(i,i) = -1/dk; Df(i,i+1) = 1/dk;\n133 end\n134 Df = sparse(Df);\n135\n136 % Backward; satisfies Db=V*dV\n137 Db = zeros(num_points, num_points);\n138 for i = 2:num_points\n139     Db(i,i-1) = -1/dk; Db(i,i) = 1/dk;\n140 end\n141 Db = sparse(Db);\n142\n143 %{\n144\n145 Define A-switch matrix\n146\n147 %}\n148\n149 A_switch = [speye(num_points).*(1-lambda(1)), speye(num_points).*lambda(1);\n150             speye(num_points).*lambda(2), speye(num_points).*(-lambda(2))];\n151\n152 %{\n153\n154 Calculate labor, capital, and wage levels\n155\n156 %}\n157\n158 % Labor\n159 L = (z_e*lambda_u + z_u*lambda_e)/(lambda_e+lambda_u);\n160\n161 % Capital\n162 K = K_partial(L,r);\n163\n164 % Wage\n165 w = w_foc(K,L);\n166\n167 %{\n168\n169 Define initial guesses\n170\n171 %}\n172\n173 % Guess for initial value of the interest rate\n174 r_0_guess = 0.03;\n175\n176 % Set bounds on interest rate\n177 r_min = 0.01;\n178 r_max = 0.04;\n179\n180 % Define the number of points * 2 matrix\n181 z = ones(num_points, 1)*z;\n182\n183 % Initial guess for value function\n184 V_0 = U(w.*z + r.*kk)./rho;\n185 V = V_0;\n186\n187 % Value function iteration\n188\n189 % Loop over number of iterations for the value function\n190 for n=1:max_iterations_vf\n191\n192     % Derivative of the forward value function\n193     dVf = Df*V;\n194\n195     % Derivative of the backward value function\n196     dVb = Db*V;\n197\n198     % Boundary condition on backwards value function i.e., the borrowing constraint; a=a_min\n199     dVb(1,:) = U_prime(w.*z(1,:) + r.*kk(1,:));\n200\n201     % Boundary condition on forward value function; a=a_max\n202     dVf(end,:) = U_prime(w.*z(end,:) + r.*kk(end,:));\n203\n204     % Indicator whether value function is concave; for stability purposes\n205     I_concave = dVb > dVf;\n206\n207     % Compute optimal consumption using forward derivative\n208     cf = U_prime_inv(dVf);\n209\n210     % Compute optimal consumption using backward derivative\n211     cb = U_prime_inv(dVb);\n212\n213     % Compute optimal savings using forward derivative\n214     sf = w.*z + r.*kk - cf;\n215\n216     % Compute optimal savings using backward derivative\n217     sb = w.*z + r.*kk - cb;\n218\n219     % Upwind scheme\n220     If = sf>0;\n221     Ib = sb<0;\n222     I0 = 1-If-Ib;\n223     dV0 = U_prime(w.*z + r.*kk); % If sf<=0==sb, set s=0\n224\n225     dV_upwind = If.*dVf + Ib.*dVb + I0.*dV0;\n226\n227     c = U_prime_inv(dV_upwind);\n228\n229     % Update value function\n230     V_stacked = V(:);\n231\n232     % Update consumption function\n233     c_stacked = c(:);\n234\n235     % A = SD\n236     SD_u = spdiags(If(:),1)*sf(:), 0, num_points, num_points)*dVf + spdiags(Ib(:),1)*sb(:), 1, 0, num_points, num_points)*Db;\n237     SD_e = spdiags(If(:),2)*sf(:), 0, num_points, num_points)*dVf + spdiags(Ib(:),2)*sb(:), 2, 0, num_points, num_points)*Db;\n238     SD = (SD_u, sparse(num_points, num_points));\n239     sparse(num_points, num_points), SD_e);\n240\n241     % P = A + A_switch\n242     P = SD + A_switch;\n243\n244     % B = [(rho + 1/Delta)*I - P]\n245     B = (rho + 1/Delta)*speye(2*num_points) - P;\n246\n247     % b = u(c) + 1/Delta*V\n248     b = U(c_stacked) + (1/Delta)*V_stacked;\n249\n250     % V = B\\b;\n251     V_update = B\\b;\n252     V_change = V_update - V_stacked;\n253     V = reshape(V_update, num_points, 2);\n254\n255     % Convergence criterion\n256     dist(n) = max(abs(V_change));\n257\n258     if dist(n)<tolerance\n259         disp('Value function converged. Iteration = ')\n260         disp(n)\n261         break\n262     end\n263 end\n264\n265 %{\n266\n267 %% KF Equation\n268\n269 %}\n270\n271 % Solve for @gdot=P*g\n272 PT = P';\n273 PT_eigs = PT;\n274 gdot_stacked = zeros(2*num_points,1);\n275\n276 % Fix one value to obtain a non-singular matrix, otherwise matrix is singular\n277 i_fix = 1;\n278 gdot_stacked(i_fix)=1;\n279\n280 row_fix = [zeros(1,i_fix-1),1,zeros(1,2*num_points-i_fix)];\n281 PT(i_fix,:) = row_fix;\n282\n283 g_stacked = PT\\gdot_stacked;\n284\n285 % Normalization\n286 g_sum = g_stacked*ones(2*num_points,1)*dk;\n287 g_stacked = g_stacked./g_sum;\n288\n289 % Reshape\n290 gg = reshape(g_stacked, num_points, 2);\n291\n292 % Solve KF equation\n293 [g_stacked_eigs, eigval] = eigs(PT_eigs, 1, 0);\n294 g_sum_eigs = g_stacked_eigs*ones(2*num_points,1)*dk;\n295 g_stacked_eigs = g_stacked_eigs./g_sum_eigs;\n296\n297 %% Plots\n298\n299 %{\n300\n301 Question 1: Optimal consumption\n302\n303 %}\n304\n305 % Initialize plot\n306 set(gcf, 'FontSize', 18)\n307\n308 % Employed\n309 plot(k_grid, c(:,2), 'LineWidth', 2, 'LineStyle', '-', 'Color', [41/255, 182/255, 164/255])\n310 hold on\n311\n312 % Unemployed\n313 plot(k_grid, c(:,1), 'LineWidth', 2, 'LineStyle', '-', 'Color', [250/255, 165/255, 35/255])\n314\n315 hold off\n316\n317 grid\n318 xlabel('Capital, k', 'FontSize', 14)\n319\n320 set(gcf, 'TickDir', 'out');\n321 box off;\n322\n323 ylabel('Consumption, c_j(k)', 'FontSize', 14)\n324\n325 xlim([k_min k_max])\n326\n327 legend(sprintf('Employed'), ...)\n328     sprintf('Unemployed'), ...)\n329     sprintf('r=%4f', r), 'Location', 'best', 'FontSize', 14)\n330\n331 % Export graph\n332 exportgraphics(gcf, '/Users/nicorotundo/Documents/GitHub/DynamicProgramming2024/problem_sets/problem_set_5/output/question_1_consumption.pdf', 'ContentType', 'vector');\n333\n334 %{\n335\n336 Question 1: Optimal savings\n337\n338 %}\n339\n340 % Calculate savings\n341 kdot = w.*z + r.*kk - c;\n342\n343 % Initialize plot\n344 set(gcf, 'FontSize', 18)\n345\n346 % Employed\n347 plot(k_grid, kdot(:,2), 'LineWidth', 2, 'LineStyle', '-', 'Color', [41/255, 182/255, 164/255])\n348 hold on\n349\n350 % Unemployed\n351 plot(k_grid, kdot(:,1), 'LineWidth', 2, 'LineStyle', '-', 'Color', [250/255, 165/255, 35/255])\n352\n353 hold off\n354\n355 grid\n356 xlabel('Capital, k', 'FontSize', 14)\n357\n358 ylabel('Savings, s_j(k)', 'FontSize', 14)\n359\n360 set(gcf, 'TickDir', 'out');\n361 box off;\n362\n363 xlim([k_min k_max])\n364\n365 legend(sprintf('Employed'), ...)\n366     sprintf('Unemployed'), ...)\n367     sprintf('r=%4f', r), 'Location', 'best', 'FontSize', 14)\n368\n369 % Export graph\n370 exportgraphics(gcf, '/Users/nicorotundo/Documents/GitHub/DynamicProgramming2024/problem_sets/problem_set_5/output/question_1_savings.pdf', 'ContentType', 'vector');\n371\n372 %{\n373\n374 Question 1: Value function\n375\n376 %}\n377\n378 % Initialize plot\n379 set(gcf, 'FontSize', 18)\n380\n381 % Employed\n382 plot(k_grid, V(:,2), 'LineWidth', 2, 'LineStyle', '-', 'Color', [41/255, 182/255, 164/255])\n383 hold on\n384\n385 % Unemployed\n386 plot(k_grid, V(:,1), 'LineWidth', 2, 'LineStyle', '-', 'Color', [250/255, 165/255, 35/255])\n387\n388 hold off\n389\n390 grid\n391 xlabel('Capital, k', 'FontSize', 14)\n392\n393 ylabel('Value Function, V_j(k)', 'FontSize', 14)\n394\n395 set(gcf, 'TickDir', 'out');\n396 box off;\n397\n398 xlim([k_min k_max])\n399\n400 legend(sprintf('Employed'), ...)\n401     sprintf('Unemployed'), ...)\n402     sprintf('r=%4f', r), 'Location', 'best', 'FontSize', 14)\n403\n404 % Export graph\n405 exportgraphics(gcf, '/Users/nicorotundo/Documents/GitHub/DynamicProgramming2024/problem_sets/problem_set_5/output/question_1_value_function.pdf', 'ContentType', 'vector');\n406\n407 %{\n408\n409 Question 2: Stationary distribution\n410\n411 %}\n412\n413 % Initialize plot\n414 set(gcf, 'FontSize', 18)\n415\n416 % Employed\n417 plot(k_grid, gg(:,2), 'LineWidth', 2, 'LineStyle', '-', 'Color', [41/255, 182/255, 164/255])\n418 hold on\n419\n420 % Unemployed\n421 plot(k_grid, gg(:,1), 'LineWidth', 2, 'LineStyle', '-', 'Color', [250/255, 165/255, 35/255])\n422\n423 hold off\n424\n425 grid\n426 xlabel('Capital, k', 'FontSize', 14)\n427\n428 ylabel('Densities, g_j(k)', 'FontSize', 14)\n429\n430 set(gcf, 'TickDir', 'out');\n431 box off;\n432\n433 xlim([-0.1 1])\n434\n435 legend(sprintf('Employed'), ...)\n436     sprintf('Unemployed'), ...)\n437     sprintf('r=%4f', r), 'Location', 'best', 'FontSize', 14)\n438\n439 % Export graph\n440 exportgraphics(gcf, '/Users/nicorotundo/Documents/GitHub/DynamicProgramming2024/problem_sets/problem_set_5/output/question_2.pdf', 'ContentType', 'vector');
```