problem_sets/problem_set_3/problem_set_3_code.m

```matlab
%{
------------------------------------------------------------------------------------------------------------------------
Purpose: Coding part of problem set 3
Created: Nico Rotundo 2024-11-13
------------------------------------------------------------------------------------------------------------------------
%}

%{
------------------------------------------------------------------------------------------------------------------------
Define parameters
------------------------------------------------------------------------------------------------------------------------
%}
clear all;

% Relative risk aversion coefficient
sigma = 2;

% Income
z_e = 0.2;
z_u = 0.1;
z = [z_e, z_u];

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

% Interest rate
r = 0.03;

% Discount rate
rho = 0.05;

%{
------------------------------------------------------------------------------------------------------------------------
Define asset grid
------------------------------------------------------------------------------------------------------------------------
%}

% Borrowing constraint minimum
a_min = -0.02;

% Borrowing constraint maximum
a_max = 3;

% Number of grid points
num_points = 1000;

% Define grid (slide 40)
a_grid = linspace(a_min, a_max, num_points)';

% Step size
delta_a = (a_max - a_min)/(num_points-1);

% Define double length a and z vectors
a_grid_expanded = [a_grid, a_grid];
z_matrix_expanded = ones(num_points, 1) * z;

% Max iterations
max_iterations = 500;

%{
------------------------------------------------------------------------------------------------------------------------
Define utility function and its derivative
------------------------------------------------------------------------------------------------------------------------
%}

% Utility function (CRRA), handling vector inputs
U = @(c) (c.^(1 - sigma)) / (1 - sigma);

% Derivative of utility, handling vector inputs
U_prime = @(c) c.^(-sigma);

% Inverse of derivative of utility
U_prime_inv = @(Vp) (Vp).^(-1 / sigma);

%{
------------------------------------------------------------------------------------------------------------------------
% Initialize value functions, consumption policies, and block matrix with transition probabilities
------------------------------------------------------------------------------------------------------------------------
%}

% Combined consumption for employed and unemployed states
% Column 1 for employed, Row 2 for unemployed
c = zeros(num_points, 2);

% Initialize arrays for forward finite difference approximation for V'_i,j
% Column 1 for employed, Row 2 for unemployed
V_prime_forward = zeros(num_points,2);

% Initialize arrays for forward finite difference approximation for V'_i,j
% Column 1 for employed, Row 2 for unemployed
V_prime_backward = zeros(num_points,2);

% Construct block matrix with transition probabilities (slide 41)
A = [-speye(num_points)*lambda(1), speye(num_points)*lambda(1);
        speye(num_points)*lambda(2),-speye(num_points)*lambda(2)
        ];

%{
------------------------------------------------------------------------------------------------------------------------
Compute optimal savings using both the forward and backward difference approximations, and use the approximation for
V'_{i, j} from the slides
------------------------------------------------------------------------------------------------------------------------
%}

% % Initial guess (slides 39 and 40)

% Employed column
V_0(:,1) = U(z(1) + r * a_grid)/rho;

% Unemployed column
V_0(:,2) = U(z(2) + r * a_grid)/rho;

% Set initial approximation of value function
v_approximation = V_0;

% Iterate from 1 to max_iterations
for i = 1:max_iterations

    % Set value function equal to current approximation of value function
    V = v_approximation;

    % Iterate over each employment state (1 = unemployed, 2 = employed)
    for j = 1:2

        % % Value function finite difference approximations (slide 32)

        % Forward difference for state j
        V_prime_forward(1:num_points-1, j) = (V(2:num_points, j) - V(1:num_points-1, j)) / delta_a;

        % State constraint a <= a_max (slide 38)
        V_prime_forward(num_points, j) = U_prime(z(j) + r * a_max);

        % Backward difference for state j
        V_prime_backward(2:num_points, j) = (V(2:num_points, j) - V(1:num_points-1, j)) / delta_a;

        % Borrowing constraint (slide 38)
        V_prime_backward(1, j) = U_prime(z(j) + r * a_min);

        % Consumption finite difference approximations for state j (hw page 1)
        c_forward(:, j) = U_prime_inv(V_prime_forward(:, j));
        c_backward(:, j) = U_prime_inv(V_prime_backward(:, j));
        c_0(:, j) = z(j) + r * a_grid;

        % Savings finite difference approximations for state j (slide 32)
        savings_forward(:, j) = z(j) + r * a_grid - c_forward(:, j);
        savings_backward(:, j) = z(j) + r * a_grid - c_backward(:, j);

        % Derivative of value function at steady state for state j
        V_prime_bar(:, j) = U_prime(c_0(:, j));

        % Upwind scheme to select V'_i,j for state j (slide 32)
        V_prime_upwind(:, j) = V_prime_forward(:, j) .* (savings_forward(:, j) > 0) + ...
                               V_prime_backward(:, j) .* (savings_backward(:, j) < 0) + ...
                               V_prime_bar(:, j) .* (1 - (savings_forward(:, j) > 0) - (savings_backward(:, j) < 0));

        % Optimal consumption for state j
        c(:, j) = U_prime_inv(V_prime_upwind(:, j));

        % Maximized utility for state j
        u_optimal(:, j) = U(c(:, j));
    end

    % Construct matrices for each state (slide 36)
    x_matrix = -min(savings_backward, 0) / delta_a;
    y_matrix = -max(savings_forward, 0) / delta_a + min(savings_backward, 0) / delta_a;
    z_matrix = max(savings_forward, 0) / delta_a;

    % % Construct S^n D^n matrix (slide 45)

    % Employed
    S_D_employed = spdiags(y_matrix(:, 1), 0, num_points, num_points) + ...
                   spdiags(x_matrix(2:num_points, 1), -1, num_points, num_points) + ...
                   spdiags([0; z_matrix(1:num_points-1, 1)], 1, num_points, num_points);

    % Unemployed
    S_D_unemployed = spdiags(y_matrix(:, 2), 0, num_points, num_points) + ...
                     spdiags(x_matrix(2:num_points, 2), -1, num_points, num_points) + ...
                     spdiags([0; z_matrix(1:num_points-1, 2)], 1, num_points, num_points);

    % N-step transition matrix (slide 45)
    P = [S_D_employed, sparse(num_points, num_points); sparse(num_points, num_points), S_D_unemployed] + A;

    % Construct discretized Bellman operator
    B = (rho + 1/num_points) * speye(2 * num_points) - P;

    % Stack optimal utility and value function matrices for both states
    u_optimal_stacked = [u_optimal(:, 1); u_optimal(:, 2)];
    V_stacked = [V(:, 1); V(:, 2)];

    % Solve system of equations
    b = u_optimal_stacked + V_stacked / num_points;
    V_stacked = B \ b;

    % Reshape V_stacked back to its original structure
    V = [V_stacked(1:num_points), V_stacked(num_points+1:2*num_points)];

    % Update approximation for the next iteration
    v_approximation = V;
end


% Optimal savings (hw page 1)
s_optimal = z_matrix_expanded + r .* a_grid_expanded - c;

% Question 1: Plot optimal consumption
figure;
    plot(a_grid, c(:, 1), 'b-', 'LineWidth', 1.5, 'DisplayName', 'Employed', 'Color', [41/255, 182/255, 164/255]);
    hold on;

    plot(a_grid, c(:, 2), 'r--', 'LineWidth', 1.5, 'DisplayName', 'Unemployed', 'Color', [250/255, 165/255, 35/255]);

    xlabel('Assets (a)');
    xlim([a_min a_max]);

    ylabel('Optimal Consumption (c)');

    legend('Employed', 'Unemployed', 'Location', 'southeast');

    title('Optimal Consumption for Employed and Unemployed States across Asset Grid');

    set(gca, 'TickDir', 'out');
    box off;

    grid on;
    hold off;

% Export graph
exportgraphics(gcf, '/Users/nicorotundo/Documents/GitHub/DynamicProgramming2024/problem_sets/problem_set_3/output/question_1.pdf', 'ContentType', 'vector');

% Question 2: Plot optimal savings
figure;
    plot(a_grid, s_optimal(:, 1), 'b-', 'LineWidth', 1.5, 'DisplayName', 'Employed', 'Color', [41/255, 182/255, 164/255]);
    hold on;

    plot(a_grid, s_optimal(:, 2), 'r--', 'LineWidth', 1.5, 'DisplayName', 'Unemployed', 'Color', [250/255, 165/255, 35/255]);

    xlabel('Assets (a)');
    xlim([a_min a_max]);

    ylabel('Optimal Savings (s)');

    legend('Employed', 'Unemployed', 'Location', 'southeast');

    title('Optimal Savings for Employed and Unemployed States across Asset Grid');

    set(gca, 'TickDir', 'out');
    box off;

    grid on;
    hold off;

% Export graph
exportgraphics(gcf, '/Users/nicorotundo/Documents/GitHub/DynamicProgramming2024/problem_sets/problem_set_3/output/question_2.pdf', 'ContentType', 'vector');


% Question 3: Plot value function
figure;
    plot(a_grid, V(:, 1), 'b-', 'LineWidth', 1.5, 'DisplayName', 'Employed', 'Color', [41/255, 182/255, 164/255]);
    hold on;

    plot(a_grid, V(:, 2), 'r--', 'LineWidth', 1.5, 'DisplayName', 'Unemployed', 'Color', [250/255, 165/255, 35/255]);

    xlabel('Assets (a)');
    xlim([a_min a_max]);

    ylabel('Value Function (V)');

    legend('Employed', 'Unemployed', 'Location', 'southeast');

    title('Value function for Employed and Unemployed States across Asset Grid');

    set(gca, 'TickDir', 'out');
    box off;

    grid on;
    hold off;

% Export graph
exportgraphics(gcf, '/Users/nicorotundo/Documents/GitHub/DynamicProgramming2024/problem_sets/problem_set_3/output/question_3.pdf', 'ContentType', 'vector');
```