% Parameters

num\_nodes = 10;

num\_elements = num\_nodes - 1;

L = 1; % Domain length

Pe = [0, 1, 7, 7000]; % Peclet numbers

alpha = 0; % Given in problem statement

u = 1; % Velocity, assumed to be 1 for this example

% Discretization

x = linspace(0, L, num\_nodes);

h = mean(diff(x)); % Element size

% Shape functions for linear elements

N = @(xi) [1 - xi, xi];

dN\_dxi = [-1, 1]; % Derivatives of the shape functions

% Initialization of global matrix and vector

K = zeros(num\_nodes);

f = zeros(num\_nodes, 1);

% Assembly of global matrix and vector

for i = 1:num\_elements

% Local to global mapping

nodes = [i, i+1];

x\_local = x(nodes);

% Element stiffness matrix and load vector

Ke = (u/h \* dN\_dxi' \* N(0.5) + (u\*h/2) \* dN\_dxi' \* dN\_dxi) \* h/2;

fe = zeros(2,1); % Assuming source term is zero

% Global assembly

K(nodes,nodes) = K(nodes,nodes) + Ke;

f(nodes) = f(nodes) + fe;

end

% Apply boundary conditions

K(1,:) = 0;

K(1,1) = 1;

f(1) = alpha;

% Solution for different Peclet numbers

phi = cell(length(Pe), 1);

for i = 1:length(Pe)

% Adjust stiffness matrix for Peclet number

K\_adj = K \* (Pe(i) \* h / u);

% Solve the linear system

phi{i} = K\_adj \ f;

end

% Plot results

for i = 1:length(Pe)

figure;

plot(x, phi{i}, '-o');

title(sprintf('Convection Diffusion for Peclet number(Pe) = %d', Pe(i)));

xlabel('Domain Length');

ylabel('Scalar variable (\phi)');

end

% Parameters

num\_nodes = 10;

num\_elements = num\_nodes - 1;

L = 1; % Domain length

Pe = [0, 1, 7, 7000]; % Peclet numbers

alpha = 0; % Given in problem statement

u = 1; % Velocity, assumed to be 1 for this example

% Discretization

x = linspace(0, L, num\_nodes);

h = mean(diff(x)); % Element size

% Shape functions for linear elements

N = @(xi) [1 - xi, xi];

dN\_dxi = [-1, 1]; % Derivatives of the shape functions

% Initialization of global matrix and vector

K = zeros(num\_nodes);

f = zeros(num\_nodes, 1);

% Assembly of global matrix and vector

for i = 1:num\_elements

% Local to global mapping

nodes = [i, i+1];

x\_local = x(nodes);

% Element stiffness matrix and load vector

Ke = (u/h \* dN\_dxi' \* N(0.5) + u \* dN\_dxi' \* dN\_dxi) \* h/2;

fe = zeros(2,1); % Assuming source term is zero

% Additional Petrov-Galerkin term for stabilization

Pe\_local = u \* h / (2 \* k); % k is the diffusion coefficient

W = Pe\_local > 1 ? h / (2 \* u) \* (1 - 1/Pe\_local) : 0;

Ke = Ke + (u \* W \* dN\_dxi' \* dN\_dxi) \* h/2;

% Global assembly

K(nodes,nodes) = K(nodes,nodes) + Ke;

f(nodes) = f(nodes) + fe;

end

% Apply boundary conditions

K(1,:) = 0;

K(1,1) = 1;

f(1) = alpha;

% Solution for different Peclet numbers

phi\_pg = cell(length(Pe), 1);

for i = 1:length(Pe)

k = 1 / Pe(i); % Assuming diffusion coefficient is the inverse of Peclet number

% Adjust stiffness matrix for Peclet number

K\_adj = K \* (Pe(i) \* h / (2 \* k)) + K;

% Solve the linear system

phi\_pg{i} = K\_adj \ f;

end

% Plot results

for i = 1:length(Pe)

figure;

plot(x, phi\_pg{i}, '-o');

hold on;

% Add analytical solution plot if available

plot(x, analytical\_solution(x, Pe(i)), '--');

hold off;

title(sprintf('Petrov-Galerkin Solution for Peclet number(Pe) = %d', Pe(i)));

xlabel('Domain Length');

ylabel('Scalar variable (\phi)');

end

% Define the analytical solution function here based on your problem

function phi\_analytical = analytical\_solution(x, Pe)

% ... Implement the analytical solution based on your specific problem

end

% Given data

Pe = 7; % Peclet number

L = 1; % Length of the domain

u = 1; % Velocity (assuming unity for simplicity)

num\_nodes\_set = [10, 30, 50, 70]; % Different mesh sizes

% Function to calculate the analytical solution

analytical\_solution = @(x) ... % Define the analytical solution based on your problem

% Loop over different mesh sizes

for mesh\_size = num\_nodes\_set

num\_nodes = mesh\_size;

num\_elements = num\_nodes - 1;

x = linspace(0, L, num\_nodes); % Nodes coordinates

h = mean(diff(x)); % Element size

% Initialize global stiffness matrix K and load vector F

K = zeros(num\_nodes);

F = zeros(num\_nodes, 1);

% Assemble the global matrix and vector using the Petrov-Galerkin method

for i = 1:num\_elements

% Define local stiffness matrix and load vector for element i

% ...

end

% Apply boundary conditions

% ...

% Solve the linear system for nodal values

phi = K \ F;

% Plot numerical vs. analytical solutions

figure;

plot(x, phi, 'o-', x, analytical\_solution(x), '--');

legend('Petrov-Galerkin Solution', 'Analytical Solution');

title(sprintf('Mesh size: %d', mesh\_size));

xlabel('Domain Length');

ylabel('Scalar variable (\phi)');

end