```matlab
close all;clear;clc;


% Parameters
a = 0;
L = 0.01;
b = L;


[x8,w8] = lglnodes(9);
x8 = (flipud(x8)+1)/2*(b-a)+a;
[x16,w16] = lglnodes(17);
x16 = (flipud(x16)+1)/2*(b-a)+a;
[x32,w32] = lglnodes(33);
x32 = (flipud(x32)+1)/2*(b-a)+a;

% Plot the nodes for each case
figure;
hold on;
plot(x8, zeros(size(x8)), 'ro', 'MarkerSize', 8, 'DisplayName', 'N=8');   % Red circles for N=8
plot(x16, zeros(size(x16)) + 0.1, 'go', 'MarkerSize', 8, 'DisplayName', 'N=16');  % Green circles for N=16
plot(x32, zeros(size(x32)) + 0.2, 'bo', 'MarkerSize', 8, 'DisplayName', 'N=32');  % Blue circles for N=32

% Set plot properties
title('Gauss-Lobatto-Legendre Nodes for N=8, N=16, and N=32');
xlabel('x');
ylabel('artificial separation');
legend('show');
ylim([-0.1, 0.3]);  % Add some separation for clarity
grid on;
hold off;

%%

N = [8;16;32;64;128;256;512;1024;2048];
L1 = zeros(length(N),1);
L2 = zeros(length(N),1);
L3 = zeros(length(N),1);
cond_D1 = zeros(length(N), 1);
cond_D2 = zeros(length(N), 1);
cond_D3 = zeros(length(N), 1);
for i = 1:9

npts = N(i)+1;
[x, ~] = lglnodes(npts);
x = flipud(x);

% Compute the first, second, and third derivative matrices
```

```
[D, D2, D3] = derv(N(i), x);
[D, D2, D3] = scale_derivative_matrices(D,D2,D3,a,b);
x = (x+1)/2*(b-a)+a;
cond_D1(i) = cond(D);
cond_D2(i) = cond(D2);
cond_D3(i) = cond(D3);



u = sin(2 * pi * x / L);

u_nu1 = D * u;

u_exact1 = (2*pi/L)*cos(2*pi*x/L);

L1(i) = max(abs(u_nu1-u_exact1));

u_nu2 = D2*u;

u_exact2 = -(2*pi/L)^2 * sin(2*pi*x/L);

L2(i) = max(abs(u_nu2-u_exact2));

u_nu3 = D3*u;

u_exact3 = -(2*pi/L)^3 * cos(2*pi*x/L);


L3(i) = max(abs(u_nu3-u_exact3));

end


% Select only the first four values (for N = 8, 16, 32, 64)
N_small = N(1:3);
L1_small = L1(1:3);
L2_small = L2(1:3);
L3_small = L3(1:3);

% Define a function for power-law fitting (logarithmic form)
power_law_fit = @(N, L) polyfit(log(N), log(L), 1);

% Fit power-law for each derivative using the first 4 values
p1_small = power_law_fit(N_small, L1_small); % 1st derivative
p2_small = power_law_fit(N_small, L2_small); % 2nd derivative
p3_small = power_law_fit(N_small, L3_small); % 3rd derivative

% Extract exponents (slope of the fit, which corresponds to the power-law exponent)
b1_small = p1_small(1);
b2_small = p2_small(1);
b3_small = p3_small(1);
```

```
% Plot the L_inf norm and the power law fit for the first 4 values
figure;
loglog(N_small, L1_small, 'o-', 'DisplayName', 'L1 Error');
hold on;
loglog(N_small, exp(p1_small(2)) * N_small.^b1_small, '--', 'DisplayName', sprintf('Fit: ↙
N^{%.2f}', b1_small));

loglog(N_small, L2_small, 'o-', 'DisplayName', 'L2 Error');
loglog(N_small, exp(p2_small(2)) * N_small.^b2_small, '--', 'DisplayName', sprintf('Fit: ↙
N^{%.2f}', b2_small));

loglog(N_small, L3_small, 'o-', 'DisplayName', 'L3 Error');
loglog(N_small, exp(p3_small(2)) * N_small.^b3_small, '--', 'DisplayName', sprintf('Fit: ↙
N^{%.2f}', b3_small));

xlabel('N');
ylabel('L_\infty norm of absolute error');
legend('show');
title('Power-Law Fit of L_\infty Norms for Derivatives (First 3 Points)');

% Display the exponents for the first 4 values
fprintf('Power-law exponent for 1st derivative (first 3 values): %.2f\n', b1_small);
fprintf('Power-law exponent for 2nd derivative (first 3 values): %.2f\n', b2_small);
fprintf('Power-law exponent for 3rd derivative (first 3 values): %.2f\n', b3_small);

%%
% Plot the condition numbers on log-log axes
figure;
loglog(N, cond_D1, 'o-', 'DisplayName', '1st Derivative Condition Number');
hold on;
loglog(N, cond_D2, 'o-', 'DisplayName', '2nd Derivative Condition Number');
loglog(N, cond_D3, 'o-', 'DisplayName', '3rd Derivative Condition Number');
xlabel('N');
ylabel('Condition Number');
legend('show');
title('Condition Number of Spectral Differentiation Matrices');




%%



% Parameters
alpha = 1.0;     % Diffusivity
tf = 0.25;       % Final time
Nt = 1000;       % Number of time steps
Ns = [10,25,50,100];          % Number of spatial nodes (LGL nodes)
dt = tf / Nt;    % Time step size
a = 0;
```

```
b = 1;
% Initial and Boundary Conditions
initial_condition = @(x) 1 - x - (1 / pi) * sin(2 * pi * x);
time_indices = round([1, Nt/3, 2*Nt/3, Nt+1]);
time_values = time_indices * dt;

L = zeros(4,4);
for i = 1:4
% Create the spatial grid using LGL nodes
N = Ns(i);
[x,w] = lglnodes(N+1);
x = flipud(x);

% Initial condition
C = initial_condition(x);
C_at_times = zeros(N+1, 4);  % 4 columns for 4 time points
C_at_times(:, 1) = C;       % Store the initial condition

% Second derivative matrix
[~,D2,~] = derv(N, x);

[D, D2, D3] = scale_derivative_matrices(D,D2,D3,a,b);
x = (x+1)/2*(b-a)+a;

w = w/2*(b-a);

% Construct the coefficient matrix for the implicit method
A = eye(N+1) - alpha * dt * D2;

% Time-stepping loop (fully implicit scheme)
for n = 1:Nt
    % Apply boundary conditions
    C(1) = 1;
    C(end) = 0;

    % Solve linear system directly: A * C_new = C_old
    C = A \ C;

    % Store the solution at the selected time steps
    if ismember(n+1, time_indices)
        index = find(time_indices == n+1);
        C_at_times(:, index) = C;
    end
end

% Analytic solution function
analytic_solution = @(x, t) 1 - x - (1 / pi) * exp(-4 * pi^2 * t) * sin(2 * pi * x);

% Compute the analytic solution at the selected time points
C_analytic = zeros(N+1, 4);
for k = 1:4
```

```
    C_analytic(:, k) = analytic_solution(x, time_values(k));
end


for j = 1:4
    L(j,i) = max(abs(C_analytic(:,j)-C_at_times(:,j)));
end

end


%%
figure;
hold on;
colors = lines(4);  % Use different colors for the plots

for i = 1:4
    % Plot numerical solution
    plot(x, C_at_times(:, i), '-o', 'LineWidth', 2, 'Color', colors(i, :), ...
         'DisplayName', ['Numerical, t = ', num2str(time_values(i))]);

    % Plot analytic solution
    plot(x, C_analytic(:, i), '--', 'LineWidth', 2, 'Color', colors(i, :), ...
         'DisplayName', ['Analytic, t = ', num2str(time_values(i))]);
end

hold off;
xlabel('x');
ylabel('C(x,t)');
title('Comparison of Numerical and Analytic Solutions at Selected Time Points');
legend show;
grid on;







%%



% Parameters
alpha = 1.0;     % Diffusivity
tf = 0.25;       % Final time
Nt = 1000;       % Number of time steps
N = 25;          % Number of Legendre polynomials (degree)
dt = tf / Nt;    % Time step size
a = 0;
b = 1;
% Create the spatial grid using LGL nodes
[x,w] = lglnodes(N+1);
```

```
x = (flipud(x)+1)/2*(b-a)+a;
w = w/2*(b-a);

% Initial condition
C = initial_condition(x);
C_at_times = zeros(N+1, 4);  % 4 columns for 4 time points
C_at_times(:, 1) = C;       % Store the initial condition

[D,~,~] = derv(N, x);

% Construct the coefficient matrix for the implicit method
G = diag(w)^-1*D'*diag(w)*D;

A = dt*G*alpha+eye(N+1);

% Time-stepping loop (fully implicit scheme)
for n = 1:Nt
    % Apply boundary conditions
    C(1) = 1;
    C(end) = 0;

    % Solve linear system directly: A * C_new = C_old
    C = A \ C;

    % Store the solution at the selected time steps
    if ismember(n+1, time_indices)
        index = find(time_indices == n+1);
        C_at_times(:, index) = C;
    end
end

% Analytic solution function
analytic_solution = @(x, t) 1 - x - (1 / pi) * exp(-4 * pi^2 * t) * sin(2 * pi * x);

% Compute the analytic solution at the selected time points
C_analytic = zeros(N+1, 4);
for k = 1:4
    C_analytic(:, k) = analytic_solution(x, time_values(k));
end

% Plot the numerical and analytic solutions at the selected time points
figure;
hold on;
colors = lines(4);  % Use different colors for the plots

for i = 1:4
    % Plot numerical solution
    plot(x, C_at_times(:, i), '-o', 'LineWidth', 2, 'Color', colors(i, :), ...
         'DisplayName', ['Numerical, t = ', num2str(time_values(i))]);

    % Plot analytic solution
```

```
    plot(x, C_analytic(:, i), '--', 'LineWidth', 2, 'Color', colors(i, :), ...
         'DisplayName', ['Analytic, t = ', num2str(time_values(i))]);
end

hold off;
xlabel('x');
ylabel('C(x,t)');
title('Comparison of Numerical and Analytic Solutions at Selected Time Points');
legend show;
grid on;










%%
% Parameters
alpha = 1.0;    % Diffusivity
tf = 0.25;      % Final time
Nt = 1000;      % Number of time steps
N =8;        % Number of spatial nodes (LGL nodes)
dt = tf / Nt;   % Time step size
a = -1;
b = 1;
m= 2;
npt = N+1+(N)*(m-1);
% Initial and Boundary Conditions
initial_condition = @(x) 1 - x - (1 / pi) * sin(2 * pi * x);
time_indices = round([1, Nt/3, 2*Nt/3, Nt+1]);
time_values = time_indices * dt;

% % Create the spatial grid using LGL nodes
[xlocal,w] = lglnodes(N+1);
xlocal = (flipud(xlocal)+1)/2*(b-a)+a;
w = w/2*(b-a);
x = linspace(0,1,npt);
x = x';
% Initial condition
C = initial_condition(x);
C_at_times = zeros(npt, 4);  % 4 columns for 4 time points
C_at_times(:, 1) = C;       % Store the initial condition

[D,~,~] = derv(N, xlocal);

% Construct the coefficient matrix for the implicit method
G = (2/((1)/m))*diag(w)^-1*D'*diag(w)*D;
BG = assemble_global_matrix(G,m);
% BG(1,:) = 0;
```

```
% BG(end,:) = 0;
% BG(1,1) = 1;
% BG(9,:) = BG(9,:)./2;
% BG(end,end) = 1;
A = dt*D+eye(npt);


% Time-stepping loop (fully implicit scheme)
for n = 1:Nt
    % Apply boundary conditions
    C(1) = 1;
    C(end) = 0;

    % Solve linear system directly: A * C_new = C_old
    C = A \ C;

    % Store the solution at the selected time steps
    if ismember(n+1, time_indices)
        index = find(time_indices == n+1);
        C_at_times(:, index) = C;
    end
end

% Analytic solution function
analytic_solution = @(x, t) 1 - x - (1 / pi) * exp(-4 * pi^2 * t) * sin(2 * pi * x);

% Compute the analytic solution at the selected time points
C_analytic = zeros(npt, 4);
for k = 1:4
    C_analytic(:, k) = analytic_solution(x, time_values(k));
end

% Plot the numerical and analytic solutions at the selected time points
figure;
hold on;
colors = lines(4);  % Use different colors for the plots

for i = 1:4
    % Plot numerical solution
    plot(x, C_at_times(:, i), '-o', 'LineWidth', 2, 'Color', colors(i, :), ...
         'DisplayName', ['Numerical, t = ', num2str(time_values(i))]);

    % Plot analytic solution
    plot(x, C_analytic(:, i), '--', 'LineWidth', 2, 'Color', colors(i, :), ...
         'DisplayName', ['Analytic, t = ', num2str(time_values(i))]);
end

hold off;
xlabel('x');
ylabel('C(x,t)');
title('Comparison of Numerical and Analytic Solutions at Selected Time Points');
```

```
legend show;
grid on;






%%
```