Ex 1

function x = gauss\_pivot(A, b)

n = length(b);

Ab = [A, b]; % Construim matricea extinsă

% Eliminare Gauss cu pivotare parțială

for k = 1:n-1

% Pivotare: găsim rândul cu valoarea maximă în coloana k

[~, max\_row] = max(abs(Ab(k:n, k)));

max\_row = max\_row + k - 1;

% Schimbăm rândurile dacă este necesar

if max\_row ~= k

Ab([k, max\_row], :) = Ab([max\_row, k], :);

end

% Eliminare Gauss

for i = k+1:n

factor = Ab(i, k) / Ab(k, k);

Ab(i, k:end) = Ab(i, k:end) - factor \* Ab(k, k:end);

end

end

% Substituție înapoi

x = zeros(n, 1);

for i = n:-1:1

x(i) = (Ab(i, end) - Ab(i, i+1:n) \* x(i+1:n)) / Ab(i, i);

end

end

% Definim matricea A și vectorul b conform sistemului de ecuații

A = [3 2 -1; 2 -2 4; -1 0.5 -1];

b = [1; -2; 0];

% Apelăm funcția pentru rezolvarea sistemului

sol = gauss\_pivot(A, b);

% Afișăm soluția

disp('Soluția sistemului este:');

disp(sol);

Ex 2.

function [X, num\_iter] = jacobi\_method(A, B, X0, eps\_sis, Nmax)

n = length(B);

X = X0;

X\_prev = X;

num\_iter = 0;

while num\_iter < Nmax

for i = 1:n

sum\_val = B(i);

for j = 1:n

if j ~= i

sum\_val = sum\_val - A(i, j) \* X\_prev(j);

end

end

X(i) = sum\_val / A(i, i);

end

% Calculul normei erorii relative

err = norm(X - X\_prev, inf) / norm(X, inf);

if err < eps\_sis

break;

end

X\_prev = X;

num\_iter = num\_iter + 1;

end

end

% Test pentru sistemul dat

A = [10 1 1; 1 10 -1; 1 -1 10];

B = [6; 3; 6];

X0 = [0; 0; 0]; % Aproximarea inițială

eps\_sis = 1e-6; % Precizie dorită

Nmax = 5000; % Număr maxim de iterații

[X\_sol, iter] = jacobi\_method(A, B, X0, eps\_sis, Nmax);

disp('Soluția sistemului:');

disp(X\_sol);

disp(['Număr de iterații: ', num2str(iter)]);

Ex 3

a)

function [X, num\_iter] = jacobi\_method(A, B, X0, eps\_sis, Nmax)

n = length(B);

X = X0;

X\_prev = X;

num\_iter = 0;

while num\_iter < Nmax

for i = 1:n

sum\_val = B(i);

for j = 1:n

if j ~= i

sum\_val = sum\_val - A(i, j) \* X\_prev(j);

end

end

X(i) = sum\_val / A(i, i);

end

% Calculul normei erorii relative

err = norm(X - X\_prev, inf) / norm(X, inf);

if err < eps\_sis

break;

end

X\_prev = X;

num\_iter = num\_iter + 1;

end

end

% Test pentru sistemul dat

A = [10 1 1; 1 10 -1; 1 -1 10];

B = [6; 3; 6];

X0 = [0; 0; 0]; % Aproximarea inițială

eps\_sis = 1e-6; % Precizie dorită

Nmax = 5000; % Număr maxim de iterații

[X\_sol, iter] = jacobi\_method(A, B, X0, eps\_sis, Nmax);

disp('Soluția sistemului:');

disp(X\_sol);

disp(['Număr de iterații: ', num2str(iter)]);

b)

function [X, num\_iter] = jacobi\_method(A, B, X0, eps\_sis, Nmax)

n = length(B);

X = X0;

X\_prev = X;

num\_iter = 0;

while num\_iter < Nmax

for i = 1:n

sum\_val = B(i);

for j = 1:n

if j ~= i

sum\_val = sum\_val - A(i, j) \* X\_prev(j);

end

end

X(i) = sum\_val / A(i, i);

end

% Calculul normei erorii relative

err = norm(X - X\_prev, inf) / norm(X, inf);

if err < eps\_sis

break;

end

X\_prev = X;

num\_iter = num\_iter + 1;

end

end

% Test pentru sistemul dat

A = [10 1 1; 1 10 -1; 1 -1 10];

B = [6; 3; 6];

X0 = [0; 0; 0]; % Aproximarea inițială

eps\_sis = 1e-6; % Precizie dorită

Nmax = 5000; % Număr maxim de iterații

[X\_sol, iter] = jacobi\_method(A, B, X0, eps\_sis, Nmax);

disp('Soluția sistemului:');

disp(X\_sol);

disp(['Număr de iterații: ', num2str(iter)]);

c)

function [A, B, X\_exact] = generate\_test\_data(n)

% Generare matrice A cu diagonală dominantă

A = rand(n) \* 10; % Matrice aleatoare

for i = 1:n

A(i, i) = sum(abs(A(i, :))) + rand() \* 10; % Asigurăm diagonală dominantă

end

% Generare vector X\_exact aleator

X\_exact = rand(n, 1) \* 10;

% Calculăm vectorul B = A \* X\_exact

B = A \* X\_exact;

end

% Testare generare date

[A\_test, B\_test, X\_test] = generate\_test\_data(3);

disp('Matricea A generată:');

disp(A\_test);

disp('Vectorul B generat:');

disp(B\_test);

disp('Soluția exactă X:');

disp(X\_test);

d)

eps\_sis = 1e-6; % Precizia dorită

Nmax = 5000; % Număr maxim de iterații

num\_tests = 3; % Număr de teste

for i = 1:num\_tests

[A, B, X\_exact] = generate\_test\_data(3); % Generare matrice

X0 = zeros(3,1); % Aproximare inițială

[X\_sol, iter] = jacobi\_method(A, B, X0, eps\_sis, Nmax);

% Calculul erorii relative

err\_rel = norm(X\_sol - X\_exact, inf) / norm(X\_exact, inf);

% Afișare rezultate

fprintf('Test %d:\n', i);

disp('Soluția exactă:');

disp(X\_exact);

disp('Soluția aproximată:');

disp(X\_sol);

disp(['Eroarea relativă: ', num2str(err\_rel)]);

% Comparăm eroarea cu eps\_sis

if err\_rel < eps\_sis

disp('Eroarea este mai mică decât precizia impusă.');

else

disp('Eroarea este mai mare decât precizia impusă.');

end

disp('------------------------------------');

end

e)

precizii = logspace(-1, -16, 10); % Precizii de la 10^-1 la 10^-16

num\_iteratii = zeros(size(precizii));

[A, B, ~] = generate\_test\_data(3); % Folosim o matrice fixă

X0 = zeros(3,1);

Nmax = 5000;

for i = 1:length(precizii)

[~, num\_iter] = jacobi\_method(A, B, X0, precizii(i), Nmax);

num\_iteratii(i) = num\_iter;

end

% Reprezentare grafică

figure;

semilogx(precizii, num\_iteratii, '-o', 'LineWidth', 2);

xlabel('Precizia impusă (\epsilon\_{sis})');

ylabel('Numărul de iterații');

title('Variatia numărului de iterații în funcție de precizie');

grid on;

Ex 4

a)

function [X, num\_iter] = gauss\_seidel(A, B, X0, eps\_sis, Nmax)

n = length(B);

X = X0;

num\_iter = 0;

while num\_iter < Nmax

X\_prev = X;

for i = 1:n

sum\_val = B(i);

for j = 1:n

if j ~= i

sum\_val = sum\_val - A(i, j) \* X(j);

end

end

X(i) = sum\_val / A(i, i);

end

% Calculul normei erorii relative

err = norm(X - X\_prev, inf) / norm(X, inf);

if err < eps\_sis

break;

end

num\_iter = num\_iter + 1;

end

end

% Test pentru sistemul dat

A = [4 1 1; 2 5 2; 1 2 3];

B = [7; 3; 5];

X0 = [0; 0; 0]; % Aproximarea inițială

eps\_sis = 1e-6; % Precizie dorită

Nmax = 5000; % Număr maxim de iterații

[X\_sol, iter] = gauss\_seidel(A, B, X0, eps\_sis, Nmax);

disp('Soluția sistemului:');

disp(X\_sol);

disp(['Număr de iterații: ', num2str(iter)]);

b)

function [A, B, X\_exact] = generate\_test\_data(n)

% Generare matrice A cu diagonală dominantă

A = rand(n) \* 10; % Matrice aleatoare

for i = 1:n

A(i, i) = sum(abs(A(i, :))) + rand() \* 10; % Asigurăm diagonală dominantă

end

% Generare vector X\_exact aleator

X\_exact = rand(n, 1) \* 10;

% Calculăm vectorul B = A \* X\_exact

B = A \* X\_exact;

end

% Testare generare date

[A\_test, B\_test, X\_test] = generate\_test\_data(3);

disp('Matricea A generată:');

disp(A\_test);

disp('Vectorul B generat:');

disp(B\_test);

disp('Soluția exactă X:');

disp(X\_test);

c)

eps\_sis = 1e-6; % Precizia dorită

Nmax = 5000; % Număr maxim de iterații

num\_tests = 3; % Număr de teste

for i = 1:num\_tests

[A, B, X\_exact] = generate\_test\_data(3); % Generare matrice

X0 = zeros(3,1); % Aproximare inițială

[X\_sol, iter] = gauss\_seidel(A, B, X0, eps\_sis, Nmax);

% Calculul erorii relative

err\_rel = norm(X\_sol - X\_exact, inf) / norm(X\_exact, inf);

% Afișare rezultate

fprintf('Test %d:\n', i);

disp('Soluția exactă:');

disp(X\_exact);

disp('Soluția aproximată:');

disp(X\_sol);

disp(['Eroarea relativă: ', num2str(err\_rel)]);

% Comparăm eroarea cu eps\_sis

if err\_rel < eps\_sis

disp('Eroarea este mai mică decât precizia impusă.');

else

disp('Eroarea este mai mare decât precizia impusă.');

end

disp('------------------------------------');

end

d)

precizii = logspace(-1, -16, 10); % Precizii de la 10^-1 la 10^-16

num\_iteratii = zeros(size(precizii));

[A, B, ~] = generate\_test\_data(3); % Folosim o matrice fixă

X0 = zeros(3,1);

Nmax = 5000;

for i = 1:length(precizii)

[~, num\_iter] = gauss\_seidel(A, B, X0, precizii(i), Nmax);

num\_iteratii(i) = num\_iter;

end

% Reprezentare grafică

figure;

semilogx(precizii, num\_iteratii, '-o', 'LineWidth', 2);

xlabel('Precizia impusă (\epsilon\_{sis})');

ylabel('Numărul de iterații');

title('Variatia numărului de iterații în funcție de precizie');

grid on;