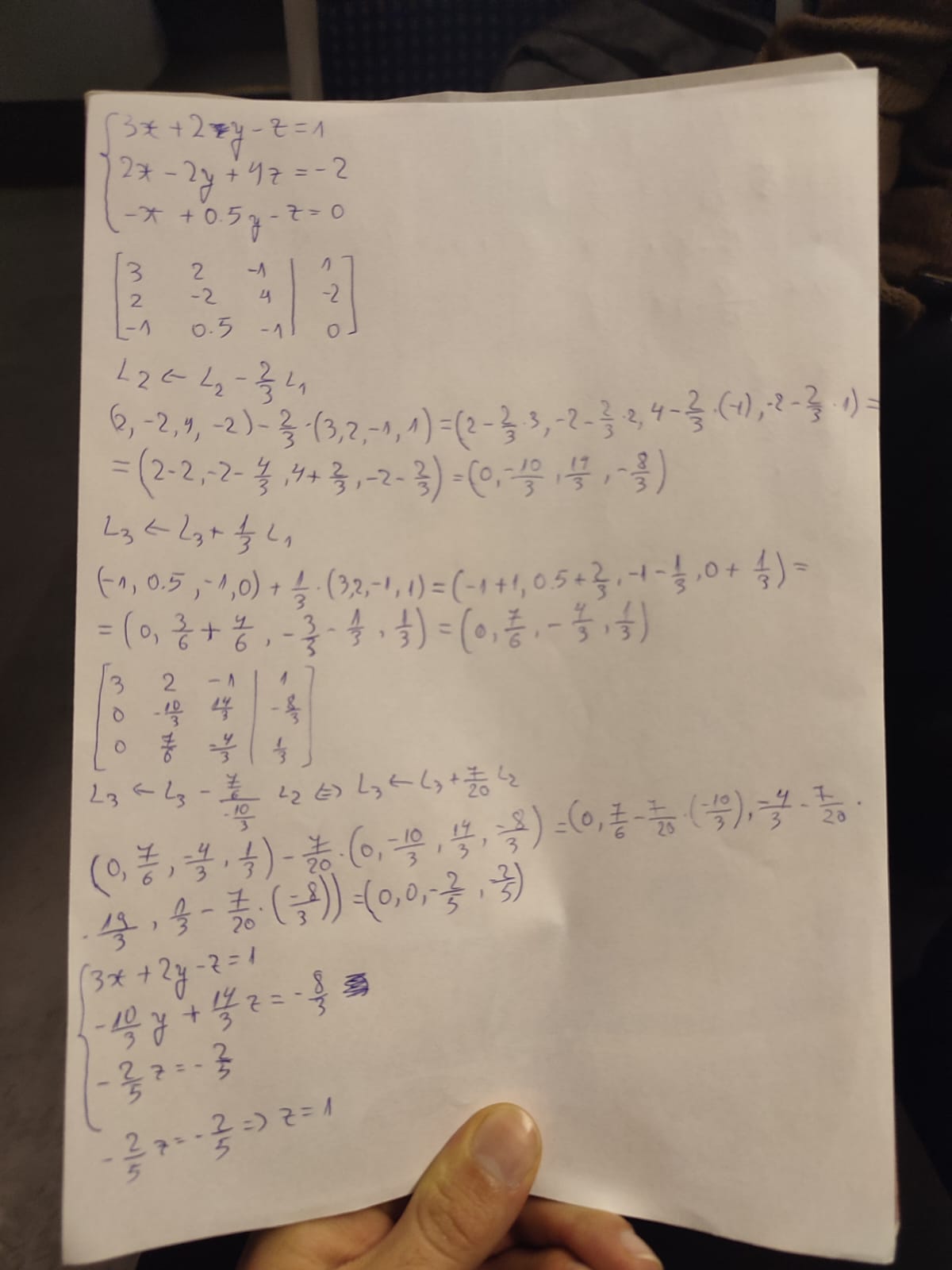
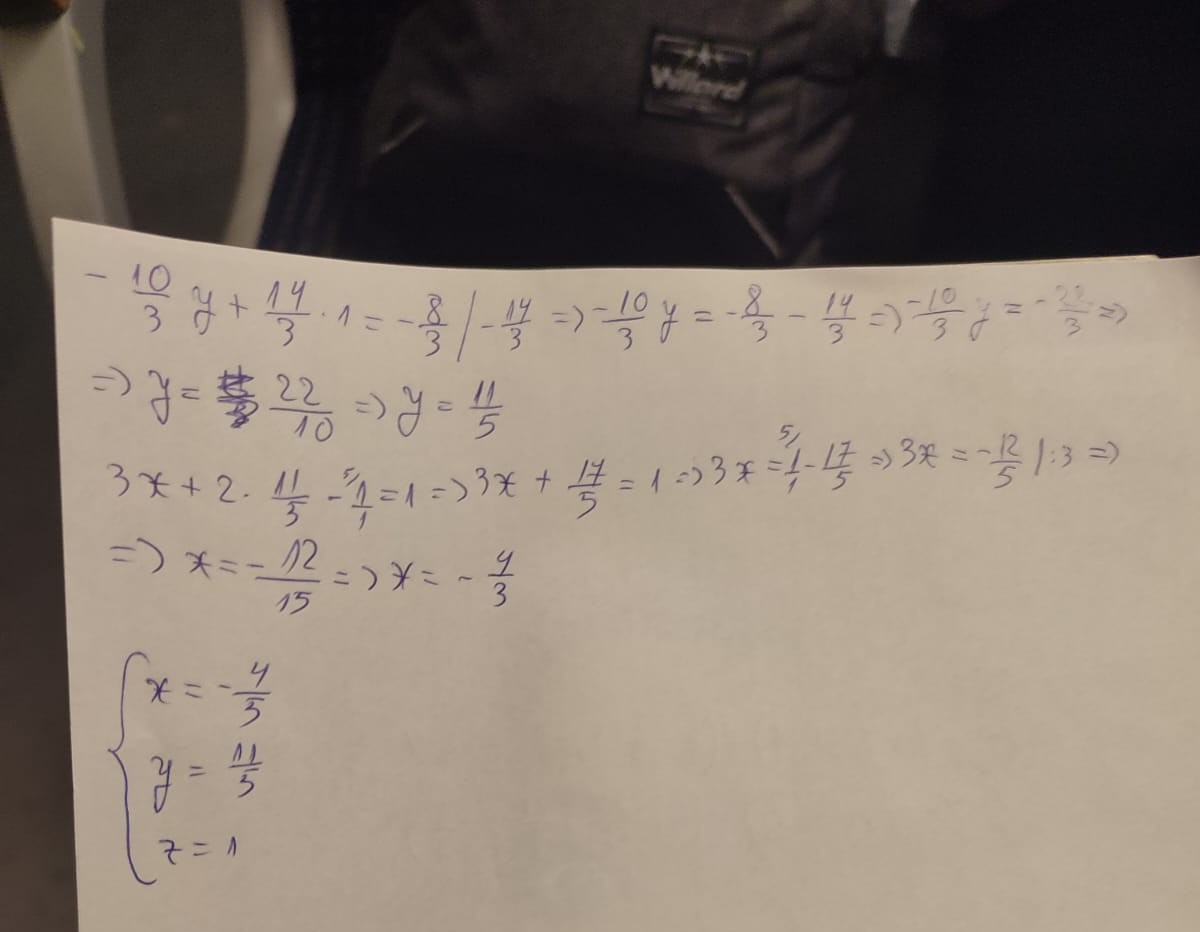
**Mocanu Marian Valentin** Tema Laborator 5









function x = rezolvare\_sistem()

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

b = [1; -2; 0];

% aplicam eliminarea Gauss

Ab = [A b]; % matricea extinsa

n = length(b);

% faza de eliminare

for i = 1:n-1

for j = i+1:n

m = Ab(j,i) / Ab(i,i);

Ab(j,:) = Ab(j,:) - m \* Ab(i,:);

end

end

% substitutia inversa

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);

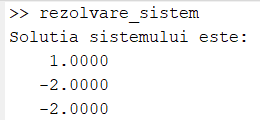
end

% afisam solutia

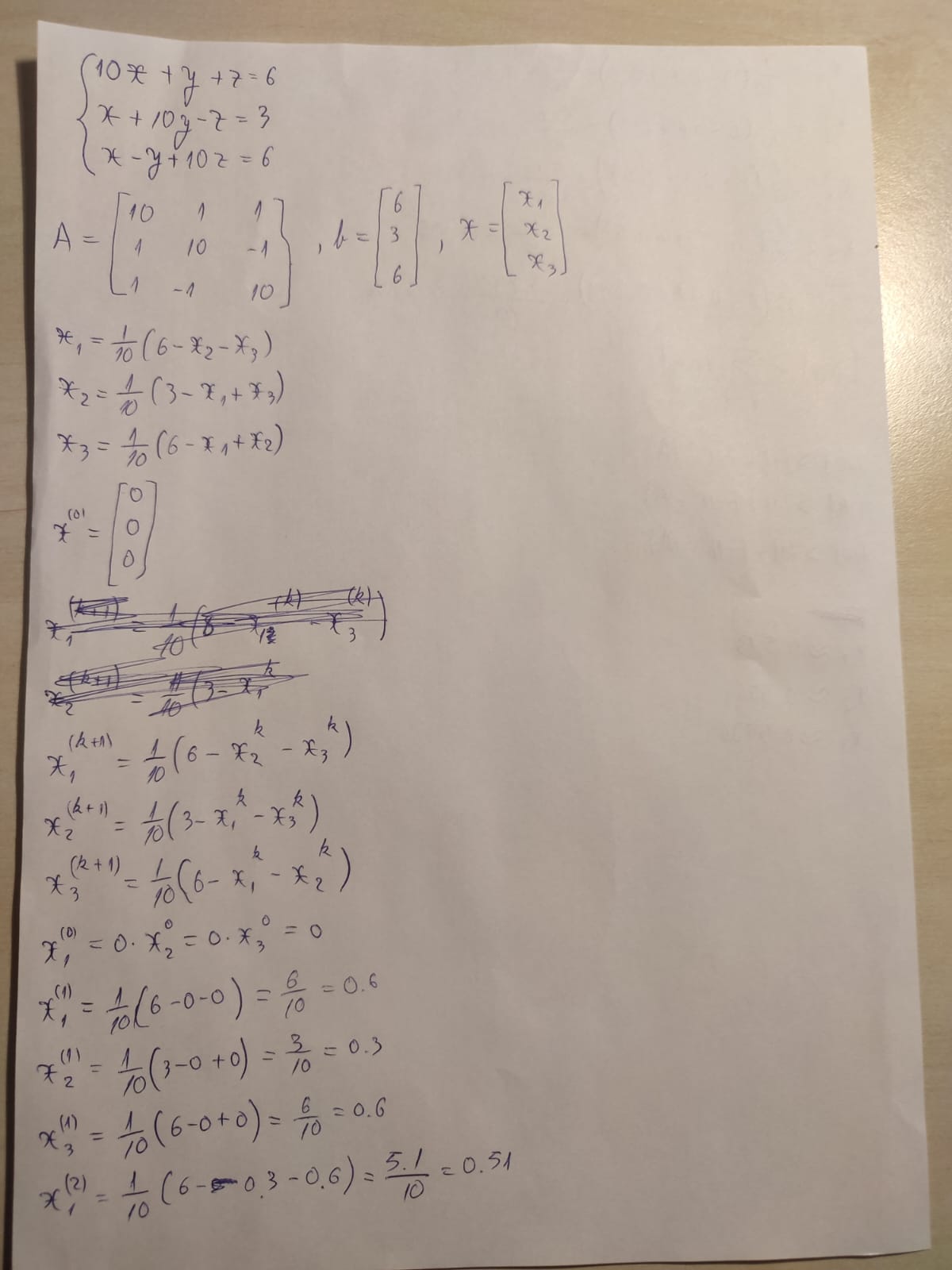
disp('Solutia sistemului este:');

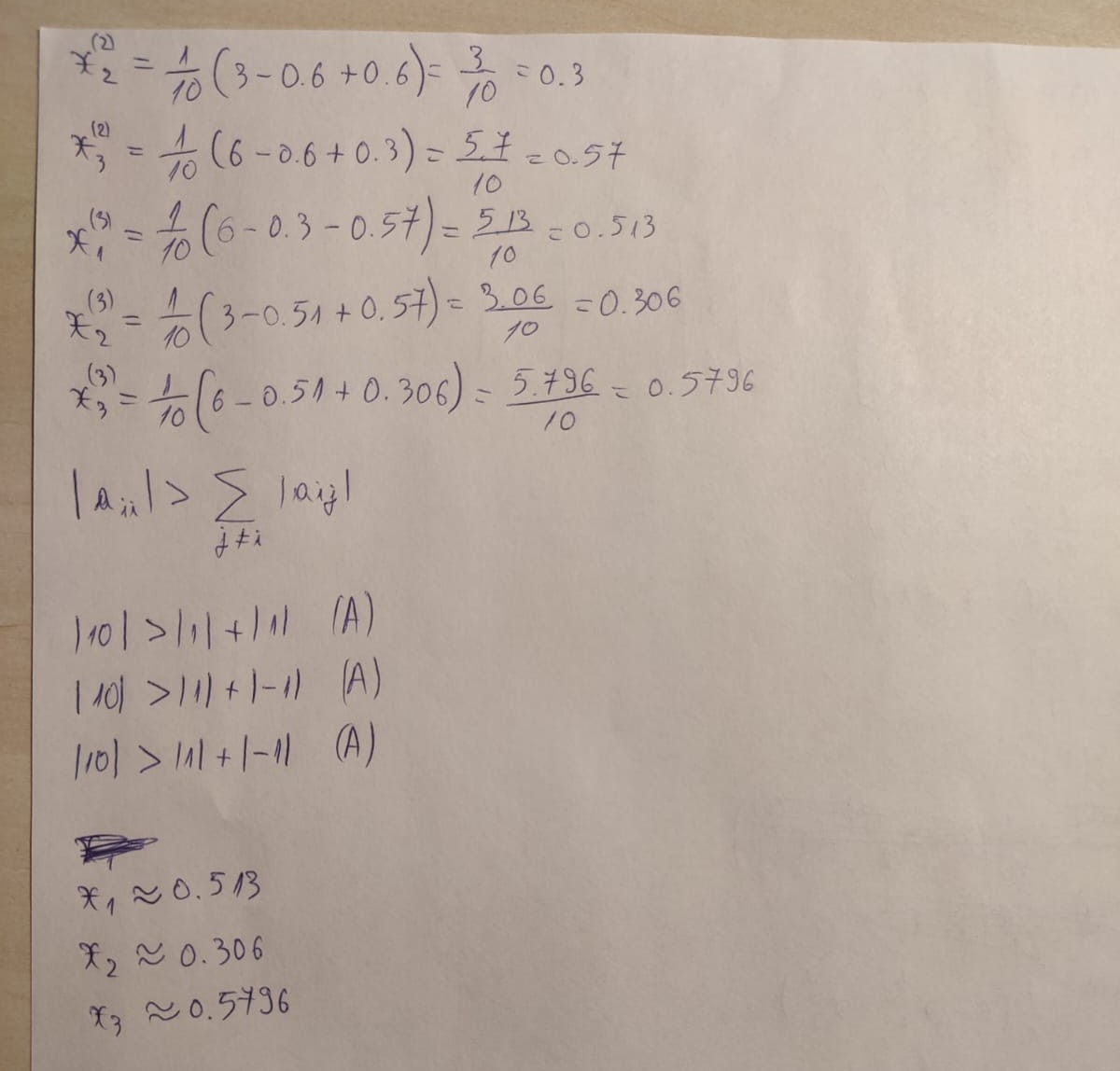
disp(x);

end









* jacobi\_method.m

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

if nargin < 5

error('Nu sunt suficienti parametri de intrare: "jacobi\_method(A, B, X0, eps\_sis, Nmax)"');

end

n = size(A,1);

if size(A,2) ~= n || size(B,1) ~= n || numel(B) ~= n || numel(X0) ~= n

error('Nepotrivire dimensionala: Asigura-te ca A este o matrice patrata, iar B și X0 sunt vectori coloana de dimensiune corespunzatoare.');

end

X = X0(:);

X\_prev = X;

iter = 0;

while iter < Nmax

X\_new = X;

for i = 1:n

sigma = A(i, :) \* X\_prev - A(i, i) \* X\_prev(i);

X\_new(i) = (B(i) - sigma) / A(i, i);

end

if norm(X\_new - X\_prev, inf) < eps\_sis

X = X\_new;

break;

end

X\_prev = X\_new;

iter = iter + 1;

end

X = X\_prev;

end

* ex2\_punctul\_b.m

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

B = [6; 3; 6];

% alegem un vector initial X0

X0 = zeros(size(B));

% definim precizia si numarul maxim de iteratii

eps\_sis = 1e-6;

Nmax = 5000;

% apelam functia

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

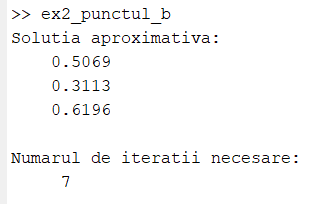
% afisam rezultatele

disp('Solutia aproximativa:')

disp(X)

disp('Numarul de iteratii necesare:')

disp(iter)





* generate\_test\_data.m

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

% generam aleator o matrice patratica cu diagonala dominanta

A = rand(n) \* 10;

for I = 1:n

A(i, i) = sum(abs(A(i, :))) + rand() \* 10;

end

% generam aleator un vector solutie exacta X\_exact

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

% calculam vectorul B ca produs A \* X\_exact

B = A \* X\_exact;

end

* ex2\_punctul\_c.m

% setam dimensiunea sistemului

n = 3;

% generam datele de test

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

% alegem o solutie initială X0

X0 = zeros(n, 1);

% definim precizia si numarul maxim de iteratii

eps\_sis = 1e-6;

Nmax = 5000;

% apelam metoda Jacobi

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

% afisam rezultatele

disp('Matricea A:')

disp(A)

disp('Vectorul B:')

disp(B)

disp('Solutia exacta X\_exact:')

disp(X\_exact)

disp('Solutia aproximativa obtinuta X\_approx:')

disp(X\_approx)

disp('Numarul de iteratii necesare:')

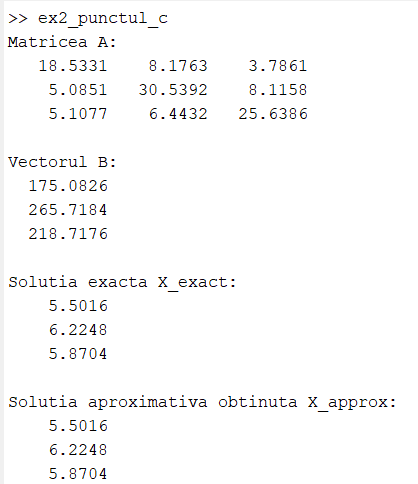
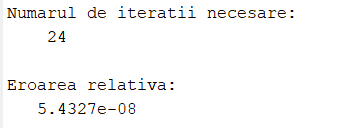
disp(iter)

% calculam eroarea relativa

error\_relative = norm(X\_approx - X\_exact, inf) / norm(X\_exact, inf);

disp('Eroarea relativa:')

disp(error\_relative)



* evaluate\_jacobi.m

function evaluate\_jacobi(n, eps\_sis, Nmax)

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

X0 = zeros(n, 1);

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

error\_relative = norm(X\_approx - X\_exact, inf) / norm(X\_exact, inf);

disp('Matricea A:');

disp(A);

disp('Vectorul B:');

disp(B);

disp('Soluția exactă X\_exact:');

disp(X\_exact);

disp('Soluția aproximativă X\_approx:');

disp(X\_approx);

disp('Numărul de iterații necesare:');

disp(iter);

disp('Eroarea relativă:');

disp(error\_relative);

end

* ex2\_punctul\_d.m

% setam dimensiunea sistemului

n = 3;

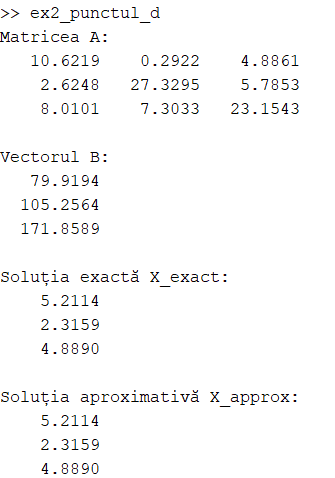
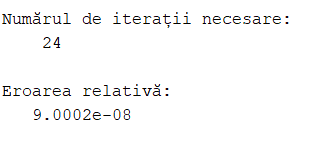
% definim precizia si numarul maxim de iteratii

eps\_sis = 1e-6;

Nmax = 5000;

% apelam functia de evaluare

evaluate\_jacobi(n, eps\_sis, Nmax);



* plot\_iterations\_vs\_precision.m

function plot\_iterations\_vs\_precision(n, Nmax)

eps\_values = logspace(-1, -16, 10);

iterations = zeros(size(eps\_values));

for I = 1:length(eps\_values)

eps\_sis = eps\_values(i);

[A, B, ~] = generate\_test\_data(n);

X0 = zeros(n, 1);

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

iterations(i) = iter;

end

figure;

semilogx(eps\_values, iterations,'-o');

xlabel('Precizia');

ylabel('Numarul de iteratii');

title('Variatia numarului de iteratii in functie de precizie');

grid on;

end

* ex2\_punctul\_e.m

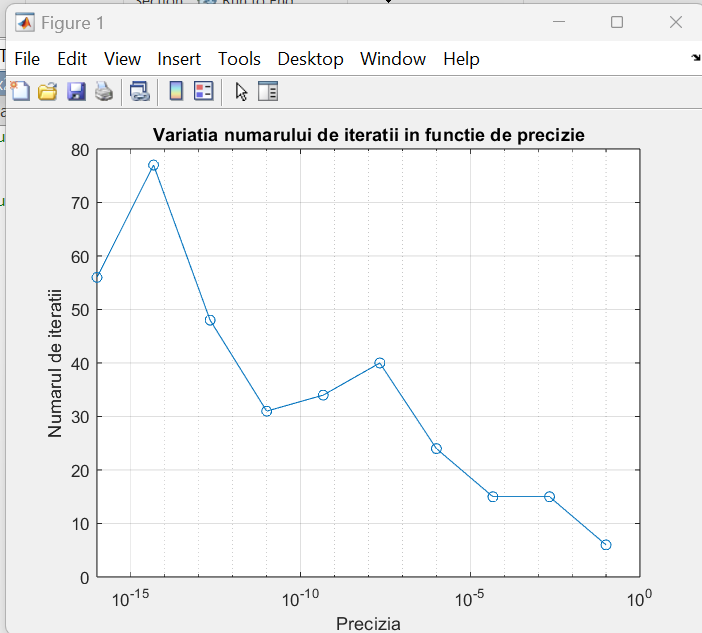
% setam dimensiunea sistemului si numarul maxim de iteratii

n = 3;

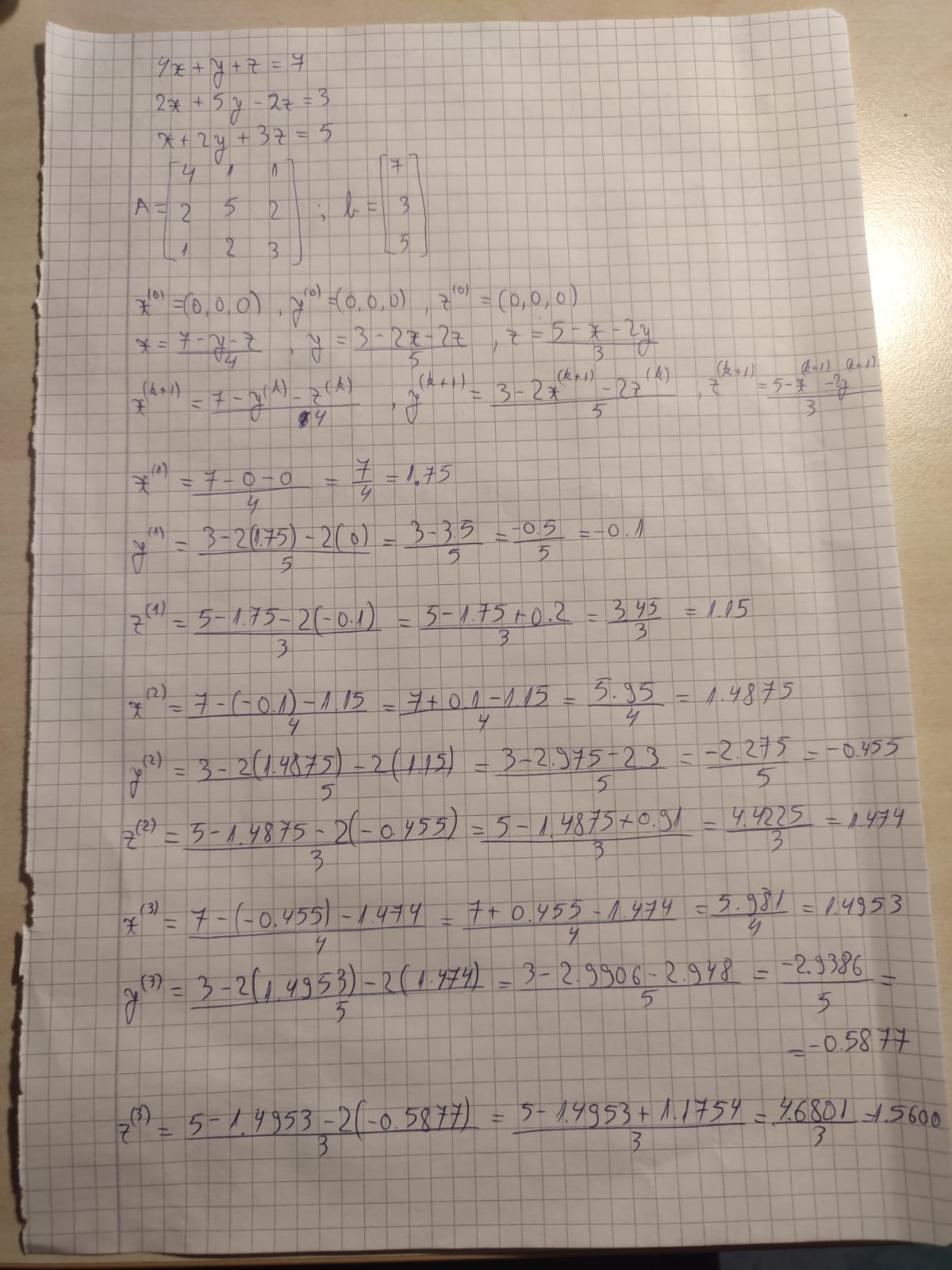
Nmax = 5000;

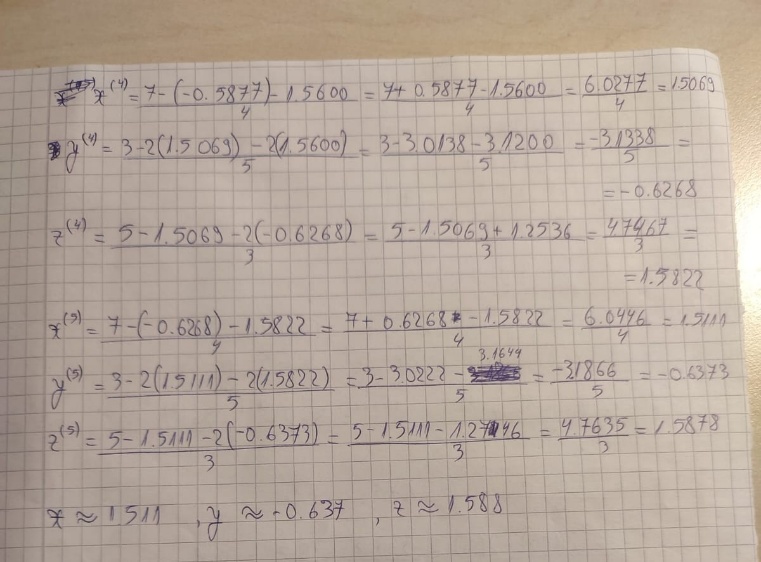
% apelam functia pentru a genera graficul

plot\_iterations\_vs\_precision(n, Nmax);











* gauss\_seidel.m

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

% verificam dimensiunile matricii A și vectorului B

[n, m] = size(A);

if n ~= m || length(B) ~= n

error('Dimensiunile matricei A si vectorului B nu sunt compatibile!');

end

% initializare

X = zeros(n, 1); % vector initial de solutii

X\_prev = X;

num\_iter = 0;

% iteratia Gauss-Seidel

while num\_iter < Nmax

num\_iter = num\_iter + 1;

for I = 1:n

sum1 = sum(A(1i, 1:1i - 1) .\* X(1:1i - 1)'); % elemente anterioare

sum2 = sum(A(1i, 1i + 1:n) .\* X\_prev(1i + 1:n)'); % elemente viitoare

X(1i) = (B(1i) - sum1 - sum2) / A(1i, 1i);

end

% verificam criteriul de oprire

if norm(X - X\_prev, inf) < eps\_sis

break;

end

X\_prev = X;

end

end

* ex3\_punctul\_b.m

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

B = [7; 3; 5];

% setam precizia si numarul maxim de iteratii

eps\_sis = 1e-6;

Nmax = 5000;

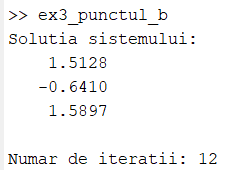
% apelam functia

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

disp('Solutia sistemului:');

disp(X);

disp(['Numar de iteratii: ',num2str(num\_iter)]);





* generate\_test\_data.m

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

% generam aleator o matrice patratica A cu diagonala dominanta

A = rand(n, n)\*10;

for I = 1:n

A(i, i) = sum(abs(A(i, :))) + rand() \* 10; % asiguram dominanta diagonala

end

% generam un vector X aleator

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

% calculam vectorul B

B = A \* X;

end

* ex3\_punctul\_c.m

n = 3;

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

disp('Matricea A:');

disp(A);

disp('Vectorul X generat (solutia exacta):');

disp(X\_exact);

disp('Vectorul B calculat:');

disp(B);



* test\_gauss\_seidel.m

function [error\_norms, num\_iter] = test\_gauss\_seidel(n, eps\_sis, Nmax)

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

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

% calculam norma relativa a erorii

error\_norm = norm(X\_approx - X\_exact) / norm(X\_exact);

disp(['Numar de iteratii: ',num2str(num\_iter)]);

disp(['Norma relativă a erorii: ',num2str(error\_norm)]);

% returnam norma erorii

error\_norms = error\_norm;

end

* ex3\_punctul\_d.m

n = 3;

eps\_sis = 1e-6; % precizia dorita

Nmax = 5000; % numar maxim de iteratii

error\_norm = test\_gauss\_seidel(n, eps\_sis, Nmax);

disp(['Norma relativă a erorii obținute: ',num2str(error\_norm)]);



* plot\_precision\_vs\_iterations.m

function plot\_precision\_vs\_iterations(n, eps\_values, Nmax)

num\_eps = length(eps\_values);

iterations = zeros(1, num\_eps);

for I = 1:num\_eps

eps\_sis = eps\_values(i);

[~, num\_iter] = test\_gauss\_seidel(n, eps\_sis, Nmax);

iterations(i) = num\_iter;

end

% reprezentam grafic variatia numarului de iteratii in functie de precizie

semilogx(eps\_values, iterations, '-gp');

xlabel('Precizia (eps)');

ylabel('Numar de iteratii');

title('Variatia numarului de iteratii in functie de precizie');

grid on;

end

* ex3\_punctul\_e.m

n = 3;

eps\_values = logspace(-1, -16, 10); % precizii între 10^-1 și 10^-16

Nmax = 5000;

plot\_precision\_vs\_iterations(n, eps\_values, Nmax);

