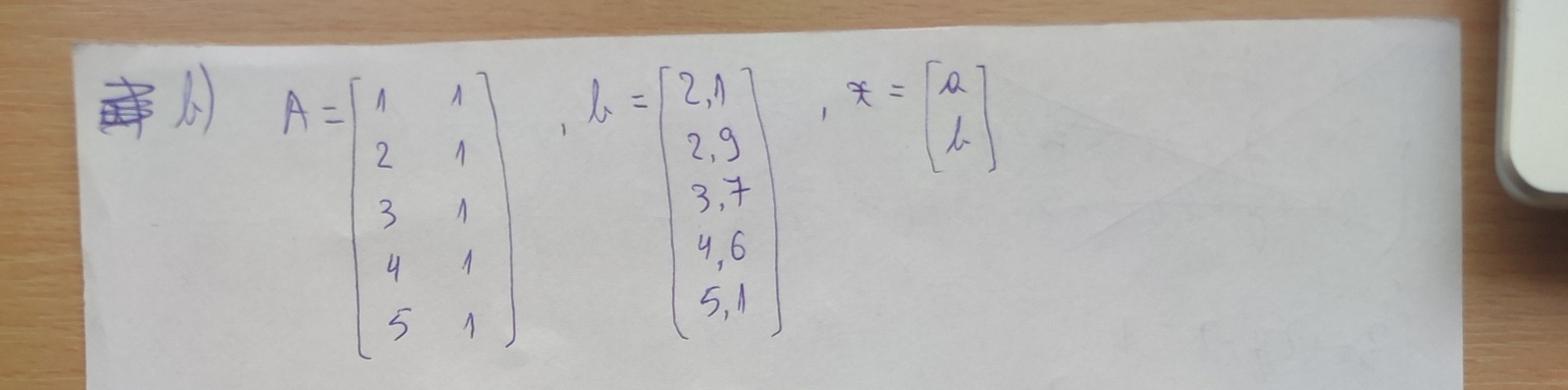
**Mocanu Marian Valentin** Tema Laborator 6+7

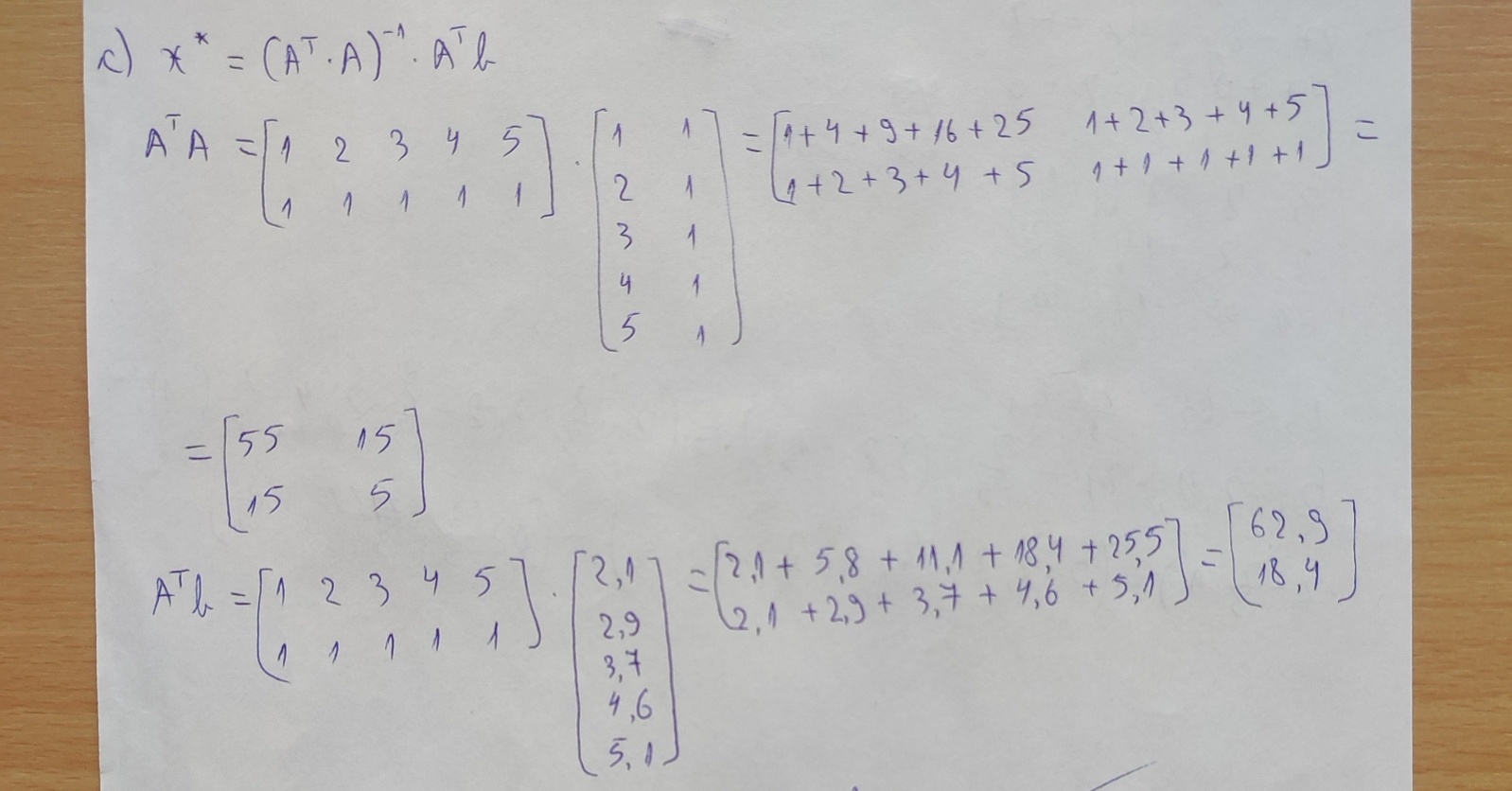


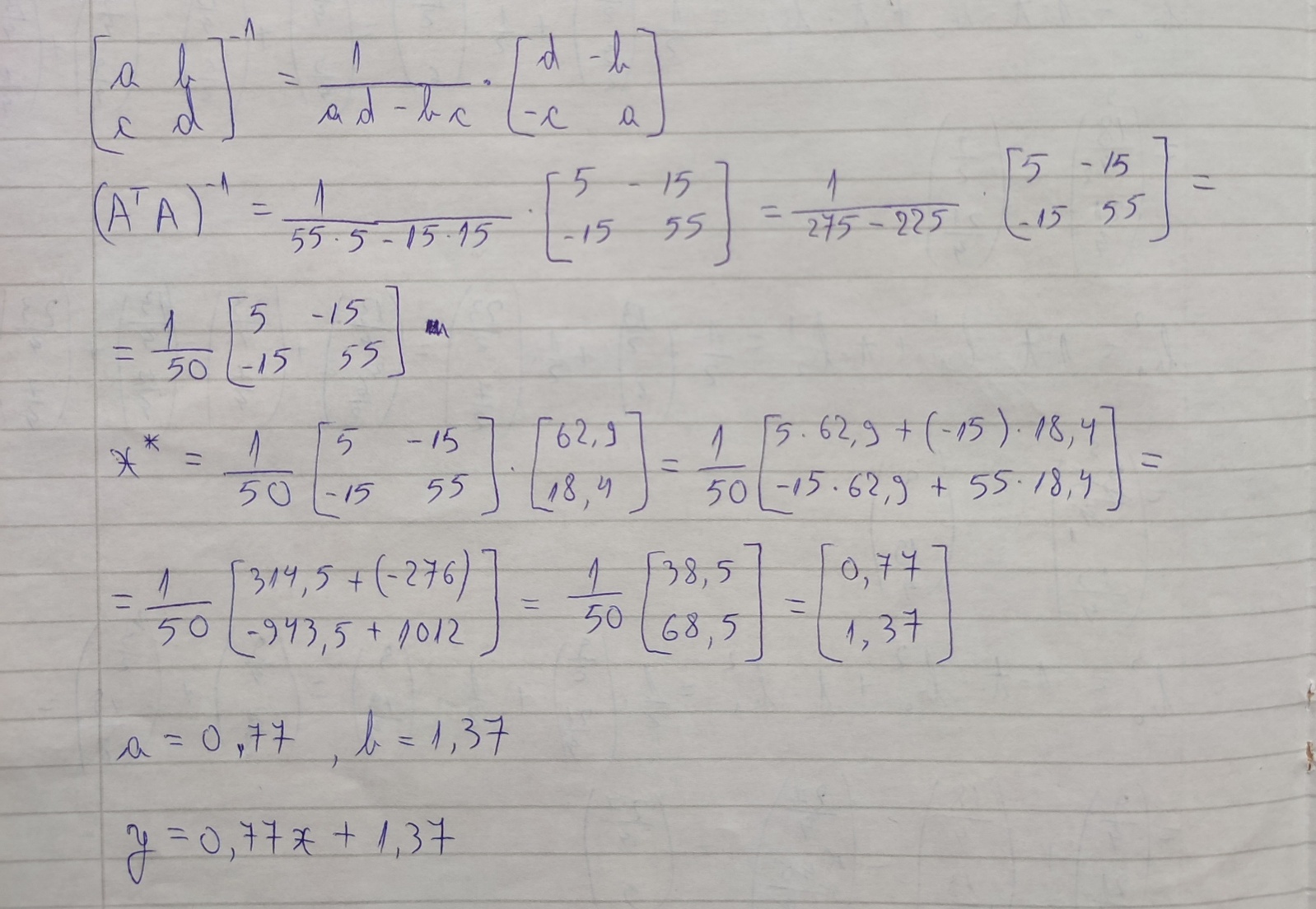
|  |  |
| --- | --- |
| x | y |
| 1 | 2.1 |
| 2 | 2.9 |
| 3 | 3.7 |
| 4 | 4.6 |
| 5 | 5.1 |













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

y = [2.1; 2.9; 3.7; 4.6; 5.1];

% construim matricea A pentru regresia liniara

A = [x, ones(size(x))];

% calculam coeficientii folosind metoda Least-Squares

x\_steluta = (A' \* A) \ (A' \* y);

% extragem coeficientii a si b

a = x\_steluta(1);

b = x\_steluta(2);

fprintf('Coeficientul a: %.2f\n', a);

fprintf('Coeficientul b: %.2f\n', b);

fprintf('Ecuatia dreptei de regresie: y = %.2fx + %.2f\n', a, b);

% generam puncte pentru plotarea dreptei

x\_fit = linspace(min(x)-1, max(x)+1, 100);

y\_fit = a \* x\_fit + b;

hold on;

scatter(x, y, 'ro', 'LineWidth', 1); % puncte experimentale (rosu)

plot(x\_fit, y\_fit, 'b-'); % dreapta de regresie (albastru)

xlabel('x');

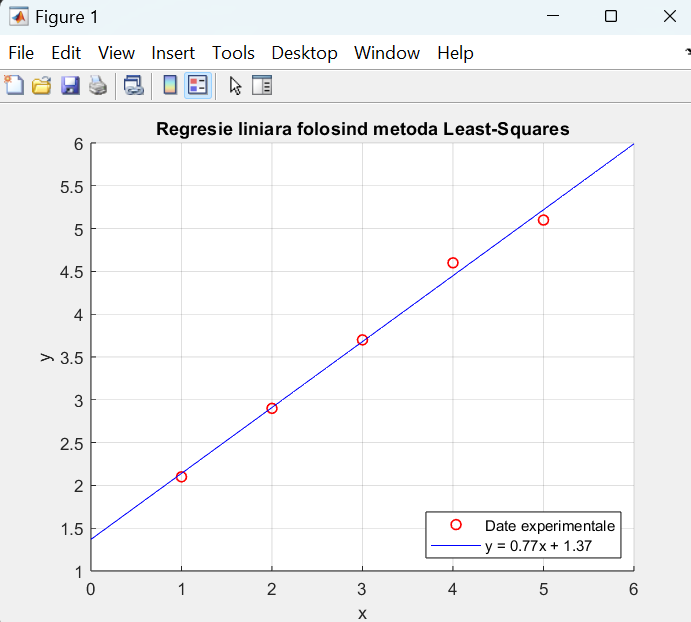
ylabel('y');

title('Regresie liniara folosind metoda Least-Squares');

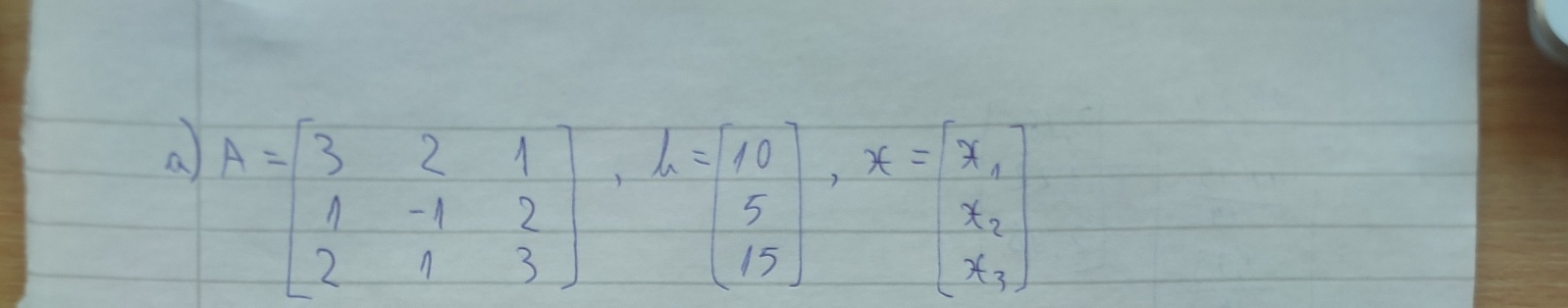
legend('Date experimentale', sprintf('y = %.2fx + %.2f', a, b),'Location','southeast');

grid on;

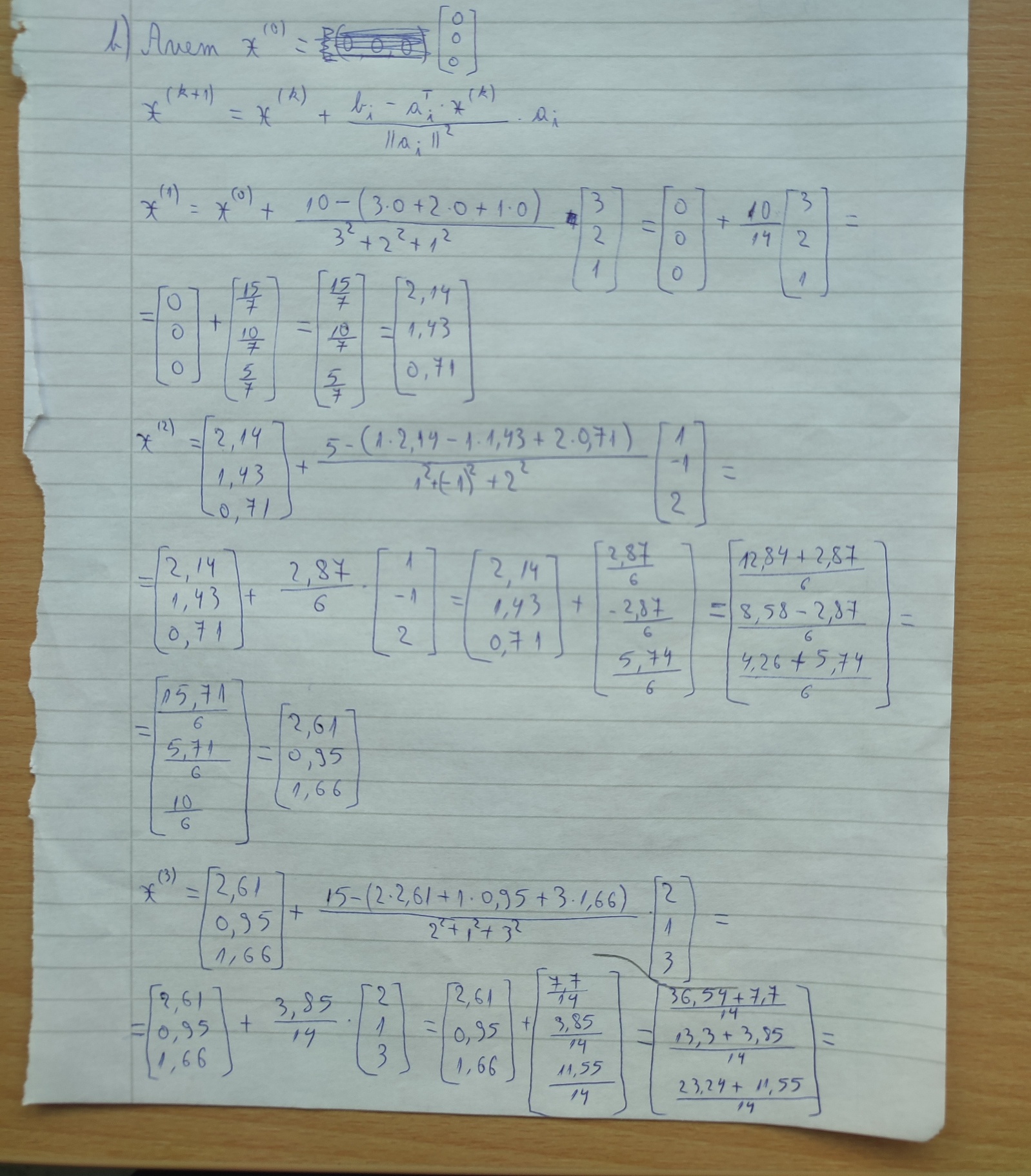
hold off;

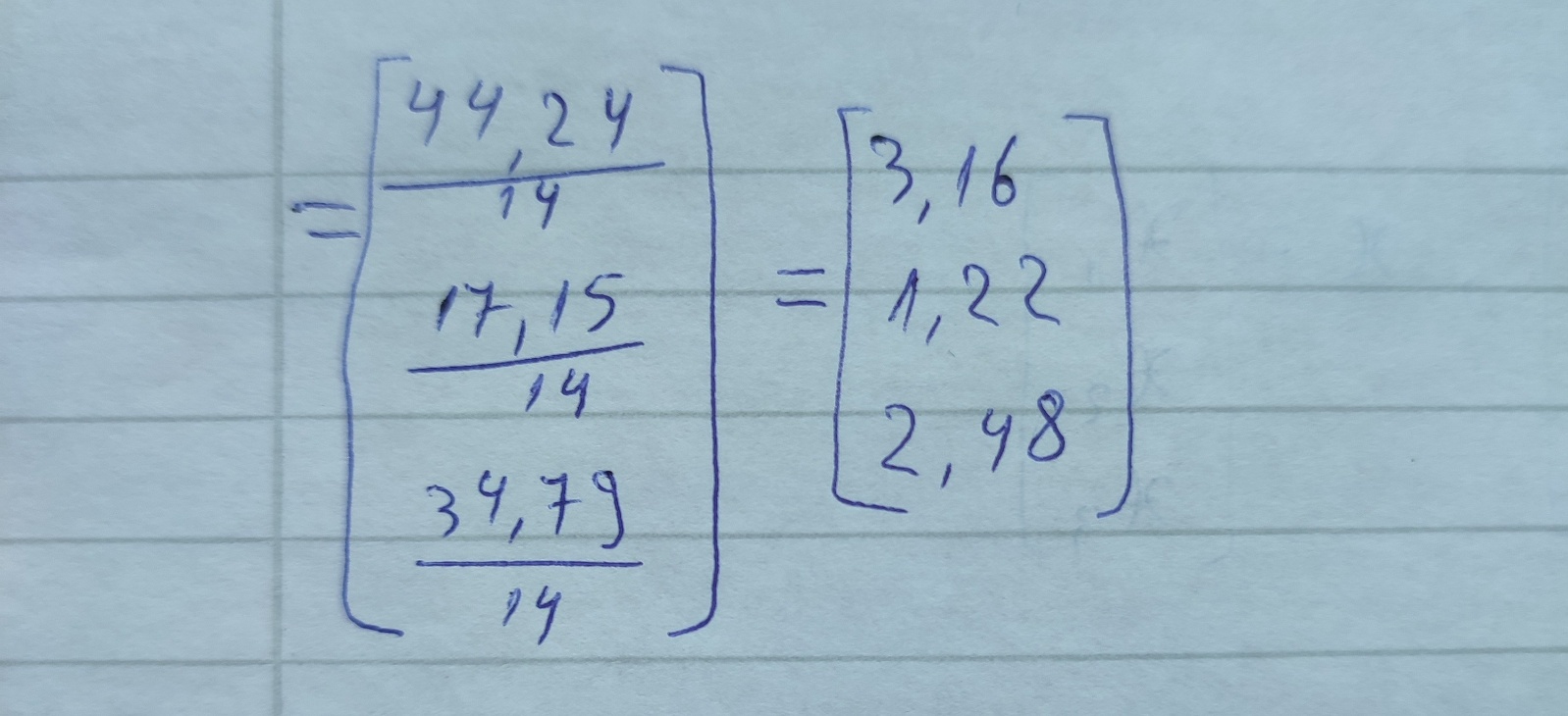














A = [3 2 1; 1 -1 2; 2 1 3];

b = [10; 5; 15];

% initializare solutie (x0 = [0;0;0])

x = zeros(3,1);

% numarul de iteratii

num\_iter = 10;

% algoritmul Kaczmarz

for k = 1:num\_iter

for i = 1:size(A,1) % parcurgem fiecare ecuatie

Ai = A(i, :); % randul i din matricea A

bi = b(i); % valoarea corespunzatoare din vectorul b

ri = bi - Ai \* x; % calculul erorii pentru ecuatia curenta

x = x + (ri / norm(Ai)^2) \* Ai'; % actualizare solutie

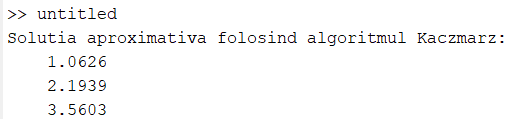
end

end

% afisare solutie aproximativa

disp('Solutia aproximativa folosind algoritmul Kaczmarz:');

disp(x);





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

b = [10; 5; 15];

% solutia exacta folosind metoda directa (Ax = b)

x\_exact = A \ b;

% initializare solutie (x0 = [0;0;0])

x = zeros(3,1);

% numarul de iteratii

num\_iter = 10;

error\_values = zeros(num\_iter, 1); % stocam eroarea la fiecare iteratie

% algoritmul Kaczmarz cu analiza erorii

for k = 1:num\_iter

for i = 1:size(A,1) % parcurgem fiecare ecuatie

Ai = A(i, :); % randul i din matricea A

bi = b(i); % valoarea corespunzatoare din vectorul b

ri = bi - Ai \* x; % calculul erorii pentru ecuatia curenta

x = x + (ri / norm(Ai)^2) \* Ai'; % actualizare solutie

end

% calculam eroarea normei intre solutia exacta si cea aproximativa

error\_values(k) = norm(x - x\_exact);

end

% afisare solutie aproximativa si eroare finala

disp('Solutia aproximativa dupa iteratii:');

disp(x);

disp('Eroarea finala comparativ cu solutia exacta:');

disp(error\_values(end));

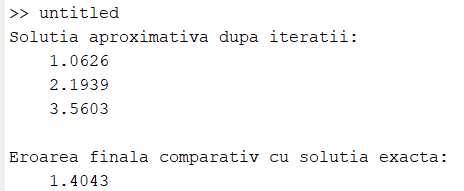
% graficul erorii

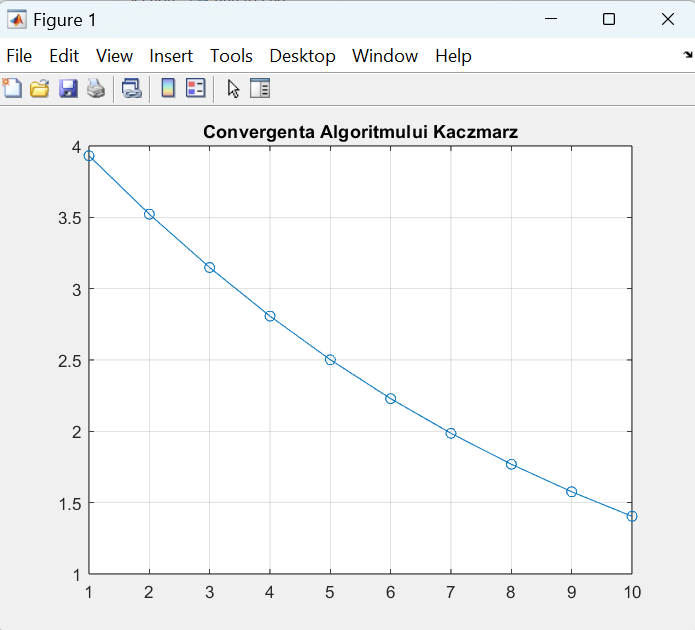
figure;

plot(1:num\_iter, error\_values, '-o')

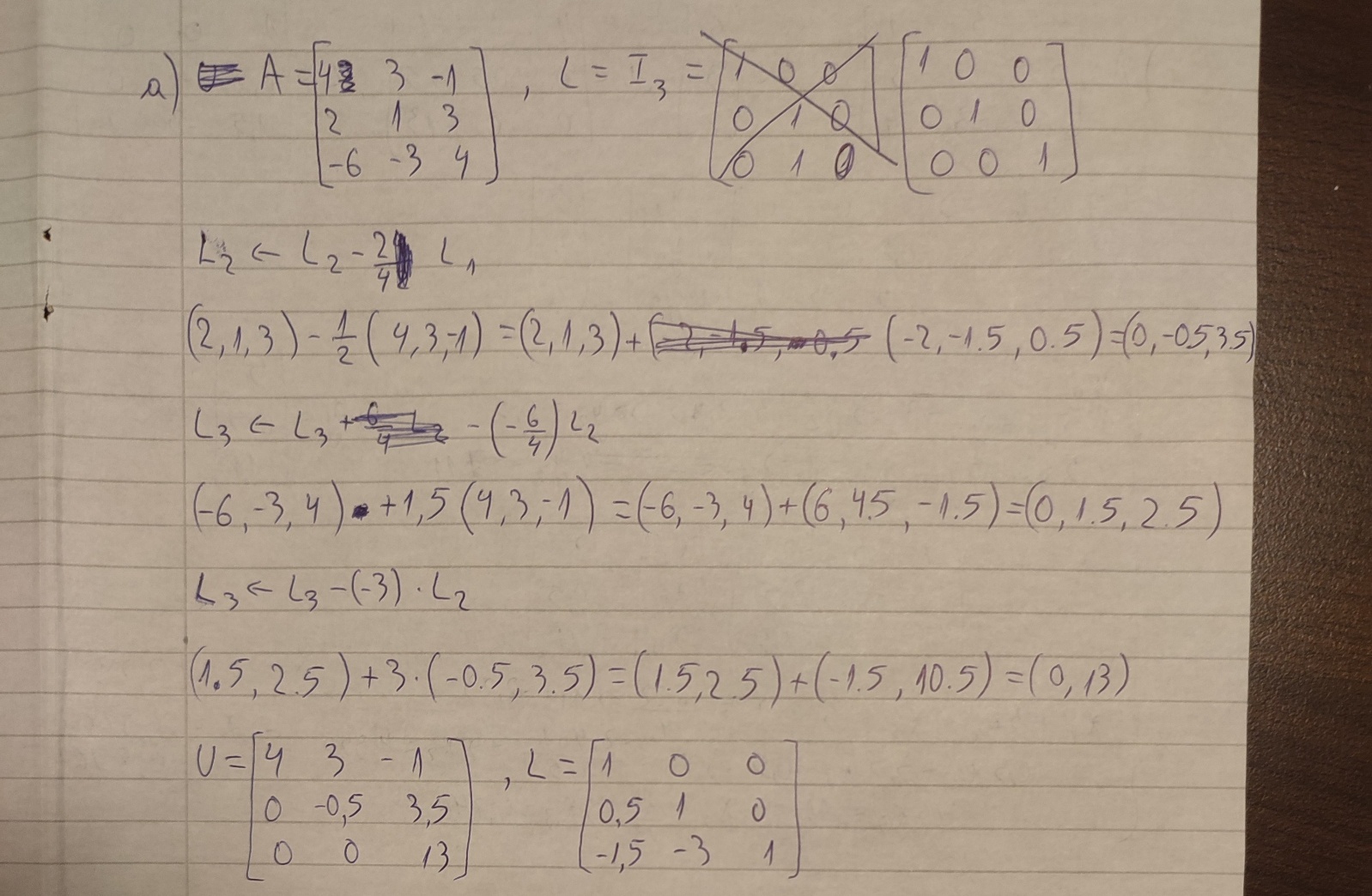
title('Convergenta Algoritmului Kaczmarz');

grid on;

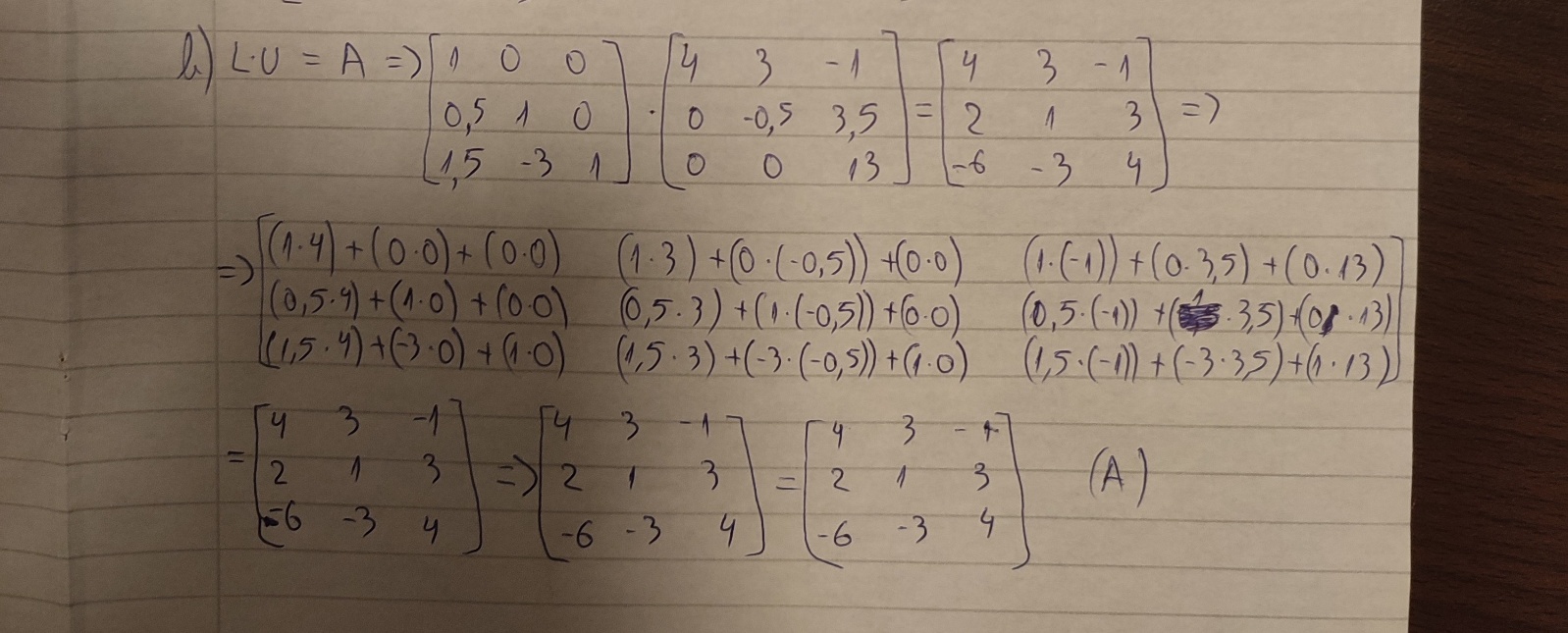




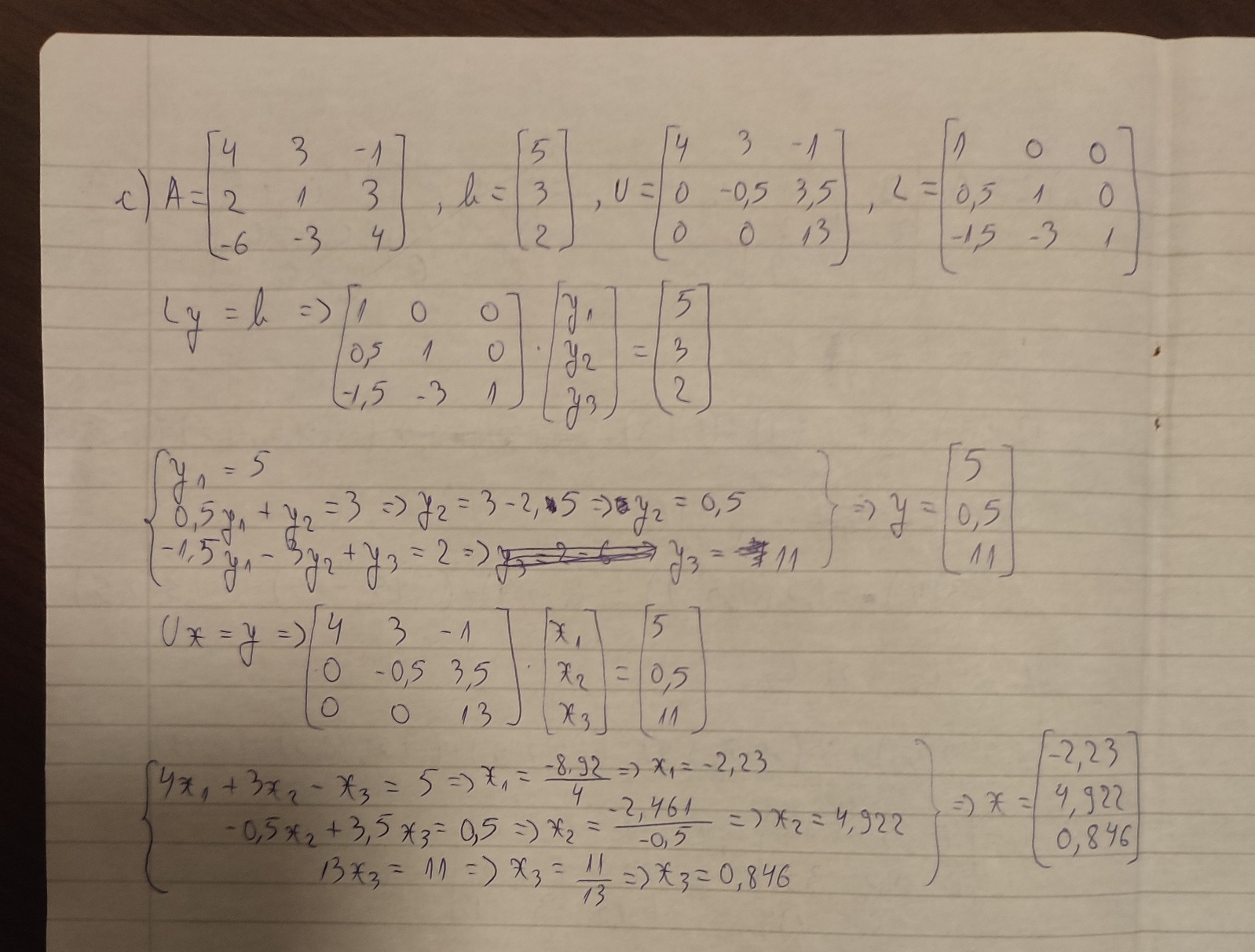














function dfp\_optimization()

% definirea functiei obiectiv

f = @(x) (x(1) - 3)^2 + (x(2) - 1)^4;

% gradientul functiei

grad\_f = @(x) [2\*(x(1)-3); 4\*(x(2)-1)^3];

% punct initial

x = [1; 2];

% matricea Hessiana inversa initiala (identitate)

H = eye(2);

% parametri algoritmului

tol = 1e-6;

max\_iter = 100;

alpha = 0.1; % pas fix

% iteratii DFP

for k = 1:max\_iter

g = grad\_f(x);

if norm(g) < tol

break;

end

% directia de cautare

p = -H \* g;

% actualizarea punctului

x\_new = x + alpha \* p;

% diferente

s = x\_new - x;

y = grad\_f(x\_new) - g;

% actualizarea matricei Hessiene inverse H folosind DFP

rho = 1 / (y' \* s);

if rho > 0

H = H - (H \* (s \* s') \* H) / (s' \* H \* s) + rho \* (y \* y');

end

% pregatirea pentru urmatoarea iteratie

x = x\_new;

end

% afisarea rezultatului final

fprintf('Minim aproximat: x = (%.6f, %.6f)\n', x(1), x(2));

fprintf('Valoarea funcției în minim: f(x) = %.6f\n', f(x));

% reprezentare grafica

[X, Y] = meshgrid(-1:0.1:5, -1:0.1:3);

Z = (X - 3).^2 + (Y - 1).^4;

contour(X, Y, Z, 50);

hold on;

plot(x(1), x(2), 'ro', 'MarkerSize', 10, 'MarkerFaceColor', 'r');

title('Metoda DFP - Minim local');

grid on;

end

