```
% metoda lui Jacobi
% A - matricea sistemului
% b - vectorul termenilor liberi
% err - toleranta (implicit 1e-3)
% x - solutia
% x0 - val initiala a vectorului solutie
function x = ex1_jacobi(A,b,err,maxit)

    [m,n]=size(A);
    x0 = zeros(size(b));
    if (m~=n) || (n~=length(b))
        error('dimensiuni ilegale')
    end

    %calculul lui T si c (pregatirea iteratiilor)
    M = diag(diag(A));
    N = M - A;
    T = inv(M)*N; %T=D^-1 *(L+U)
    c = inv(M)*b; %c=D^-1 *b
    alfa = norm(T,inf);
    x=x0(:);

    true = 1;
    while maxit>0
        %xi+1 = T*xi+c
        x0 = x;
        x = T*x0+c;

        if norm(x-x0,inf)<(1-alfa)/alfa*err
            true = 0;
            return
        end

        maxit=maxit-1;
    end

    fprintf("nu converge in numarul maxim de iteratii\n");
end
```

```
% metoda relaxarii (Successive OverRelaxation)
% A - matricea sistemului
% b - vectorul termenilor liberi
% omega - parametrul relaxarii
% x0 - vector de pornire
% err - toleranta (implicit 1e-3)
% x - solutia
function x = ex2_SOR(A,b,omega,err,maxit)
    if (omega<=0) || (omega>=2)
        error('parametrul relaxarii ilegal')
```

```
    end

    [m,n]=size(A);
    x0=zeros(size(b));
    if (m~=n) || (n~=length(b))
        error('dimensiuni ilegale')
    end

    %calculul lui T si c (pregatirea iteratiilor)
    %L=tril(A,-1): doar elem. cu i>j
    M = 1/omega*diag(diag(A))+tril(A,-1);
    N = M-A;
    T = M\N;
    c = M\b;
    alfa = norm(T,inf);
    x = x0(:);

    true = 1;
    while maxit>0
        x0 = x;
        x = T*x0+c;
        if norm(x-x0,inf)<(1-alfa)/alfa*err
            true = 0;
            return
        end

        maxit=maxit-1;
    end

    fprintf("nu converge in numarul maxim de iteratii\n");
end
%-------------------------------------------------------------------------------
--------------

% determina valoarea optima a parametrului relaxarii
% A - matricea sistemului
function omega=ex2_find_omega(A)
    %determin matricea Jacobi
    M = diag(diag(A));
    N = M-A;
    T = M\N;
    e = eig(T);

    %raza spectrala a matricei Jacobi
    rt = max(abs(e));

    omega = 2/(1+sqrt(1-rt^2));
end
```

```
% genereaza primul sistem
% n - dimensiunea sistemului
% A - matricea sistemului: tridiagonala cu 5 pe diag principala, -1 pe
                                                % subdiagonale si 0 in rest
% b - matricea termenilor liberi
function [A,b] = ex3_generare_sistem_1(n)
    o1=ones(n-1,1);

    %rara
    A=spdiags([-ones(n,1),5*ones(n,1),-ones(n,1)],-1:1,n,n);

    %o face densa pt afisare
    fprintf("A = \n");
    disp(full(A));

    b=A*ones(n,1);
end
%-------------------------------------------------------------------------------
--------------

% genereaza al doilea sistem
% n - dimensiunea sistemului
% A - matricea sistemului
% b - matricea termenilor liberi
function [A,b] = ex3_generare_sistem_2(n)
    %(x,y,z) vector toate elem. x, de lungime y, nr. coloane z
    v = repmat(5,n,1);
    u = repmat(-1,n-1,1);
    u1 = repmat(-1,n-3,1);

    A = diag(v) + diag(u,1) + diag(u,-1) + diag(u1,3) + diag(u1,-3);

    fprintf("A = \n");
    disp(full(A));

    b = repmat(1,n,1);
    b(1) = 3; b(2) = 2; b(3) = 2;
    b(n) = 3; b(n-1) = 2; b(n-2) = 2;
end
%-------------------------------------------------------------------------------
--------------

% Rezolvati sistemele cu toate metodele implementate.

n = 10;
err = 10^-2;
maxit=10;

%PRIMUL SISTEM
```

```
fprintf("PRIMUL SISTEM\n")
```

```
PRIMUL SISTEM
```

```
[A, b] = ex3_generare_sistem_1(n);
```

```
A =
     5    -1     0     0     0     0     0     0     0     0
    -1     5    -1     0     0     0     0     0     0     0
     0    -1     5    -1     0     0     0     0     0     0
     0     0    -1     5    -1     0     0     0     0     0
     0     0     0    -1     5    -1     0     0     0     0
     0     0     0     0    -1     5    -1     0     0     0
     0     0     0     0     0    -1     5    -1     0     0
     0     0     0     0     0     0    -1     5    -1     0
     0     0     0     0     0     0     0    -1     5    -1
     0     0     0     0     0     0     0     0    -1     5
```

```
fprintf("b:\n");
```

```
b:
```

```
disp(b);
```

```
     4
     3
     3
     3
     3
     3
     3
     3
     3
     4
```

```
tic
x_jacobi_1 = ex1_jacobi(A,b,err,maxit);
toc
```

```
Elapsed time is 0.002543 seconds.
```

```
fprintf("JACOBI\n");
```

```
JACOBI
```

```
disp(x_jacobi_1);
```

```
    0.9987
    0.9978
    0.9968
    0.9964
    0.9961
    0.9961
    0.9964
    0.9968
    0.9978
    0.9987
```

```
tic
```

```
x_gauss_seidel_1 = ex4_gauss_seidel(A,b,err,maxit);
toc
```

```
Elapsed time is 0.001919 seconds.
```

```
omega_1 = ex2_find_omega(A);
fprintf("GAUSS SEIDEL\n");
```

```
GAUSS SEIDEL
```

```
disp(x_gauss_seidel_1);
```

```
    0.9971
    0.9963
    0.9962
    0.9961
    0.9961
    0.9961
    0.9977
    0.9991
    0.9997
    0.9999
```

```
tic
x_sor_1 = ex2_SOR(A,b,omega_1,err,maxit);
toc
```

```
Elapsed time is 0.001931 seconds.
```

```
fprintf("SOR\n")
```

```
SOR
```

```
disp(x_sor_1);
```

```
    0.9984
    0.9981
    0.9980
    0.9980
    0.9980
    0.9980
    0.9998
    1.0000
    1.0000
    1.0000
```

```
%AL DOILEA SISTEM
fprintf("\n\n\nAL DOILEA SISTEM\n")
```

```
AL DOILEA SISTEM
```

```
[A1, b1] = ex3_generare_sistem_2(n);
```

```
A =
     5    -1     0    -1     0     0     0     0     0     0
    -1     5    -1     0    -1     0     0     0     0     0
```

```
     0    -1     5    -1     0    -1     0     0     0     0
    -1     0    -1     5    -1     0    -1     0     0     0
     0    -1     0    -1     5    -1     0    -1     0     0
     0     0    -1     0    -1     5    -1     0    -1     0
     0     0     0    -1     0    -1     5    -1     0    -1
     0     0     0     0    -1     0    -1     5    -1     0
     0     0     0     0     0    -1     0    -1     5    -1
     0     0     0     0     0     0    -1     0    -1     5
```

```
fprintf("b:\n");
```

```
b:
```

```
disp(b1);
```

```
     3
     2
     2
     1
     1
     1
     1
     2
     2
     3
```

```
x_jacobi_2 = ex1_jacobi(A1,b1,err,maxit);
```

```
nu converge in numarul maxim de iteratii
```

```
fprintf("JACOBI\n");
```

```
JACOBI
```

```
disp(x_jacobi_2);
```

```
    0.9884
    0.9838
    0.9812
    0.9770
    0.9756
    0.9756
    0.9770
    0.9812
    0.9838
    0.9884
```

```
x_gauss_seidel_2 = ex4_gauss_seidel(A1,b1,err,maxit);
omega2 = ex2_find_omega(A1);
fprintf("GAUSS SEIDEL\n");
```

```
GAUSS SEIDEL
```

```
disp(x_gauss_seidel_2);
```

```
    0.9963
    0.9957
    0.9957
```

```
    0.9955
    0.9959
    0.9965
    0.9972
    0.9980
    0.9985
    0.9991
```

```
x_sor_2 = ex2_SOR(A1,b1,omega_1,err,maxit);
fprintf("SOR\n")
```

```
SOR
```

```
disp(x_sor_2);
```

```
    0.9979
    0.9976
    0.9977
    0.9976
    0.9979
    0.9982
    0.9986
    0.9990
    0.9993
    0.9996
```

```
% metoda Gauss_Seidel
% A - matricea sistemului
% b - vectorul termenilor liberi
% err - toleranta (implicit 1e-3)
% z - solutia
function z = ex4_gauss_seidel(A, b, err, maxit)
    [m, n] = size(A);
    if (m ~= n) || (n ~= length(b))
        error('dimensiuni ilegale')
    end

    % Initialize variables
    x = zeros(n, 1);
    M = tril(A);
    N = M - A;
    T = M \ N;
    c = M \ b;
    alfa = norm(T, inf);

    % Iterative solution
    while maxit > 0
        x_new = T * x + c;  % Update x
        if norm(x_new - x, inf) < (1 - alfa) / alfa * err
            z = x_new;  % Converged solution
            return
        end
        x = x_new;  % Update x for the next iteration
```

```
        maxit = maxit - 1;
    end

    % If max iterations reached, return the last computed value
    z = x;
    fprintf("nu converge in numarul maxim de iteratii\n");
end


%-------------------------------------------------------------------------
--------------
function [A,b] = ex4_mat_diag_dominanta(n)
    max_element = 50;
    A = randi([-max_element,max_element],n);

    %modulul sumei pe randuri + smth pentru diagonala
    out = sum(abs(A),2) + randi(max_element, n,1);

    for i=1:n
        A(i,i) = out(i);
    end

    %solutia 1,2,....,n
    x=(1:n)';
    b = A*x;
end

%-------------------------------------------------------------------------
--------------

n = 10;
err = 10^-2;
maxit=50;
```

```
A = 4×4
    14    35    22    10
     3    16    20    25
     5    48    39    23
    42     2    40    33
b = 4×1
    81
    64
   115
   117
```

```
[A,b] = ex4_mat_diag_dominanta(n);
fprintf("A:\n");
disp(A);
```

```
    14    35    22    10
     3    16    20    25
```

```
     5    48    39    23
    42     2    40    33
```

```
fprintf("b:\n")
disp(b);
```

```
    81
    64
   115
   117
```

```
x_jacobi = ex1_jacobi(A,b,err,maxit);
```

```
nu converge in numarul maxim de iteratii
```

```
fprintf("JACOBI: \n");
```

```
JACOBI:
```

```
disp(x_jacobi);
```

```
   1.0e+45 *

   -1.4392
   -0.9256
   -0.6708
   -0.9590
```

```
x_gauss_seidel = ex4_gauss_seidel(A,b,err,maxit);
```

```
nu converge in numarul maxim de iteratii
```

```
fprintf("GAUSS SEIDEL: \n");
```

```
GAUSS SEIDEL:
```

```
disp(x_gauss_seidel);
```

```
   1.0e+27 *

    1.4743
    0.9441
   -0.5867
   -1.2225
```

```
omega = ex2_find_omega(A);
x_sor = ex2_SOR(A,b,omega,err,maxit)
```

```
nu converge in numarul maxim de iteratii
x_sor = 4×1 complex
10^34 ×
  -1.7746 + 2.0302i
   3.8891 - 0.6231i
  -0.7814 + 4.2468i
  -3.3731 - 7.4288i
```

```
fprintf("SOR: \n");
```

```
SOR:
```

```
disp(x_sor);
```

```
   1.0e+34 *

  -1.7746 + 2.0302i
   3.8891 - 0.6231i
  -0.7814 + 4.2468i
  -3.3731 - 7.4288i
```

```

```