EPFL

Numerical Analysis and Computational Mathematics

Fall Semester 2024 – CSE Section

Prof. Laura Grigori

Assistant: Israa Fakih

Session 9 – November 13, 2024

Solutions – Linear systems: direct and iterative methods

Exercise I (MATLAB)

a) We recall the following properties.

Proposition 1 *For a non-singular matrix $A \in \mathbb{R}^{n\times n}$, its LU Gauss factorization exists and is unique if and only if the principal submatrices A_i of A of order $i = 1, \dots, n-1$ are non-singular.*

Proposition 2 *If one of the following hypothesis for the matrix $A \in \mathbb{R}^{n\times n}$ holds:*

- *A is strictly diagonally dominant by row,*
- *A is strictly diagonally dominant by column,*
- *A is symmetric and positive definite,*

then there exists a unique LU Gauss factorization of A.

We observe that the matrix A_1 does not satisfy the hypotheses of Proposition 2. So, in order to verify the existence and uniqueness of the LU Gauss factorization (without the necessity of performing the pivoting technique), we need to rely on Proposition 1. We use the following MATLAB commands to define A_1 and compute the determinants of the principal submatrices of A:

```
n = 4;
A1 = diag( 3 * ones( n, 1 ), 0 ) + diag( - 2 * ones( n - 1, 1 ), 1 ) ...
       + diag( - 1 * ones( n - 1, 1 ), - 1 );
A1_det_sub = [];
for p = 1 : n - 1
    A1_det_sub = [ A1_det_sub, det( A1( 1 : p, 1 : p ) ) ];
end
A1_det_sub
```

```
% A1_det_sub =
%    3.0000    7.0000   15.0000
```

We notice that the principal submatrices $A_{1,i}$ of A_1 for $i = 1, \ldots, n-1$ are non-singular ($\det(A_{1,i}) \neq 0$ for $i = 1, \ldots, n-1$) and, in addition, A_1 is non-singular ($\det(A_1) = 31 \neq 0$). We conclude that there exists a unique LU Gauss factorization of A_1 and performing the pivoting technique is not strictly necessary to compute the matrices L and U.

We observe that the matrix A_2 is not strictly diagonally dominant by row or column, but it is real symmetric. We recall that a symmetric, real square matrix is positive definite if and only if all its eigenvalues are strictly positive. To verify if A_2 is positive definite, we compute its eigenvalues in MATLAB:

```
A2 = hilb( n );
format short e
A2_eig = eig(A2)'
% A2_eig =
%     9.6702e-05   6.7383e-03   1.6914e-01   1.5002e+00
format
```

We deduce the all the eigenvalues are real and positive, and so A_2 is real, symmetric, and positive definite. From Proposition 2 we deduce that there exists a unique LU Gauss factorization of A_2 (without pivoting).

b) By recalling that the MATLAB function `lu` may use the pivoting technique even when not strictly necessary, we use the following MATLAB commands for the linear systems associated to A_1 and A_2. For A_1:

```
n = 9;  % Matrix A_1
A1 = diag( 3 * ones( n, 1 ), 0 ) + diag( - 2 * ones( n - 1, 1 ), 1 ) ...
       + diag( - 1 * ones( n - 1, 1 ), - 1 );
x1_ex = ones( n, 1 );       b1 = A1 * x1_ex;
[ L1, U1, P1 ] = lu( A1 );
[ y1 ] = forward_substitutions( L1, P1 * b1 );
[ x1 ] = backward_substitutions( U1, y1 );
e1_rel = norm( x1_ex - x1 ) / norm( x1_ex )
% e1_rel =
%    1.2820e-16
r1_rel = norm( b1 - A1 * x1 ) / norm( b1 )
% r1_rel =
%    3.5804e-16
A1_cond = cond(A1)
% A1_cond =
%    26.3742
```

We see that the relative error is $e_{\text{rel},1} = 1.2820 \cdot 10^{-16}$, while the relative residual is $r_{\text{rel},1} = 5.3475 \cdot 10^{-16}$, both very small and close to machine epsilon. The result can be explained by observing that the matrix A_1 is well-conditioned. Indeed, when using a direct method to approximate the solution of the linear system, we have the following general relation between the relative error and relative residual:

$$e_{\text{rel}} = \frac{\|\mathbf{x} - \widehat{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq K_2(A)\, r_{\text{rel}}, \tag{1}$$

where $\widehat{\mathbf{x}}$ is the approximate solution of the linear system $A\mathbf{x} = \mathbf{b}$ and $\mathbf{r} = \mathbf{b} - A\widehat{\mathbf{x}}$, with $r_{\text{rel}} = \frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}$. When considering the linear system associated to A_1, which is well-conditioned, the right hand side of the previous error estimate is small ($K_2(A_1)\, r_{\text{rel},1} = 1.4104 \cdot 10^{-14}$). So we expect the relative error on the solution $\mathbf{x}_1$ to be small as well, as numerically observed.

We repeat the MATLAB commands for A_2:

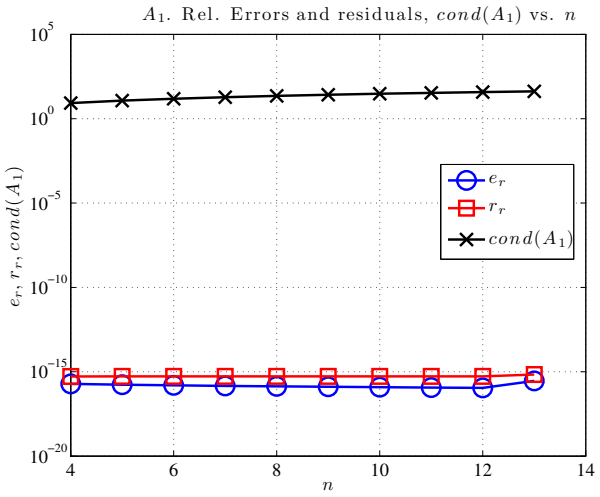
```
A2 = hilb( n );   % Matrix A_2
x2_ex = x1_ex;   b2 = A2 * x2_ex;
[ L2, U2, P2 ] = lu( A2 );
[ y2 ] = forward_substitutions( L2, P2 * b2 );
[ x2 ] = backward_substitutions( U2, y2 );
e2_rel = norm( x2_ex - x2 ) / norm( x2_ex )
% e2_rel =
%      1.0906e-05
r2_rel = norm( b2 - A2 * x2 ) / norm( b2 )
% r2_rel =
%      1.3242e-16
A2_cond = cond(A2)
% A2_cond =
%      4.9315e+11
```

We see that the relative residual is $r_{\text{rel},2} = 1.6218 \cdot 10^{-16}$, still very small and close to machine epsilon. On the other hand, the relative error is $e_{\text{rel},2} = 2.5435 \cdot 10^{-6}$, about ten orders of magnitude larger than $r_{\text{rel},2}$. The reason for this can be found by looking at the large condition number $K_2(A_2)$. Using error estimate (1), we deduce that $K_2(A_2)\, r_{\text{rel},2} = 7.9977 \cdot 10^{-5}$, so that $e_{\text{rel},2}$ may be significantly larger than the relative residual $r_{\text{rel},2}$. When the matrix is ill-conditioned, the relative residual is not a satisfactory error indicator, and the approximate solution may be affected by a large error.

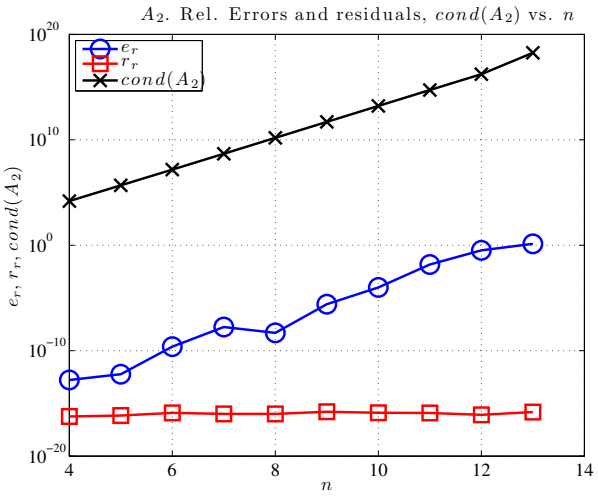
c) We use the following MATLAB commands to plot relative errors, relative residuals, and condition numbers of the linear systems (see Figure 1).

```
c1_v = [ ];   e1_rel_v = [ ];   r1_rel_v = [ ];
c2_v = [ ];   e2_rel_v = [ ];   r2_rel_v = [ ];
n_vect = 4 : 13;
for n = n_vect
    A1 = diag( 3 * ones( n, 1 ), 0 ) + diag( - 2 * ones( n - 1, 1 ), 1 ) ...
      + diag( - 1 * ones( n - 1, 1 ), - 1 );
    x1_ex = ones( n, 1 );     b1 = A1 * x1_ex;
    [ L1, U1, P1 ] = lu( A1 );
    [ y1 ] = forward_substitutions( L1, P1 * b1 );
    [ x1 ] = backward_substitutions( U1, y1 );
    e1_rel_v = [ e1_rel_v, norm( x1_ex - x1 ) / norm( x1_ex ) ];
    r1_rel_v = [ r1_rel_v, norm( b1 - A1 * x1 ) / norm( b1 ) ];
    c1_v = [ c1_v, cond( A1 ) ];

    A2 = hilb( n );
    x2_ex = x1_ex;    b2 = A2 * x2_ex;
    [ L2, U2, P2 ] = lu( A2 );
    [ y2 ] = forward_substitutions( L2, P2 * b2 );
    [ x2 ] = backward_substitutions( U2, y2 );
    e2_rel_v = [ e2_rel_v, norm( x2_ex - x2 ) / norm( x2_ex ) ];
    r2_rel_v = [ r2_rel_v, norm( b2 - A2 * x2 ) / norm( b2 ) ];
```



A_1



A_2

Figure 1: Relative errors (blue), relative residuals (red), and condition numbers (black) for the linear systems associated to A_1 (left) and A_2 (right) vs. n.

```
    c2_v = [ c2_v, cond( A2 ) ];
end
figure( 11 ); % for A_1
semilogy( n_vect, e1_rel_v, '-ob', n_vect, r1_rel_v, '-sr', ...
          n_vect, c1_v, '-xk'  );
legend('e_{rel}', 'r_{rel}', 'cond(A_1)');
figure( 12 );  % for A_2
semilogy( n_vect, e2_rel_v, '-ob', n_vect, r2_rel_v, '-sr', ...
          n_vect, c2_v, '-xk'  );
legend('e_{rel}', 'r_{rel}', 'cond(A_2)');
```

In Figure 1 (left), we observe that the relative errors and residuals corresponding to the solutions of the first linear system are small for all sizes n. Also, the matrix A_1 remains well-conditioned. Consequently, the approximate solution $\widehat{\mathbf{x}}_1$ for $n = 13$ is satisfactory: the relative error with respect to the exact solution $\mathbf{x}_1$ is of order 10^{-15}.

In Figure 1 (right), we observe that the relative errors and condition numbers of A_2 increase for increasing values of n. The second linear system is ill-conditioned and the conditioning of A_2 worsens as its size increases. Even if the relative residual is small for all the considered n, the relative error may be large in presence of large condition numbers of the matrix (see point b)). For instance, for $n = 13$, we obtain that $e_{\text{rel},2} = 1.3583$, with $K_2(A_2) = 1.7590 \cdot 10^{18}$ and $r_{\text{rel},2} = 1.4961 \cdot 10^{-16}$. The relative residual is not a satisfactory error indicator in this case.

Exercise II (MATLAB)

a) We recall the following Proposition for the convergence of iterative methods for the solution of the linear system $A\mathbf{x} = \mathbf{b}$, with $A \in \mathbb{R}^{n\times n}$ non-singular and $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$, starting from any initial solution $\mathbf{x}^{(0)} \in \mathbb{R}^n$.

Proposition 3 *Let us consider an iterative method in the form* $\mathbf{x}^{(k+1)} = B\mathbf{x}^{(k)} + \mathbf{g}$*, for* $k = 0, 1, \ldots$*, where the iteration matrix is* $B = I - P^{-1}A$ *for some invertible preconditioning matrix* P *and* $\mathbf{g} = P^{-1}\mathbf{b}$*, with an initial solution* $\mathbf{x}^{(0)}$*. Then, the iterative method converges to the solution* $\mathbf{x}$ *for any* $\mathbf{x}^{(0)}$ *if and only if the spectral radius*[1] *of the iteration matrix* B*, denoted by* $\rho(B)$*, is* < 1*. The smaller* $\rho(B)$ *is, the faster the convergence of the method.*

Specifically, for the Jacobi and Gauss-Seidel methods we can determine a priori the convergence properties, provided that certain hypotheses are satisfied by A; we recall the following Propositions.

Proposition 4 *If the nonsingular matrix* A *is strictly diagonally dominant by row, then the Jacobi and Gauss-Seidel methods converge (for any* $\mathbf{x}^{(0)}$*).*

Proposition 5 *If the nonsingular matrix* A *is real symmetric and positive definite, then the Gauss-Seidel method converges (for any* $\mathbf{x}^{(0)}$*).*

Proposition 6 *If the nonsingular matrix* A *is tridiagonal, and all its diagonal elements are non-zero, then the Jacobi and Gauss-Seidel methods are either both divergent or both convergent. In the latter case, the Gauss-Seidel method converges faster; more precisely, the spectral radius of the iteration matrix associated to the Gauss-Seidel method is equal to the square of that of Jacobi.*

We start from the matrix A_1. We observe that it is not strictly diagonally dominant nor tridiagonal. As such, we cannot infer a priori the convergence of the Jacobi method, but we need to calculate the spectral radius $\rho_{1,J} = \rho(B_{1,J})$ of the associated iteration matrix $B_{1,J}$. Similarly, the matrix A_1 is not symmetric. So, in order to establish the convergence of the Gauss-Seidel method, we need to calculate the spectral radius $\rho_{1,GS} = \rho(B_{1,GS})$.

We use the following MATLAB commands:

```
n1 = 3;
A1 = [ 3 -2 1; 2 1.65 -1; 0 1 4 ];
P1_J = diag( diag( A1 ) );  % preconditioning matrix Jacobi
B1_J = eye( n1, n1 ) - inv( P1_J ) * A1; % iteration matrix Jacobi
rho1_J = max(abs(eig( B1_J ))),   % spectral radius B1_J
% rho1_J =
%     0.9851
P1_GS = diag( diag( A1 ) ) - ( - 1 * tril( A1, -1 ) ); % prec.matr.Gauss-Seidel
B1_GS = eye( n1, n1 ) - inv( P1_GS ) * A1; % iteration matrix Gauss-Seidel
rho1_GS = max(abs(eig( B1_GS ))),   % spectral radius B1_GS
% rho1_GS =
%     1.0606
```

From Proposition 3, since $\rho_{1,J} = 0.9851 < 1$, the Jacobi method converges for all the choices of the initial solution $\mathbf{x}^{(0)}$. Still, the convergence is expected to be slow due to the fact that $\rho_{1,J}$ is very close to 1. Conversely, the Gauss-Seidel method does not converge to the solution of the linear systems associated to the matrix A_1 for all the choices of $\mathbf{x}^{(0)}$, since $\rho_{1,GS} = 1.0606 > 1$.

We move to A_2. A_2 is symmetric and positive definite, since all its eigenvalues are strictly positive. We verify this with MATLAB:

[1] We recall that for a square matrix C of size n, the spectral radius is $\rho(C) = \max_{i=1,\ldots,n} |\lambda_i(C)|$, where $\{\lambda_i(C)\}_{i=1}^{n}$ are the eigenvalues of C.

```
n2 = 3;
A2 = [ 5 -3 -2; -3 3 0; -2 0 4 ];
eig_A2 = eig(A2)'
% eig_A2 =
%     0.4103    3.7126    7.8771
```

In virtue of Proposition 5, we deduce that the Gauss-Seidel method is convergent without the need to explicitly compute $\rho_{2,GS} = \rho(B_{2,GS})$ (below, we compute $\rho_{2,GS}$ only for verification purposes). However, we need to explicitly calculate the spectral radius of the iteration matrix associated to the Jacobi method, according to Proposition 3, since the hypotheses of Propositions 4, 5, and 6 are not satisfied. We use the following MATLAB commands:

```
P2_J = diag( diag( A2 ) ); % Jacobi
B2_J = eye( n2, n2 ) - inv( P2_J ) * A2;
rho2_J = max(abs(eig( B2_J )))
% rho2_J =
%     0.8944
P2_GS = diag( diag( A2 ) ) - ( - 1 * tril( A2, -1 ) ); % Gauss-Seidel
B2_GS = eye( n2, n2 ) - inv( P2_GS ) * A2;
rho2_GS = max(abs(eig( B2_GS )))
% rho2_GS =
%     0.8000
```

We confirm that the Gauss-Seidel method is convergent, since $\rho_{2,GS} = 0.8 < 1$. The Jacobi method is also convergent, since $\rho_{2,J} = 0.8944 < 1$. We observe that the convergence of the Gauss-Seidel method is expected to be faster than that of the Jacobi method for the linear systems associated to A_2, since, in this case, $\rho_{2,GS} < \rho_{2,J} < 1$.

We consider A_3 last. The matrix is tridiagonal, and all diagonal elements are non-zero. So, according to Proposition 6, the Gauss-Seidel and Jacobi methods are either both convergent or both divergent. As such, it suffices to verify the convergence of one of the two methods. For instance, we observe that A_3 is symmetric and positive definite (all the eigenvalues are positive with the minimum being 2.001). So, the Gauss-Seidel method is convergent according to Proposition 5. Therefore, from Proposition 6, the Jacobi method also converges, and $\rho_{3,GS} = \rho_{3,J}^2$. We define the matrix A_3 and verify these conclusions by means of the following MATLAB commands:

```
n3 = 100;
A3 = diag( 4 * ones( n3, 1 ), 0 ) + diag( - 1 * ones( n3 - 1, 1 ), 1 ) + ...
     diag( - 1 * ones( n3 - 1, 1 ), -1 );
P3_J = diag( diag( A3 ) );  % Jacobi
B3_J = eye( n3, n3 ) - inv( P3_J ) * A3;
rho3_J = max(abs(eig( B3_J )))
% rho3_J =
%     0.4998
P3_GS = diag( diag( A3 ) ) - ( - 1 * tril( A3, -1 ) ); % Gauss-Seidel
B3_GS = eye( n3, n3 ) - inv( P3_GS ) * A3;
rho3_GS = max(abs(eig( B3_GS )))
% rho3_GS =
%     0.2498
```

We verify that $\rho_{3,GS} = \rho_{3,J}^2 = 0.2498 < 1$. So, both Jacobi and Gauss-Seidel methods converge, with the Gauss-Seidel method being the fastest.

b) We implement the MATLAB functions `jacobi.m` and `gauss_seidel.m` as follows:

```
function [ x, k, res ] = jacobi( A, b, x0, tol, kmax )
% JACOBI solve the linear system A x = b by means of the
% Jacobi iterative method; diagonal elements of A must be nonzero.
% Stopping criterion based on the residual.
%  [ x, k, res ] = jacobi( A, b, x0, tol, kmax )
%  Inputs:  A    = matrix (square matrix)
%           b    = vector (right hand side of the linear system)
%           x0   = initial solution (colum vector)
%           tol  = tolerence for the stopping driterion based on residual
%           kmax = maximum number of iterations
%  Outputs: x    = solution vector (column vector)
%           k    = number of iterations at convergence
%           res  = value of the norm of the residual at convergence
%

n = size( A, 1 );

k = 0;
x = x0;
res = norm( A * x - b );

x_old = x0;

while( k < kmax && res > tol )
    for i = 1 : n
        j_v_old = [ 1 : i - 1, i + 1 : n ];
        x( i ) = 1 / A( i, i ) * ( b( i ) ...
                                   - A( i, j_v_old ) * x_old( j_v_old ) );
    end
    res = norm( A * x - b );
    k = k + 1;
    x_old = x;
end

return
```

```
function [ x, k, res ] = gauss_seidel( A, b, x0, tol, kmax )
% GAUSS_SEIDEL solve the linear system A x = b by means
% of the Gauss-Seidel iterative method; diagonal elements of A
% must be nonzero. Stopping criterion based on the residual.
%  [ x, k, res ] = gauss_seidel( A, b, x0, tol, kmax )
%  Inputs:  A    = matrix (square matrix)
%           b    = vector (right hand side of the linear system)
%           x0   = initial solution (colum vector)
%           tol  = tolerence for the stopping driterion based on residual
%           kmax = maximum number of iterations
%  Outputs: x    = solution vector (column vector)
%           k    = number of iterations at convergence
%           res  = value of the norm of the residual at convergence
%
```

```
n = size( A, 1 );

k = 0;
x = x0;
res = norm( A * x - b );

x_old = x0;

while( k < kmax && res > tol )
    for i = 1 : n
        j_v = 1 : i - 1;
        j_v_old = i + 1 : n;
        x( i ) = 1 / A( i, i ) * ( b( i ) ...
                                   - A( i, j_v ) * x( j_v ) ...
                                   - A( i, j_v_old ) * x_old( j_v_old ) );
    end
    res = norm( A * x - b );
    k = k + 1;
    x_old = x;
end

return
```

c) We start by solving the linear system $A_1\mathbf{x}_1 = \mathbf{b}_1$, for which only the Jacobi method converges. We use the following MATLAB commands:

```
x1 = ones( n1, 1 );     b1 = A1 * x1;     x0 = zeros( n1, 1 );
[ x1_J, k1_J, res1_J ] = jacobi( A1, b1, x0, 1e-6, 1000 );
err1_J = norm( x1 - x1_J ),    k1_J,    res1_J
% err1_J =
%    2.7745e-07
% k1_J =
%    969
% res1_J =
%    9.6699e-07
```

We see that the Jacobi method converges to the solution $\mathbf{x}_{1,J}$ in $k_{1,J} = 969$ iterations, with corresponding error $e_{1,J}^{(969)} = \|\mathbf{x}_1 - \mathbf{x}_{1,J}^{(969)}\| = 2.7745 \cdot 10^{-7}$ and norm of the residual $r_{1,J}^{(969)} = \|\mathbf{r}_{1,J}^{(969)}\| = 9.6699 \cdot 10^{-7}$. The convergence is very slow due to the fact that the spectral radius of the iteration matrix $B_{1,J}$ is $\rho_{1,J} = 0.9851$, very close to 1.

We repeat the procedure for the linear system $A_2\mathbf{x}_2 = \mathbf{b}_2$, for which both the Jacobi and Gauss-Seidel methods are convergent. We use the following MATLAB commands:

```
x2 = ones( n2, 1 );   b2 = A2 * x2;   x0 = zeros( n2, 1 );
[ x2_J, k2_J, res2_J ] = jacobi( A2, b2, x0, 1e-6, 1000 );  % Jacobi
err2_J = norm( x2 - x2_J ),    k2_J,   res2_J
% err2_J =
%    1.6855e-06
% k2_J =
%    124
% res2_J =
%    8.8408e-07
[ x2_GS, k2_GS, res2_GS ] = gauss_seidel( A2, b2, x0, 1e-6, 1000 ); % Gauss-S.
```

```
err2_GS = norm( x2 - x2_GS ),   k2_GS,   res2_GS
% err2_GS =
%    1.4712e-06
% k2_GS =
%    63
% res2_GS =
%    9.8080e-07
```

For the Jacobi method, we obtain $k_{2,J} = 124$, $e_{2,J}^{(124)} = 1.6855 \cdot 10^{-6}$, $r_{2,J}^{(124)} = 8.8408 \cdot 10^{-7}$. For the Gauss-Seidel method, we obtain $k_{2,GS} = 63$, $e_{2,GS}^{(63)} = 1.4712 \cdot 10^{-6}$, $r_{2,GS}^{(63)} = 9.8080 \cdot 10^{-7}$. The convergence of the Gauss-Seidel method is faster than that of the Jacobi method since $\rho_{2,GS} < \rho_{2,J} < 1$ (see point a)).

Finally, for the linear system $A_3\mathbf{x}_3 = \mathbf{b}_3$, we use the following MATLAB commands:

```
x3 = ones( n3, 1 );    b3 = A3 * x3;    x0 = zeros( n3, 1 );
[ x3_J, k3_J, res3_J ] = jacobi( A3, b3, x0, 1e-6, 1000 );  % Jacobi
err3_J = norm( x3 - x3_J ), k3_J, res3_J
% err3_J =
%    2.8220e-07
% k3_J =
%    25
% res3_J =
%    5.6510e-07
[ x3_GS, k3_GS, res3_GS ] = gauss_seidel( A3, b3, x0, 1e-6, 1000 ); % Gauss-S.
err3_GS = norm( x3 - x3_GS ), k3_GS, res3_GS
% err3_GS =
%    2.1655e-07
% k3_GS =
%    16
% res3_GS =
%    4.3476e-07
```

For the Jacobi method, we obtain $k_{3,J} = 25$, $e_{3,J}^{(25)} = 2.8220 \cdot 10^{-7}$, $r_{3,J}^{(25)} = 5.6510 \cdot 10^{-7}$. For the Gauss-Seidel method, we obtain $k_{3,GS} = 16$, $e_{3,GS}^{(16)} = 2.1655 \cdot 10^{-7}$, $r_{3,GS}^{(16)} = 4.3476 \cdot 10^{-7}$. The convergence of the two methods is relatively fast since $\rho_{3,J}$ and $\rho_{3,GS}$ are significantly smaller than 1 (see point a)). Moreover, the convergence of the Gauss-Seidel method is faster than that of the Jacobi method since $\rho_{3,GS} < \rho_{3,J}$.

d) We can not establish a priori the convergence properties of the Jacobi and Gauss-Seidel methods according to Propositions 4, 5, or 6, since their hypotheses are not satisfied. We compute with MATLAB the spectral radii $\rho_{4,J}$ and $\rho_{4,GS}$ of the iterations matrices $B_{4,J}$ and $B_{4,GS}$ associated to the methods. Since the matrix A_4 depends on the parameter γ, the spectal radii $\rho_{4,J}$ and $\rho_{4,GS}$ do too. We use the following MATLAB commands to obtain the result shown in Figure 2:

```
gamma_v = linspace( -10, 25, 2001 );
rho4_J_v = [ ];   rho4_GS_v = [ ];
for gamma = gamma_v
    n4 = 4;
    A4 = [ 8 gamma -2 -1; -2 2 -gamma -3; -1 -2 18 -18; -1 -3 -7 25 ];
    P4_J = diag( diag( A4 ) );  % Jacobi
```

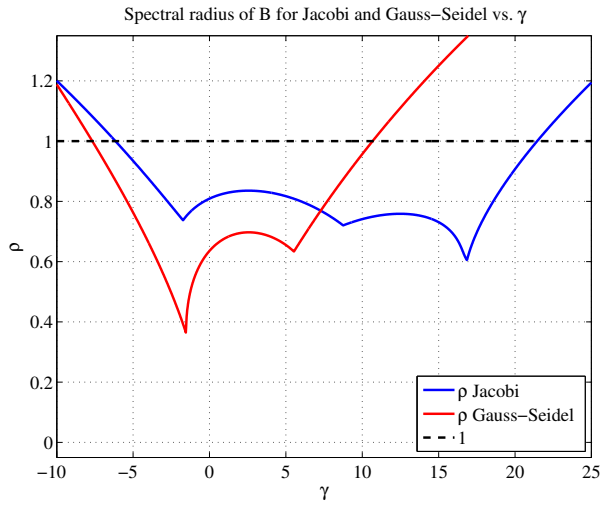


Figure 2: Spectral radii $\rho_{4,J}$ and $\rho_{4,GS}$ of the iteration matrices of the Jacobi and Gauss-Seidel methods vs. γ.

```
    B4_J = eye( n4, n4 ) - inv( P4_J ) * A4;
    rho4_J_v = [ rho4_J_v, max( abs( eig( B4_J ) ) ) ];
    P4_GS = diag( diag( A4 ) ) - ( - 1 * tril( A4, -1 ) );  % Gauss-S.
    B4_GS = eye( n4, n4 ) - inv( P4_GS ) * A4;
    rho4_GS_v = [ rho4_GS_v, max( abs( eig( B4_GS ) ) ) ];
end
plot( gamma_v, rho4_J_v, '-b', gamma_v, rho4_GS_v, '-r', ...
    gamma_v, 1 + 0 * gamma_v, '--k' );
legend('\rho Jacobi','\rho Gauss-Seidel','1' );
```

From Figure 2 we deduce the following: for any initial solution $\mathbf{x}^{(0)}$,

- the Jacobi method converges for $-6.159 < \gamma < 21.45$;
- the Gauss-Seidel method converges for $-7.673 < \gamma < 10.68$;
- the Gauss-Seidel method converges faster than the Jacobi method for $-6.159 < \gamma < 7.280$;
- the Jacobi method converges faster than the Gauss-Seidel method for $7.280 < \gamma < 10.68$.

For $\gamma = 0$, both the Jacobi and Gauss-Seidel methods are convergent, but the convergence of the Gauss-Seidel method is faster than that of Jacobi since $\rho_{4,GS} < \rho_{4,J} < 1$. With similar arguments, we conclude that Jacobi works best for $\gamma = 9$ ($\rho_{4,J} < \rho_{4,GS} < 1$). Finally, for $\gamma = 15$, we must select the Jacobi method, since the Gauss-Seidel method does not converge ($\rho_{4,J} < 1$, but $\rho_{4,GS} > 1$).