ENS409 Project 1 Report İlker Gül

Question 1

Find the solution for the system given below and compare to the value obtained using MATLAB's "\" operator. Let $\mathbf{x}_0 = [0, 0, -0.5, -0.5]^T$ and tolerance 10^{-6}.

$$A = \begin{bmatrix} 1 & -1 & 1 & -1 \\ -1 & 3 & -3 & 3 \\ 2 & -4 & 7 & -7 \\ 3 & 7 & -10 & 14 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 0 \\ 2 \\ -2 \\ -8 \end{bmatrix}$$

Solution

We are now going to apply 4 different algorithms for the question. Two of them are direct methods which are Gaussian Elimination and LU decomposition. I applied the algorithms with respect to lecture notes.

First, let me start with gaussian elimination.

```
function gaussianelimination
    A = [1 -1 1 -1;
    -1 3 -3 3;
    2 -4 7 -7;
    3 7 -10 14];

    b = [0 2 -2 -8]';
    augmented_matrix = [A, b];
    % Initially, we need to perform forward elimination
    % in order to receive the coefficients of unknown for backward
    % substitution
    disp('------- Gaussian Elimination -------')
    disp('augmented matrix before forward elimination:')
    disp(augmented_matrix)
    for iterorder = 1:4
    rowcoefficient = augmented_matrix(iterorder,iterorder);
       if iterorder ~= 4
          for remaining = iterorder+1:4
             iterleliminationnumber = augmented_matrix(remaining,iterorder);
             eliminationcoeff = (iterleliminationnumber/rowcoefficient)*(-
1);
             subtractionrow =
augmented_matrix(iterorder,:).*eliminationcoeff;
             augmented_matrix(remaining,:) =
augmented_matrix(remaining,:)+subtractionrow;
          end
       end
    end

    % Now, we can proceed with the back substitution
    value_vector = zeros(4, 1);
    value_vector(4,1) = augmented_matrix(4, 5)/augmented_matrix(4, 4);
    for i = 3:-1:1
```

```
        sum = 0;
        for j = 1:4
             sum = sum + augmented_matrix(i,j)*value_vector(j,1);
        end
        eq_result = augmented_matrix(i,5);
        x_entry = (eq_result - sum)/augmented_matrix(i,i);
        value_vector(i,1) = x_entry;
    end
    disp('augmented matrix after forward elimination:')
    disp(augmented_matrix)
    disp('solution for the system is as follows:')
    disp(value_vector);

end
```

In the function, I first applied forward elimination and after forward elimination I applied back substitution to find the values of the unknowns.

```
Exact result by the \ operation:
    1.0000
    1.0000
   -4.5000
   -4.5000

------- Gaussian Elimination -------
augmented matrix before forward elimination:
     1    -1     1    -1     0
    -1     3    -3     3     2
     2    -4     7    -7    -2
     3     7   -10    14    -8

augmented matrix after forward elimination:
     1    -1     1    -1     0
     0     2    -2     2     2
     0     0     3    -3     0
     0     0     0     4   -18

solution for the system is as follows:
    1.0000
    1.0000
   -4.5000
   -4.5000
```

When we applied the function we get the following results.

Now, let me continue with the LU decomposition.

```
function ludecomposition

    disp('------- LU Decomposition -------')
    A = [1 -1 1 -1;
    -1 3 -3 3;
    2 -4 7 -7;
    3 7 -10 14];

    b = [0 2 -2 -8]';
    Upper_triangular_matrix = A;
    Lower_triangular_matrix = eye(4);
    Uppercoefficients = [];
    %Now, we are going to perform LU decomposition
    %First, Find U
    for iterorder = 1:4
    rowcoefficient = Upper_triangular_matrix(iterorder,iterorder);
       if iterorder ~= 4
          for remaining = iterorder+1:4
             itereliminationnumber =
Upper_triangular_matrix(remaining,iterorder);
             eliminationcoeff = (itereliminationnumber/rowcoefficient)*(-
1);
             Uppercoefficients = [Uppercoefficients; eliminationcoeff*(-
1)];
             subtractionrow =
Upper_triangular_matrix(iterorder,:).*eliminationcoeff;
             Upper_triangular_matrix(remaining,:) =
Upper_triangular_matrix(remaining,:)+subtractionrow;
          end
       end
    end
    disp('Upper triangluar matrix after forward elimination:');
    disp(Upper_triangular_matrix);
    Lower_triangular_matrix(2,1) = Uppercoefficients(1);
    Lower_triangular_matrix(3,1) = Uppercoefficients(2);
    Lower_triangular_matrix(4,1) = Uppercoefficients(3);
    Lower_triangular_matrix(3,2) = Uppercoefficients(4);
    Lower_triangular_matrix(4,2) = Uppercoefficients(5);
    Lower_triangular_matrix(4,3) = Uppercoefficients(6);
    disp('Lower triangluar matrix after finding forward elimination
coefficients:');
    disp(Lower_triangular_matrix);



    z_vector = zeros(4, 1);
    z_vector(1,1) = b(1,1)/Lower_triangular_matrix(1,1);
    for i = 2:4
        sum = 0;
        for j = 1:i
             sum = sum + Lower_triangular_matrix(i,j)*z_vector(j,1);
        end
        eq_result = b(i,1);
        x_entry = (eq_result - sum)/Lower_triangular_matrix(i,i);
        z_vector(i,1) = x_entry;
    end

    value_vector = zeros(4, 1);
    value_vector(4,1) = z_vector(4,1)/Upper_triangular_matrix(4,4);
    for i = 3:-1:1
```

```
        sum = 0;
        for j = 1:4
             sum = sum + Upper_triangular_matrix(i,j)*value_vector(j,1);
        end
        eq_result = z_vector(i,1);
        x_entry = (eq_result - sum)/Upper_triangular_matrix(i,i);
        value_vector(i,1) = x_entry;
    end
    disp('solution after first step (z vector) is as follows:')
    disp(z_vector);
    disp('solution for the system is as follows:')
    disp(value_vector);

end
```

Here, I first calculated the upper triangular matrix. Then, I found the lower triangular matrix with respect to coefficients found in the process of Forward elimination.

Then, I applied the basis of LU decomposition, generated the z vector. Then, I found the values of unknowns by solving the Z vector.

```
------- LU Decomposition -------
Upper triangluar matrix after forward elimination:
     1    -1     1    -1
     0     2    -2     2
     0     0     3    -3
     0     0     0     4

Lower triangluar matrix after finding forward elimination coefficients:
     1     0     0     0
    -1     1     0     0
     2    -1     1     0
     3     5    -1     1

solution after first step (z vector) is as follows:
     0
     2
     0
   -18

solution for the system is as follows:
    1.0000
    1.0000
   -4.5000
   -4.5000
```

Then, I applied the iterative methods. First, I started with the Jacobi iteration

```
function jacobi
    disp('------- Jacobi Iterative Method -------')
    A = [1 -1 1 -1;
```

```
    -1 3 -3 3;
    2 -4 7 -7;
    3 7 -10 14];

    b = [0 2 -2 -8]';
    X0 = [0,0, -0.5, -0.5]';
    exact_result = A\b;
    D = eye(4);
    for i = 1:4
        D(i,i) = A(i,i);
    end
    disp('Diagonal matrix of Jacobi:');
    disp(D);

    L = zeros(4,4);
    L(2,1) = -A(2,1);
    L(3,1) = -A(3,1);
    L(3,2) = -A(3,2);
    L(4,1) = -A(4,1);
    L(4,2) = -A(4,2);
    L(4,3) = -A(4,3);
    disp('Lower Triangular matrix of Jacobi:');
    disp(L);

    U = zeros(4,4);
    U(1,2) = -A(1,2);
    U(1,3) = -A(1,3);
    U(1,4) = -A(1,4);
    U(2,3) = -A(2,3);
    U(2,4) = -A(2,4);
    U(3,4) = -A(3,4);
    disp('Upper Triangular matrix of Jacobi:');
    disp(U);

    tol = 1;
    iter = 0;
    temp = L+ U;
    JacobiEstimate = [];
    while (tol > 10^(-6) && iter < 71)
        iter = iter + 1;
        JacobiEstimate = D\temp*X0 + D\b;
        Error = abs(X0 - JacobiEstimate);
%          disp(iter)
%          disp('Absoulute error of each variable with respect to exact
value by the \ operation:')
%          disp(Error);
        tol = max(Error);
        X0 = JacobiEstimate;

    end
    if(iter ~= 71)
        disp('Iteration number is as follows:');
        disp(iter);
    end
    disp('Jacobi estimate values are as follows:');
    disp(JacobiEstimate);
end
```

Here, I used approximate error for comparing with tolerance and I decied a threshold as 71. If the number of iteration reached to 71 and we still did not get the error less than the given tolerance, we terminate the iteration. I applied the formula given in the lecture notes to apply Jacobi method.

```
------- Jacobi Iterative Method -------
Diagonal matrix of Jacobi:
     1     0     0     0
     0     3     0     0
     0     0     7     0
     0     0     0    14

Lower Triangular matrix of Jacobi:
     0     0     0     0
     1     0     0     0
    -2     4     0     0
    -3    -7    10     0

Upper Triangular matrix of Jacobi:
     0     1    -1     1
     0     0     3    -3
     0     0     0     7
     0     0     0     0

Jacobi estimate values are as follows:
   1.0e+15 *

   -4.3050
    3.2520
   -2.6654
    1.5110
```

Since this matrix is not a diagonally dominant matrix. If it was a diagonally dominant we can state this is going to converge. However, as you can see by using Jacobi this matrix does not converge but diverge.

Now, let me use Gauss-Seidel. Here, it differs from Jacobi since it uses also the current findings for calculating the value of unknowns.

```
function gauss_seidel
    disp('------- Gauss-Seidel -------')
    A = [1 -1 1 -1;
    -1 3 -3 3;
```

```
    2 -4 7 -7;
    3 7 -10 14];

    b = [0 2 -2 -8]';
    exact_result = A\b;
    tol = 1;
    iter = 1;
    X = zeros(4,30);
    X(:,1) = [0,0, -0.5, -0.5]';
    while (tol > 10^(-6) && iter < 71)
        iter = iter + 1;
        X(1,iter) = (b(1)-(A(1,2)*X(2, iter -1))-(A(1,3)*X(3, iter -1))-
(A(1,4)*X(4, iter -1)))/A(1,1);
        X(2,iter) = (b(2)-(A(2,1)*X(1, iter))-(A(2,3)*X(3, iter -1))-
(A(2,4)*X(4, iter -1)))/A(2,2);
        X(3,iter) = (b(3)-(A(3,1)*X(1, iter))-(A(3,2)*X(2, iter))-
(A(3,4)*X(4, iter -1)))/A(3,3);
        X(4,iter) = (b(4)-(A(4,1)*X(1, iter))-(A(4,2)*X(2, iter))-
(A(4,3)*X(3, iter)))/A(4,4);
        Error = abs(X(:,iter-1) - X(:,iter));

%          disp(iter -1);
%          disp('Absoulute error of each variable with respect to exact
value by the \ operation:')
%          disp(Error);

        tol = max(Error);
    end
    if(iter ~= 71)
        disp('Iteration number is as follows:');
        disp(iter);
    end
    disp('Gauss-Seidel estimate values are as follows:');
    disp(X(:,iter));

end
```

Here, I used approximate error. Then, we can see that we use the current findings of the unknown matrix to calculate the values of the other variables in the variable matrix.

```
------- Gauss-Seidel -------
Gauss-Seidel estimate values are as follows:
    1.0000
    1.0000
   -4.5000
   -4.5000
```

Here, it beautifully converges.

Question 2

Initialize the random number generator by typing `rand('seed', xyzw)` where `xyzw` are the last four digits of your student ID number. Generate a random $m \times m$ matrix A and a corresponding $m \times 1$ vector $\mathbf{b}$ for $m = 3, 7, 9$. Solve $A\mathbf{x} = \mathbf{b}$ and find an inverse of A. For the iterative methods set $\mathbf{x}_0$ to zero vector and the tolerance to 10^{-4}. Comment on the results.

Solution

Now, let's take a look at the q2main.m. Here, we are going implement 3 different matrices.

```
clear;
clc;
rand('seed', 6352);
A = rand(3,3);
b = rand(3,1);
A2 = rand(7,7);
b2 = rand(7,1);
A3 = rand(9,9);
b3 = rand(9,1);
result = A2\b2;
fprintf("A matrix:\n")
disp(A2);
fprintf("Exact result by the \\ operation:\n")
disp(result);
gaussianelimination_q2(A2,b2);
fprintf("\n")
ludecomposition_q2(A2,b2);
fprintf("\n")
jacobi_q2(A2,b2);
fprintf("\n")
gauss_seidel_q2(A2,b2);
```

Here, we seed the rand and then generated matrices that are 3x3, 7x7 and 9x9. Then, we sent them to the functions and try to get the expected results.

```
A matrix:
    0.9976    0.7305    0.6302
    0.9928    0.6785    0.9939
    0.7114    0.0620    0.4945

Exact result by the \ operation:
    0.0958
    0.7669
   -0.0767

------- Gaussian Elimination -------
augmented matrix before forward elimination:
    0.9976    0.7305    0.6302    0.6075
    0.9928    0.6785    0.9939    0.5392
    0.7114    0.0620    0.4945    0.0777

augmented matrix after forward elimination:
    0.9976    0.7305    0.6302    0.6075
         0   -0.0485    0.3667   -0.0654
         0         0   -3.4224    0.2626

solution for the system is as follows:
    0.0958
    0.7669
   -0.0767
```

```
------- LU Decomposition -------
Upper triangluar matrix after forward elimination:
    0.9976    0.7305    0.6302
         0   -0.0485    0.3667
         0         0   -3.4224

Lower triangluar matrix after finding forward elimination coefficients:
    1.0000         0         0
    0.9952    1.0000         0
    0.7132    9.4557    1.0000

solution after first step (z vector) is as follows:
    0.6075
   -0.0654
    0.2626

solution for the system is as follows:
    0.0958
    0.7669
   -0.0767
```

```
------- Jacobi Iterative Method -------
Diagonal matrix of Jacobi:
    0.9976         0         0
         0    0.6785         0
         0         0    0.4945

Lower Triangular matrix of Jacobi:
         0         0         0
   -0.9928         0         0
   -0.7114   -0.0620         0

Upper Triangular matrix of Jacobi:
         0   -0.7305   -0.6302
         0         0   -0.9939
         0         0         0

Jacobi estimate values are as follows:
   1.0e+11 *

   -3.2711
   -5.2461
   -3.0436
```

```
------- Gauss-Seidel -------
Iteration number is as follows:
    71

Gauss-Seidel estimate values are as follows:
   1.0e+92 *

    0.1240
   -0.1300
    0.0836
   -0.1065
   -0.3803
    2.2934
   -2.1212
```

```
A matrix:
    0.5498    0.9212    0.3845    0.5204    0.4634    0.8351    0.0716
    0.6913    0.9418    0.7352    0.8898    0.6202    0.0656    0.3106
    0.2091    0.3035    0.8189    0.1637    0.9042    0.8585    0.2095
    0.2409    0.5873    0.6179    0.7276    0.6656    0.1979    0.8166
    0.2136    0.0246    0.3352    0.0416    0.1932    0.0990    0.3725
    0.9219    0.4726    0.9495    0.9768    0.9423    0.1340    0.2280
    0.4142    0.3448    0.3829    0.7119    0.4486    0.8463    0.8170

Exact result by the \ operation:
    2.5142
    1.2026
   -4.7853
   -2.3324
    4.9221
   -1.2889
    1.5964

------- Gaussian Elimination -------
augmented matrix before forward elimination:
    0.5498    0.9212    0.3845    0.5204    0.4634    0.8351    0.0716    0.7550
    0.6913    0.9418    0.7352    0.8898    0.6202    0.0656    0.3106    0.7412
    0.2091    0.3035    0.8189    0.1637    0.9042    0.8585    0.2095    0.2689
    0.2409    0.5873    0.6179    0.7276    0.6656    0.1979    0.8166    0.9828
    0.2136    0.0246    0.3352    0.0416    0.1932    0.0990    0.3725    0.2837
    0.9219    0.4726    0.9495    0.9768    0.9423    0.1340    0.2280    0.8939
    0.4142    0.3448    0.3829    0.7119    0.4486    0.8463    0.8170    0.3848

augmented matrix after forward elimination:
    0.5498    0.9212    0.3845    0.5204    0.4634    0.8351    0.0716    0.7550
         0   -0.2165    0.2516    0.2354    0.0374   -0.9845    0.2206   -0.2082
         0         0    0.6183   -0.0851    0.7199    0.7538    0.1346    0.0267
         0         0         0    0.7906   -0.2776   -1.8116    0.8280    0.4466
         0         0         0         0   -0.0031    0.2741    0.6258    0.6307
    0.0000         0         0         0         0   60.6780  134.3913  136.3388
   -0.0000         0         0   -0.0000         0         0   -1.4930   -2.3835

solution for the system is as follows:
    2.5142
    1.2026
   -4.7853
   -2.3324
    4.9221
   -1.2889
    1.5964
```

```
------- LU Decomposition -------
Upper triangluar matrix after forward elimination:
    0.5498    0.9212    0.3845    0.5204    0.4634    0.8351    0.0716
         0   -0.2165    0.2516    0.2354    0.0374   -0.9845    0.2206
         0         0    0.6183   -0.0851    0.7199    0.7538    0.1346
         0         0         0    0.7906   -0.2776   -1.8116    0.8280
         0         0         0         0   -0.0031    0.2741    0.6258
    0.0000         0         0         0         0   60.6780  134.3913
   -0.0000         0         0   -0.0000         0         0   -1.4930

Lower triangluar matrix after finding forward elimination coefficients:
    1.0000         0         0         0         0         0         0
    1.2574    1.0000         0         0         0         0         0
    0.3803    0.2162    1.0000         0         0         0         0
    0.4381   -0.8485    1.0722    1.0000         0         0         0
    0.3885    1.5393   -0.3261   -0.6965    1.0000         0         0
    1.6769    4.9515   -1.5226   -1.5064 -214.0065    1.0000         0
    0.7534    1.6126   -0.5056   -0.1299 -119.4159    0.5715    1.0000

solution after first step (z vector) is as follows:
    0.7550
   -0.2082
    0.0267
    0.4466
    0.6307
  136.3388
   -2.3835

solution for the system is as follows:
    2.5142
    1.2026
   -4.7853
   -2.3324
    4.9221
   -1.2889
    1.5964
```

```
------- Gauss-Seidel -------
Iteration number is as follows:
    71

Gauss-Seidel estimate values are as follows:
   1.0e+07 *

    7.8556
   -6.2640
    5.9824
   -1.9580
   -4.4973
   -4.6740
    4.8748
```

```
------- Jacobi Iterative Method -------
Diagonal matrix of Jacobi:
    0.5498         0         0         0         0         0         0
         0    0.9418         0         0         0         0         0
         0         0    0.8189         0         0         0         0
         0         0         0    0.7276         0         0         0
         0         0         0         0    0.1932         0         0
         0         0         0         0         0    0.1340         0
         0         0         0         0         0         0    0.8170

Lower Triangular matrix of Jacobi:
         0         0         0         0         0         0         0
   -0.6913         0         0         0         0         0         0
   -0.2091   -0.3035         0         0         0         0         0
   -0.2409   -0.5873   -0.6179         0         0         0         0
   -0.2136   -0.0246   -0.3352   -0.0416         0         0         0
   -0.9219   -0.4726   -0.9495   -0.9768   -0.9423         0         0
   -0.4142   -0.3448   -0.3829   -0.7119   -0.4486   -0.8463         0

Upper Triangular matrix of Jacobi:
         0   -0.9212   -0.3845   -0.5204   -0.4634   -0.8351   -0.0716
         0         0   -0.7352   -0.8898   -0.6202   -0.0656   -0.3106
         0         0         0   -0.1637   -0.9042   -0.8585   -0.2095
         0         0         0         0   -0.6656   -0.1979   -0.8166
         0         0         0         0         0   -0.0990   -0.3725
         0         0         0         0         0         0   -0.2280
         0         0         0         0         0         0         0

Jacobi estimate values are as follows:
   1.0e+42 *

   -1.5170
   -0.5679
   -1.0309
   -0.7556
   -1.1530
   -4.8889
   -1.0894
```

```
------- Gauss-Seidel -------
Iteration number is as follows:
    71

Gauss-Seidel estimate values are as follows:
  1.0e+134 *

    0.0039
   -0.0283
   -0.0233
    0.0370
   -0.0103
    0.0603
   -0.7529
    0.7121
    1.3675
```

As you can see in each case since they are not diagonally dominant matrices, they did not converge.

```
    1.6527   -1.9440    1.8010
    1.3044    0.2711   -2.2074
   -2.5413    2.7629   -0.2922

    3.1282   -3.7927   -2.1432    1.3294    3.0961    2.2230   -1.6437
    1.4470   -0.2712   -0.4629    1.1547    0.5986   -0.4534   -1.2056
   -5.7444    8.5774    3.4983   -5.6165   -1.0903   -3.6394    3.4721
   -3.0762    3.6329    1.1519   -2.2366   -3.4303   -0.9681    2.6630
    5.0025   -8.2693   -2.2588    5.8793    0.9647    3.8100   -4.0952
   -0.7778    1.3426    0.9401   -1.5890   -0.7601   -0.8313    1.4835
    1.2352   -1.9987   -1.0950    1.8377    1.9356    0.3828   -0.6698

    3.3521    1.8209    0.2836   -7.1899    0.8982   -2.9138    7.4738   -6.1047    5.8457
   -3.7344   -1.3826    0.7172    8.5224   -3.0267    4.2272   -9.9789    8.5770   -8.8109
    5.6062    2.2804   -0.6091  -12.3653    1.3350   -4.0339   14.4918  -11.9475   12.2302
   -5.7712   -1.4874    0.7056   11.0580   -2.4608    3.4476  -12.3021   10.7812   -9.5238
   -1.0384   -0.4320    1.2629   -1.3628   -1.4905    1.8444   -0.4147    1.4030   -1.1331
    1.1649    0.2709    0.2579   -0.9688   -0.0969   -0.9920    2.7360   -2.1925    1.6015
   -0.3020   -1.4789   -0.9454    3.4708    1.4760    0.5265   -3.3574    2.1210   -2.0057
   -0.2822   -0.1323   -1.2044    1.6303    1.9217   -0.5707   -1.3248    1.0354   -1.0258
   -0.7172    0.3988    0.2134    1.3652    0.4548    0.0013   -1.6622    0.4559   -0.8482
```

These are the inverse matrices of those 3 random matrices

Question 3

Consider the system $A\mathbf{x} = \mathbf{b}$ where A is $m \times m$ symmetric matrix and $\mathbf{b}$ is $m \times 1$ vector, given as:

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 2 & -1 \\ 0 & 0 & 0 & 0 & \cdots & -1 & 2 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

The exact solution of the system is $\mathbf{x} = [1 \quad 1 \quad \cdots \quad 1]^T$. Use iterative techniques to obtain the solution of the system. Set $\mathbf{x}_0$ to zero vector and the tolerance to 10^{-6}. Plot the number of iterations necessary for the Jacobi and Gauss-Seidel methods to converge vs. m, where $m = 5, 15, 25, \ldots, 75$. Comment on the results.

Initially, I have solved the problem by direct methods. Here, I will give the values of the direct methods and the actual method consecutively.

Solution

```
Exact result by the \ operation:
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000

------- Gaussian Elimination -------
augmented matrix before forward elimination:
     2    -1     0     0     0     1
    -1     2    -1     0     0     0
     0    -1     2    -1     0     0
     0     0    -1     2    -1     0
     0     0     0    -1     2     1

augmented matrix after forward elimination:
    2.0000   -1.0000         0         0         0    1.0000
         0    1.5000   -1.0000         0         0    0.5000
         0         0    1.3333   -1.0000         0    0.3333
         0         0         0    1.2500   -1.0000    0.2500
         0         0         0         0    1.2000    1.2000

solution for the system is as follows:
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
```

```
------- LU Decomposition -------
Upper triangluar matrix after forward elimination:
    2.0000   -1.0000         0         0         0
         0    1.5000   -1.0000         0         0
         0         0    1.3333   -1.0000         0
         0         0         0    1.2500   -1.0000
         0         0         0         0    1.2000

Lower triangluar matrix after finding forward elimination coefficients:
    1.0000         0         0         0         0
   -0.5000    1.0000         0         0         0
         0   -0.6667    1.0000         0         0
         0         0   -0.7500    1.0000         0
         0         0         0   -0.8000    1.0000

solution after first step (z vector) is as follows:
    1.0000
    0.5000
    0.3333
    0.2500
    1.2000

solution for the system is as follows:
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
```

```
Exact result by the \ operation:
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000

------- Gaussian Elimination -------
solution for the system is as follows:
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
```

```
------- LU Decomposition -------
solution after first step (z vector) is as follows:
    1.0000
    0.5000
    0.3333
    0.2500
    0.2000
    0.1667
    0.1429
    0.1250
    0.1111
    1.1000

solution for the system is as follows:
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
```

```
Exact result by the \ operation:
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000

------- Gaussian Elimination -------
solution for the system is as follows:
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
```

```
------- LU Decomposition -------
solution after first step (z vector) is as follows:
    1.0000
    0.5000
    0.3333
    0.2500
    0.2000
    0.1667
    0.1429
    0.1250
    0.1111
    0.1000
    0.0909
    0.0833
    0.0769
    0.0714
    0.0667
    0.0625
    0.0588
    0.0556
    0.0526
    1.0500

solution for the system is as follows:
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
```

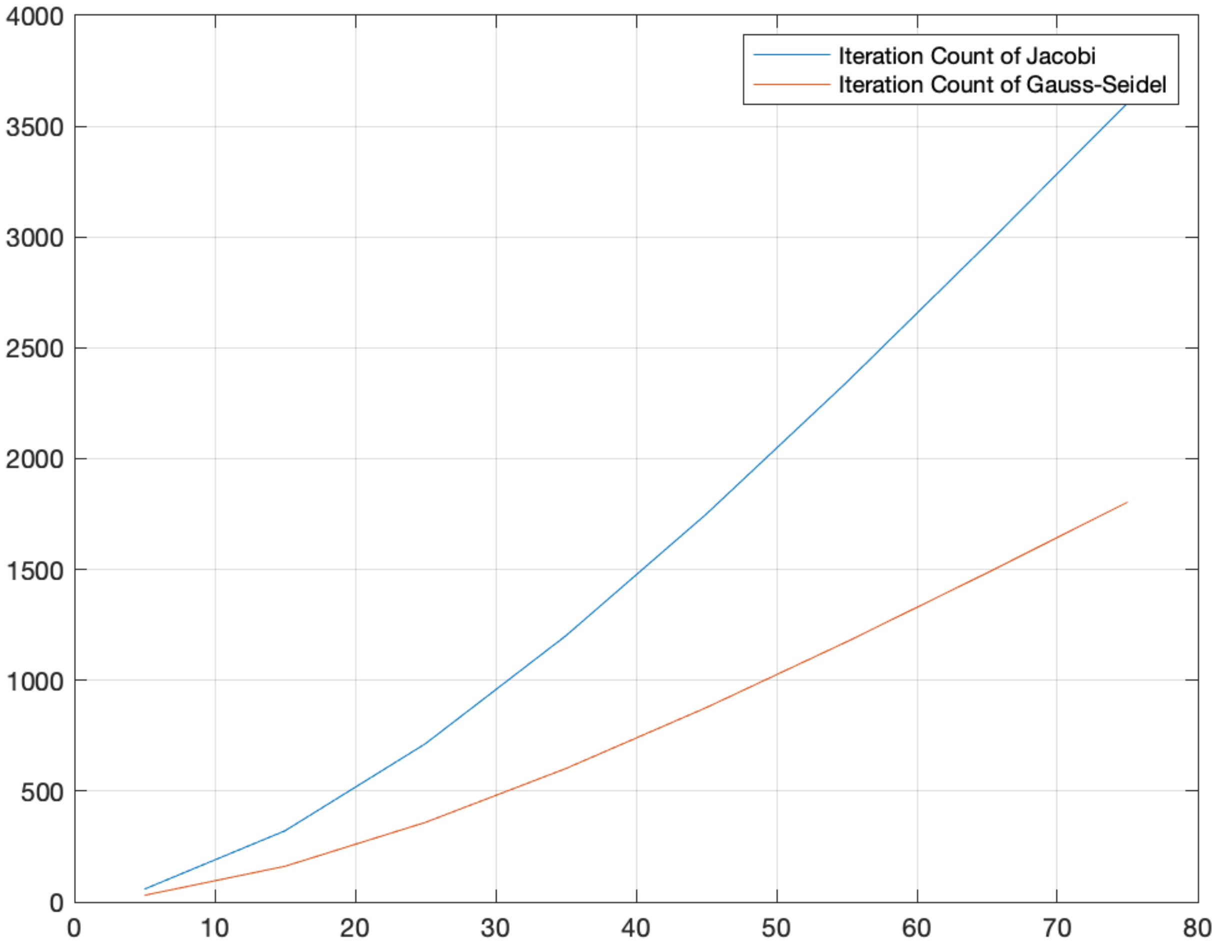
As you can see by using direct methods we can find the exact answers. Now, let’s look at the iterative methods.

Here is an example of 15x15 matrix. As you can see it is diagonally dominant matrix. Thus, we expect a convergence in iterative methods.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	-1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	-1	2	-1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	-1	2	-1	0	0	0	0	0	0	0	0	0	0	0
4	0	0	-1	2	-1	0	0	0	0	0	0	0	0	0	0
5	0	0	0	-1	2	-1	0	0	0	0	0	0	0	0	0
6	0	0	0	0	-1	2	-1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	-1	2	-1	0	0	0	0	0	0	0
8	0	0	0	0	0	0	-1	2	-1	0	0	0	0	0	0
9	0	0	0	0	0	0	0	-1	2	-1	0	0	0	0	0
10	0	0	0	0	0	0	0	0	-1	2	-1	0	0	0	0
11	0	0	0	0	0	0	0	0	0	-1	2	-1	0	0	0
12	0	0	0	0	0	0	0	0	0	0	-1	2	-1	0	0
13	0	0	0	0	0	0	0	0	0	0	0	-1	2	-1	0
14	0	0	0	0	0	0	0	0	0	0	0	0	-1	2	-1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	2

```
clear;
clc;

Tempx = [];
Tempy = [];
Tempxg = [];
Tempyg = [];

count = 0;
while count < 1
    count = count + 1;
    m = input('Please enter the iteration number:');

    A = zeros(m,m);
    fprintf('\n');
    for i = 1:m
        if(i == 1)
            A(i,i) = 2;
            A(i,i+1) = -1;
        elseif(i == m)
            A(i,i-1) = -1;
            A(i,i) = 2;
        else
            A(i,i-1) = -1;
            A(i,i) = 2;
            A(i,i+1) = -1;
        end
    end
    b = zeros(m,1);
    b(1,1) = 1;
    b(m,1) = 1;
    result = A\b;
    fprintf("Exact result by the \\ operation:\n")
    disp(result);
    gaussianelimination_q3(A,b)
    fprintf("\n")
    ludecomposition_q3(A,b);
    fprintf("\n")
    [t1,t2] = jacobi_q3(A,b);
    fprintf("\n")
    [t3,t4] = gauss_seidel_q3(A,b);
```

```
    Tempx = [Tempx; t1];
    Tempy = [Tempy; t2];
    Tempxg = [Tempxg; t3];
    Tempyg = [Tempyg; t4];

end

plot(Tempx, Tempy, 'DisplayName','Iteration Count of Jacobi');
hold on;
plot(Tempxg, Tempyg,'DisplayName','Iteration Count of Gauss-Seidel');
grid on;
legend;
```



Here is the following graph

Both method converged beautifully.