Group Members:  
Ali Valiyev, Murathan Bilgen, Orhan Ashrafov, Hüseyin Eren Demirtaş.

Solutions of Exercise 1

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | 17 | 10.049875621120890 | 7 |
|  | 18 | 10.488088481701515 | 7 |
|  | 10 | 6.480740698407860 | 5 |

Solutions of Exercise 2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Matrix Norm (p) | Vector Norm (q) | S/U |  |  |  |  |  |  |
| 1 | 1 | \* | 0.9099 | 0.8581 | 0.9537 | 0.8672 | 0.8788 | 0.8799 |
| 1 | 2 | \* | 0.8894 | 0.8507 | 0.8541 | 0.9792 | 1.0049 | 0.9021 |
| 1 | inf | \* | 0.7624 | 0.7600 | 0.7223 | 1.2471 | 1.3046 | 1.0491 |
| 2 | 1 | \* | 1.0210 | 0.9628 | 1.0701 | 0.8509 | 0.8622 | 0.8633 |
| 2 | 2 | \* | 0.9980 | 0.9545 | 0.9583 | 0.9608 | 0.9860 | 0.8851 |
| 2 | inf | \* | 0.8555 | 0.8528 | 0.8104 | 1.2236 | 1.2801 | 1.0293 |
| inf | 1 | \* | 1.0273 | 0.9688 | 1.0768 | 0.5651 | 0.5726 | 0.5734 |
| inf | 2 | \* | 1.0042 | 0.9605 | 0.9643 | 0.6381 | 0.6548 | 0.5878 |
| inf | inf | \* | 0.8608 | 0.8581 | 0.8155 | 0.8126 | 0.8501 | 0.6836 |

\* I made a new table for compatibility so that I can analyze all combinations. The table is at below.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Matrix Norm (p) | Vector Norm (q) |  |  |  |  |  |  |
| 1 | 1 | S | S | S | S | S | S |
| 1 | 2 | S | S | S | S | U | S |
| 1 | inf | S | S | S | U | U | U |
| 2 | 1 | U | S | U | S | S | S |
| 2 | 2 | S | S | S | S | S | S |
| 2 | inf | S | S | S | U | U | U |
| inf | 1 | U | S | U | S | S | S |
| inf | 2 | U | S | S | S | S | S |
| inf | inf | S | S | S | S | S | S |

Solutions of Exercise 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case | Residual error | Large/Small | xTrue | Solution error | Large/Small |
| 1 | 9.9998e-13 | Small | [0 ; 0] | 1.4142 | Large |
| 2 | 1.4142e-05 | Small | [1 ; 0] | 1.0000e-05 | Small |
| 3 | 281.4285 | Large | [1 ; 0] | 140.7160 | Large |
| 4 | 1.0000e+09 | Large | [1 ; 1] | 1.0000e-03 | Small |

Solutions of Exercise 4

|  |  |  |
| --- | --- | --- |
| n | Time | Ratio |
| 161 | 0.0011 | 1.4636e+05 |
| 321 | 0.0032 | 1.0031e+05 |
| 641 | 0.0158 | 4.0570e+04 |
| 1281 | 0.0670 | 1.9119e+04 |
| 2561 | 0.3141 | 8.1535e+03 |
| 5121 | 2.1050 | 2.4328e+03 |
| 10241 | 14.5561 | 703.5538 |

From the table, we can see that, the order of the time complexity of the solution of linear system by finding inverses is , so it is .

|  |  |  |  |
| --- | --- | --- | --- |
| n | Time | Ratio | Inverse/Solution (Time) |
| 161 | 4.1330e-04 | 1.4636e+05 | 2.6615 |
| 321 | 8.9530e-04 | 2.7672e+04 | 3.5742 |
| 641 | 0.0036 | 4.0570e+04 | 4.3889 |
| 1281 | 0.0216 | 1.9119e+04 | 3.1019 |
| 2561 | 0.1431 | 8.1535e+03 | 2.1950 |
| 5121 | 0.7373 | 2.4328e+03 | 2.8550 |
| 10241 | 5.0208 | 703.5538 | 2.8992 |

Solutions of Exercise 5  
We have merged the two table in the exercise and fill them up with obtained result. From the table we can see that just like in exercise 4, again the time complexity is . Also, we can see that from the table, calculating the inverses cost approximately 3 times more expensive than directly solving the equation. Thus, we have learned that, if our purpose is just to solve a system, we should never calculate inverse of the coefficient matrix.

Solutions of Exercise 6

1. We would like to use the given code in the exercise 6 in order to find the first-row reduction step of the Hilbert matrix that has dimension 5. For this purpose, firstly, we shall define a new script named with “exer6.m”, that contains the exactly given code as follows  
   After defining such script, we have run it and obtained some newly defined variables such as U, L, Irow, Jcol etc. In order to check whether the code is really working or not, we have looked at the all entries of the first column in the second through last row of the matrix U.   
   , so the code is working correctly.

n=5;

Jcol=1;

A=hilb(5);

L=eye(n); % square n by n identity matrix

U=A;

for Irow=Jcol+1:n

% compute Irow multiplier and save in L(Irow,Jcol)

L(Irow,Jcol)=U(Irow,Jcol)/U(Jcol,Jcol);

% multiply row "Jcol" by L(Irow,Jcol) and subtract from row "Irow"

% This vector statement could be replaced with a loop

U(Irow,Jcol:n)=U(Irow,Jcol:n)-L(Irow,Jcol)\*U(Jcol,Jcol:n);

end

1. In order to see that, we really have L\*U = A, we have used the following code line and after running the code line we have obtained following output immediately as follows  
   Therefore, from the output we can see that, L\*U = A. Also, not directly checking the entries one by one, we can use norms in order to confirm ourself that the given equality really holds. For instance, in our example, we have used the following code that calculates the “Frobenius” norm to check whether the given two matrices are equal or not as follows  
   Thus, as we expected, we really have that L\*U = A, so the code works correctly.

>> A == L\*U

ans =

5×5 logical array

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

>> norm(L\*U-A,'fro')/norm(A,'fro')

ans =

0

1. We have defined the new script file as the exercise suggested us to do by adding the first script file only few line codes that takes loop of row reduction formula to each column of matrix.
2. After defining the model in order to check whether it is running correctly or not we shall apply Hilbert matrix with 5 dimension to the model and obtained the results as follows
3. Although, in the U matrix, the value of the entry of 5th row of 5th column is 0, when we expand the format from short to the long, we have seen that its actual value is as follows
4. We can see from above, our L matrix is lower triangular, and U matrix is upper triangular. Also, the diagonal entries of the matrix L are all zero. However, there is an inaccuracy in the model that comes from the fact that the algorithm does not use partial pivoting while applying the Gaussian elimination. Because of this inaccuracy, although it seems like that U is a triangular matrix in the output, in reality it is not actually an upper triangular matrix. Some of the entries of the matrix U are very close to the zero, such as U(3,2) but not 0.
5. Just like in the previous case, let us use the Frobenius norm to check whether the given two matrices are equal or not. For this purpose, we have used the following code line and obtain;  
   From the output we can see that the result is quite close to the 0 but not exactly the 0. This inaccuracy again comes from the non-pivoting. Apart from this inaccuracy, we have that the multiplication of the L and U gives us the original Hilbert matrix with 5 dimensions.

>> norm(L\*U-A,'fro')/norm(A,'fro')

ans =

3.040918255335998e-17

1. We have defined a randomly generated square matrix R with 100 dimensions, with using “R = rand(100,100);” code line. After defined such matrix, we have applied our “gauss\_lu.m” function to the matrix R. Since we do not want to check each 100 dimensions triangular matrices U’s and L’s entries one by one, we have used “istril(L)” and “istiu(U)” functions to check we really have the factorization. From the output, we have seen that, again we have a very close factorization as the norm is very close to the 0, but not exactly the L U factorization we have here. Just like in Hilbert Matrix case, there is a small inaccuracy that comes from the non-pivoting, if we had the partial pivoting, then there would not be such a problem.

Solutions of Exercise 7

1. In order to calculate the L U decomposition of the matrix A1, firstly we have defined A1 in to the workspace as a variable. Then, we have applied “gauss\_lu.m” function to evaluate the decomposition. However, we have obtained the following output as the L and U matrices;
2. The method failed for the given matrix A1 while finding the L U decomposition. The main reason for that while making the row reduction, at some steps the algorithm tries to divide large numbers with small number (or even tries to divide a number with 0) and also it tries to multiply 0 with Inf. This undefined and undetermined types of operation cause to obtain such term like “Inf” and “NaN”. We can see from the output that; first problem occurs while applying row reduction to 3rd column of 4th row of the matrix A. The algorithm tries to divide 4 with 0. That causes to obtain Inf. After this step, to write L matrices 3rd column of 4th row entry, the algorithm tries to multiply Inf with 0 that causes NaN. The problem occurs for few steps after this point, such as at 4th and 5th column of the matrix A.
3. The determinant of the matrix A1 is -8 and the condition number of the matrix A1 is 15.4003 with respect to the usual norm, which MatLab uses. Since the condition number is NOT that large and the determinant is non-zero, we have that A1 is well-conditioned.