A new Coding method in MATLAB used for solving a System of n Linear Equations by LU Decomposition

Digvijay Singh[1], Dr L. N. Das[2]

[1 2]*Department of Applied Mathematics,Delhi Technological University,Shahabad Daulatpur, Delhi,INDIA*

lndas@dce.ac.in

---***---

Abstract - *MATLAB coding which we have executed is not restricted to LU Decomposition only; rather it can be used for the matrix computation operations such as Cholesky method etc. In the existing MATLAB coding, to solve a system of n linear equations using LU Decomposition, a pivot matrix is used to reassign the largest element of each column to the principal diagonal in the coefficient matrix. Whereas in our MATLAB coding the pivot matrix is not required. In this paper we have executed a MATLAB coding for the solution of a system of n linear equations using LU Decomposition*

Key Words: LU decomposition

1.1 Introduction

MATLAB is a highly resourceful and useful tool used by a majority of the academic community to create models, simulations and solve many problems from the fields of engineering, science and mathematics. It is particularly known for its ease in handling arrays and matrices. A lot of in-built functions are provided to make computation easy.

In this paper, we have defined a MATLAB coding that is useful to solve a system of n linear equations with n variables. In the literature[1], we find that a matrix called pivot matrix is used to obtain the solution accordingly. In the pivot matrix multiplication, the main aim is to rearrange the columns elements of the coefficient matrix, so that the largest element of each column falls on the principal diagonal. This pivotization is more time taking. We suggest a new method to overcome these difficulties. In this paper, we have presented the brief definition and MATLAB coding of the LU Decomposition of the coefficient matrix of the linear equations, and using this decomposition have found the solution of the system of equations.

1.2 Required Concepts

2.1Rank -The rank of a matrix A is the size of the largest collection of linearly independent columns of

2.2Coefficient Matrix - The coefficient matrix refers to a matrix consisting of the coefficients of the variables in a set of linear equation.

2.3Augmented Matrix -An augmented matrix is a matrix obtained by appending the columns of coefficient matrix with the constant matrix, usually for the purpose of performing the same elementary row operations on each of the given matrices.

2.4Inverse of a Matrix -For a square matrix A, the inverse is written A^{-1}. When A is multiplied by A^{-1} the result is the identity matrix I. Non-square matrices do not have inverses. Not all square matrices have inverses.

2.5Zero Matrix -A zero matrix is the additive identity of the additive group of matrices.

2.6Identity Matrix -A square matrix in which all the elements of the principal diagonal are ones and all other elements are zeros. The effect of multiplying a given matrix by an identity matrix is to leave the given matrix unchanged.

2.7 Positive Definite Matrix - Asymmetricn xnrealmatrixA is said to be positive definite if z^TAz is positive for every non-zero column vectorz of n real numbers. Here z^T denotes the transpose of z.

2.8Matrix Multiplication -

Let $A_{m \times p}$ and $B_{p \times n}$ be two matrices whose elements are represented by,

$A = [a_{ij}]$ & $B = [b_{ij}]$ for i = 1,2,3,... m and j = 1,2,3,... n.

Then their matrix product is given by[2],

$C_{ij} = \qquad a_{ik} * b_{kj}$

2.9System of n Linear Equations –

A system of linear equations is a set or collection of equations that are dealt with all together at once.

A system of m equations in n variables is defined as,

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
\vdots \quad \vdots \quad \vdots &\quad \vdots \\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m
\end{aligned}$$

wherea_{ij}, b_i, are real numbers and x_j are variables for i = 1,2,…,m, j = 1,2,…,n. In matrix notation it is represented as,

AX = b

Three cases arise to the solution of this system of equations:

1. Unique solution
2. Infinite solutions
3. No solution

The uniqueness of the solution ofthissystem can be seen using matrix algebra[3] by comparing the rank of the coefficient matrix A and augmented matrix K as follows (where n is the number of variables/unknowns)

1. If rank(A) = rank(K) = n, there exists a unique solution.
2. If rank(A) = rank(K) < n, there exist infinitely many solutions.

There are various methods to solve this system of equations like substitution, cross-multiplication, matrix algebra methods etc. We concentrate here on LU Decomposition method.

1.3 LU Decomposition Method

In LU Decomposition[4], every square matrix A can be decomposed into a product of a lower triangular matrix L and an upper triangular matrix U.

A = LU

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}.$$

The sufficient condition for a matrix A to be decomposable is positive definiteness.

Now, for forming system of equations solvable for unique solution, one of the upper or lower triangular matrices should have 1's in the diagonal elements.

If upper triangular matrix, U has diagonal elements as 1, it is called Crout's Method.

If lower triangular matrix, L has diagonal elements as 1, it is called Doolittle's Method.

AX = B

Let A = LU

LUX = B

Let Z = UX

LZ = B

$Z = L^{-1}B$

Here, A is the coefficient matrix, B is the constant matrix, X is the variable matrix (whose elements are to be determined), L is the lower triangular matrix (whose elements are to be determined) and U is the upper triangular matrix (whose elements are to be determined).

2.1 MATLAB Functions

The following MATLAB functions[5] have been used in the below defined coding:

size(X)returns the row and column size of matrix X.

zeros(m)defines a square zero matrix of size m.

eye(m)defines an identity matrix of size m.

solve(f, x)solves an equation f for variable x.

2.2 MATLAB Codes

Below is provided the MATLAB R2014b (version 8.4.0)code written for Crout's Method.

```
function [ L, U, X ] = LU_Crout( A, b )

% Function to find solution of AX = b by Crout's
% LU Decomposition method.
% Input = A, b
% Output = L, U, X

 n = size(A);

flag = 1;

if( n(1) ~= n(2) )
fprintf('\nNot square matrix\n');
flag = 0;
return;
end

if( flag == 1 )
    K = [A b];
end

if( flag == 1 )
if( rank(A) ~= rank(K) )
fprintf('\nNo solution');
return;
elseif( rank(A) == rank(K) && rank(A) < n(1) )
fprintf('\nInfinite solutions')
return;
end
end
end


symstemp;

L = sym(zeros(n(1)));
U = sym(eye(n(1)));

T = sym(zeros(n(1)));

fori = 1:n(1)
 for j = 1:n(1)
if( i>= j )
L(i,j) = temp;
elseU(i,j) = temp;
end
for k = 1:n(1)
T(i,j) = T(i,j) + L(i,k)*U(k,j);
end
eq_ans = solve(T(i,j) == A(i,j), temp);
T(i,j) = eq_ans;
if( i>= j )
L(i,j) = T(i,j);
elseU(i,j) = T(i,j);
end
end
end

```

```
59 Z = L\b;   % Z = inv(L)*b

60

61 X = U\Z;   % X = inv(U)*Z

62

63end
```

2.3 Explanation of the codes

Line 1 defines a function LU_Crout that takes A (coefficient matrix) and b (constant column vector) as inputs, and gives decomposed lower and upper triangular matrices (L&U respectively), as well as X, solution to the system AX = b as output.

Lines 8-31 are measures taken to check whether A is a square matrix. If not, then LU Decomposition cannot be used. These measures also check if the system has a unique solution or not, by comparing rank of coefficient matrix, augmented matrix and the number of variables.

The main code where the LU Decomposition is done and the solution is obtained is contained in lines 34-63.

In lines 36 & 37, we initially set zero values to unknown elements of L&U, thus getting the zero and identity matrix respectively, both of sizen x n.

This is done because it is known thatL matrix is lower triangular matrix and U is upper triangular matrix with 1 on diagonals.

If we need to use Doolittle's method, we take L as identity matrix and U as zero matrix.

We declare a symbolic variable temp in line 34 to represent the unknown element of L or U in the current iteration.

We now perform matrix multiplication using the algorithm provided in 2.8, and store the obtained values in a temporary zero matrix, T (defined in line 39)

The entire algorithm relies on the property that as we perform matrix multiplication row-wise, in the product equation there is only one unknown, and values of all unknowns before it have been found out. The product equation is obtained by equating the current matrix multiplication vector with the corresponding value from the coefficient matrix, which can be solved using function solve(f, x) given in 2.11.

The condition in line 43-46 checks which variable is unknown in the current multiplication iteration. This unknown is replaced by temp in the product equation.

Using solve(f, x) where x = temp, we get value of unknown.

At each iterative step, the calculated value of unknown is replaced in the L or U matrix.

After the final iteration, the unknown elements of L and U are replaced, and they can be used to find the solution vector X.

In line 59, we find out $Z = L^{-1}B$ using the command Z = L\B. Similarly, we find out $X = U^{-1}Z$ using the command X = U\Z.

Thus, solution to the system of equations is obtained..

2.4. Advantages of Proposed method

The main advantage of the written coding is to overcome the problems existing in the given MATLAB coding. MATLAB cannot currently handle symbolic matrices; rather the coding is supported by symbolic variables. The coding we have stated is not restricted to LU Decomposition. It can be used for other matrix computation operations such as Cholesky Decomposition etc. In the existing MATLAB coding to solve a system of n linear equations using LU Decomposition, a pivot matrix is assigned to reassign the order of column elements of the coefficient matrix. In our coding the pivot matrix is not required.

Moreover in our coding it is not needed to specify the extrapolation of iterative formula for each method.

3. Examples

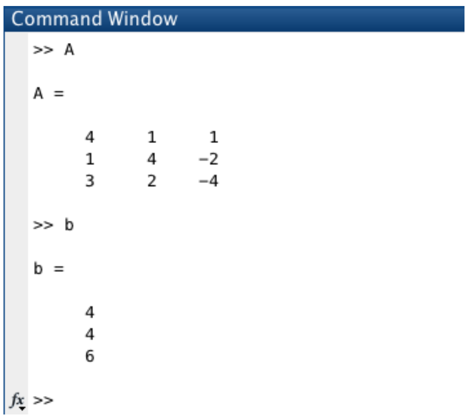
We verified the following examples in our executed MATLAB coding.

Consider the system of equations –

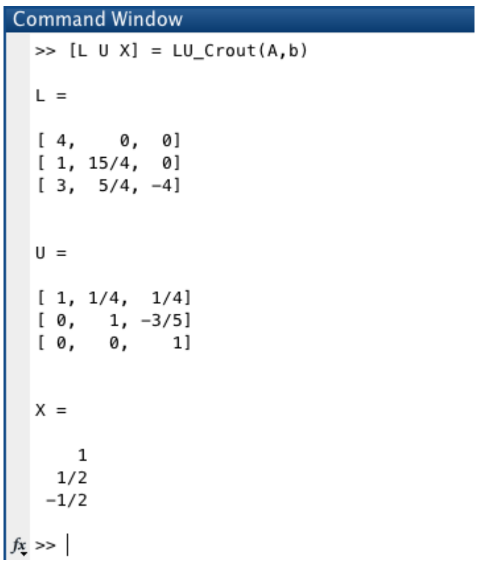
$$4x + y + z = 4$$

$$x + 4y - 2z = 4$$

$$3x + 2y - 4z = 6$$



```
Command Window
>> A

A =

     4     1     1
     1     4    -2
     3     2    -4

>> b

b =

     4
     4
     6

fx >>
```



```
Command Window
>> [L U X] = LU_Crout(A,b)

L =

[ 4,    0,  0]
[ 1, 15/4,  0]
[ 3,  5/4, -4]


U =

[ 1, 1/4,  1/4]
[ 0,   1, -3/5]
[ 0,   0,    1]


X =

    1
  1/2
 -1/2

fx >> |
```

Consider the system of equations –

$a + b - 2c + d + 3e - f = 4$

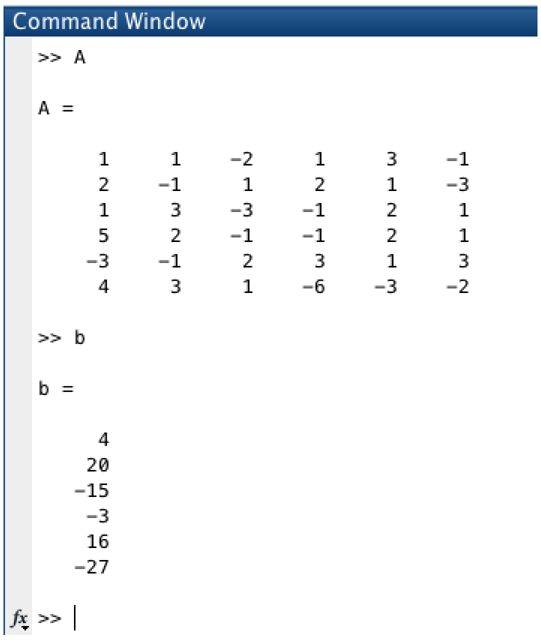
$2a - b + c + 2d + e - 3f = 20$

$a + 3b - 3c - d + 2e + f = -15$

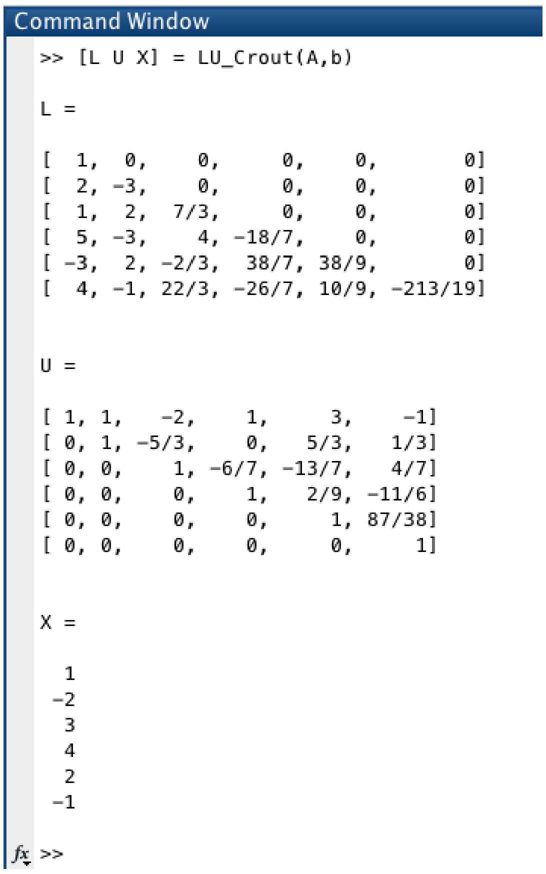
$5a + 2b - c - d + 2e + f = -3$

$-3a - b + 2c + 3d + e + 3f = 16$

$4a + 3b + c - 6d - 3e - 2f = -27$



```
Command Window
>> A

A =

     1     1    -2     1     3    -1
     2    -1     1     2     1    -3
     1     3    -3    -1     2     1
     5     2    -1    -1     2     1
    -3    -1     2     3     1     3
     4     3     1    -6    -3    -2

>> b

b =

     4
    20
   -15
    -3
    16
   -27

fx >> |
```



```
Command Window
>> [L U X] = LU_Crout(A,b)

L =

[  1,  0,    0,     0,    0,       0]
[  2, -3,    0,     0,    0,       0]
[  1,  2,  7/3,     0,    0,       0]
[  5, -3,    4, -18/7,    0,       0]
[ -3,  2, -2/3,  38/7, 38/9,       0]
[  4, -1, 22/3, -26/7, 10/9, -213/19]


U =

[ 1, 1,   -2,    1,     3,    -1]
[ 0, 1, -5/3,    0,   5/3,   1/3]
[ 0, 0,    1, -6/7, -13/7,   4/7]
[ 0, 0,    0,    1,   2/9, -11/6]
[ 0, 0,    0,    0,     1, 87/38]
[ 0, 0,    0,    0,     0,     1]


X =

  1
 -2
  3
  4
  2
 -1

fx >>
```

REFERENCES

[1] http://rosettacode.org/wiki/LU_decomposition - Creating a MATLAB function
[2] Erwin Kreyszig, Advanced Engineering Mathematics, 9th Edition – page 279
[3] Erwin Kreyszig, Advanced Engineering Mathematics, 9th Edition – page 302
[4] Erwin Kreyszig, Advanced Engineering Mathematics, 9th Edition – page 841
[5] RudraPratap, Getting Started with MATLAB, Oxford University Press, 2010