C Program for LU Factorization

June 13, 2020

For this **C program for LU factorization**, consider a general linear system AX = b, such that the given matrix [A] is factorized into the product of two upper and lower triangular matrices. The solution of linear simultaneous equations sought this way is called LU factorization method. This method is also known as the Triangular method or the LU Decomposition method.

So, the basic principle here is – "A square matrix [A] can be written as the product of a lower triangular matrix [L] and an upper triangular matrix [U], one of them being unit triangular, if all the principal minors of [A] are non-singular. Now, see how the matrix undergoes decomposition in the program for **LU Factorization in C** language.

[A] = [L][U]

where, L is the lower triangular matrix and U is the upper triangular matrix. The elements of these three matrices row-wise are:

[A] = {a11, a12, a13, a21, a22, a23, a31, a32, a33}
[L] = {1, 0, 0, l21, 1, 0, l31, l32, 1}
[U] = {u11, u12, u13, 0, u22, u23, 0, 0, u33}

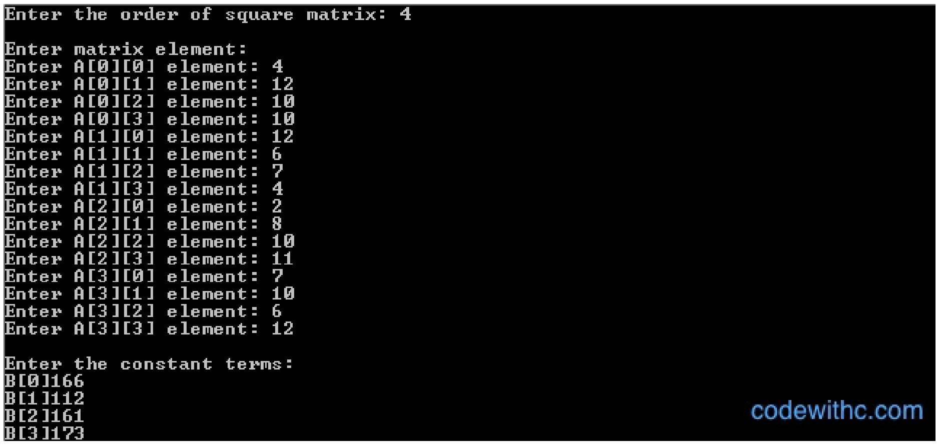
The representation obtained upon putting these elements in the 1st expression aforementioned is termed as LU factorization. It is simply a product of a lower and an upper triangular matrix decomposed from the parent matrix A. The diagonal non-zero entry of [L] is the ith pivot during factorization. The elements of matrix A can be obtained by multiplying [L] and [U].

This factorization method is preferred over the Gauss Elimination method in computers and programming languages like C. It facilitates obtaining products in double length. This C program below illustrates the application of LU Factorization method based on the things mentioned above.

Source Code for LU Factorization in C:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    float A[20][20]= {0},L[20][20]= {0}, U[20][20];
    float B[20]= {0}, X[20]= {0},Y[20]= {0};
    int i,j,k,n;
    printf("Enter the order of square matrix: ");
    scanf("%d",&n);
    printf("\nEnter matrix element:\n");
    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
        {
            printf("Enter A[%d][%d] element: ", i,j);
            scanf("%f",&A[i][j]);
        }
    }
    printf("\nEnter the constant terms: \n");
    for(i=0; i<n; i++)
    {
        printf("B[%d]",i);
        scanf("%f",&B[i]);
    }
    for(j=0; j<n; j++)
    {
        for(i=0; i<n; i++)
        {
            if(i<=j)
            {
                U[i][j]=A[i][j];
                for(k=0; k<i-1; k++)
                    U[i][j]-=L[i][k]*U[k][j];
                if(i==j)
                    L[i][j]=1;
                else
                    L[i][j]=0;
            }
            else
            {
                L[i][j]=A[i][j];
                for(k=0; k<=j-1; k++)
                    L[i][j]-=L[i][k]*U[k][j];
                L[i][j]/=U[j][j];
                U[i][j]=0;
            }
        }
    }
    printf("[L]: \n");
    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
            printf("%9.3f",L[i][j]);
        printf("\n");
    }
    printf("\n\n[U]: \n");
    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
            printf("%9.3f",U[i][j]);
        printf("\n");
    }
    for(i=0; i<n; i++)
    {
        Y[i]=B[i];
        for(j=0; j<i; j++)
        {
            Y[i]-=L[i][j]*Y[j];
        }
    }
    printf("\n\n[Y]: \n");
    for(i=0; i<n; i++)
    {
        printf("%9.3f",Y[i]);
    }
    for(i=n-1; i>=0; i--)
    {
        X[i]= Y[i];
        for(j=i+1; j<n; j++)
        {
            X[i]-=U[i][j]*X[j];
        }
        X[i]/=U[i][i];
    }
    printf("\n\n[X]: \n");
    for(i=0; i<n; i++)
    {
        printf("%9.3f",X[i]);
    }
    getch();
}
```

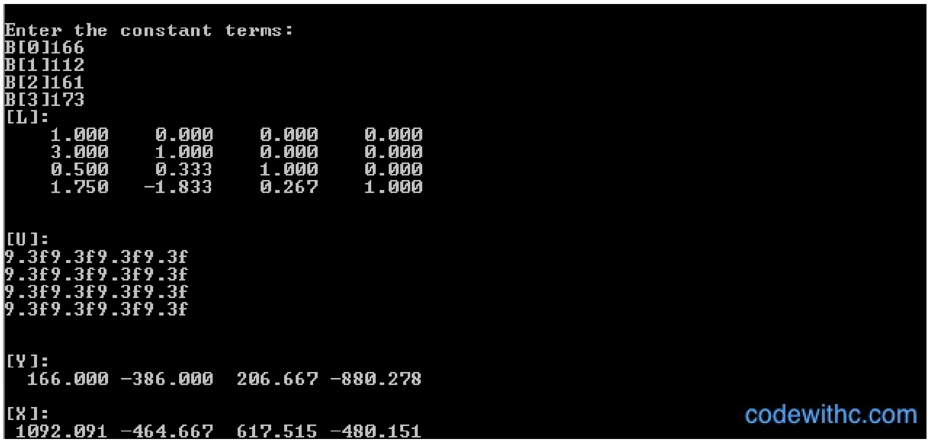
Input/Output:



```
Enter the order of square matrix: 4

Enter matrix element:
Enter A[0][0] element: 4
Enter A[0][1] element: 12
Enter A[0][2] element: 10
Enter A[0][3] element: 10
Enter A[1][0] element: 12
Enter A[1][1] element: 6
Enter A[1][2] element: 7
Enter A[1][3] element: 4
Enter A[2][0] element: 2
Enter A[2][1] element: 8
Enter A[2][2] element: 10
Enter A[2][3] element: 11
Enter A[3][0] element: 7
Enter A[3][1] element: 10
Enter A[3][2] element: 6
Enter A[3][3] element: 12

Enter the constant terms:
B[0]166
B[1]112
B[2]161
B[3]173
```



```
Enter the constant terms:
B[0]166
B[1]112
B[2]161
B[3]173
[L]:
    1.000    0.000    0.000    0.000
    3.000    1.000    0.000    0.000
    0.500    0.333    1.000    0.000
    1.750   -1.833    0.267    1.000


[U]:
9.3f9.3f9.3f9.3f
9.3f9.3f9.3f9.3f
9.3f9.3f9.3f9.3f
9.3f9.3f9.3f9.3f


[Y]:
  166.000 -386.000  206.667 -880.278

[X]:
 1092.091 -464.667  617.515 -480.151
```

Also see,
LU Decomposition Algorithm/Flowchart
Numerical Methods Tutorial Compilation

If you have any questions related to this post – the LU Factorization (LU Decomposition) method or its C source code presented above, do mention them in the comments section.