

# LU Factorization C code

Paht Juangphanich

## I. Introduction

The purpose of this report is to document the procedure of programming LU decomposition in C. I've used a Mersenne Twister 64 bit program by coded by Takuji Nishimura and Makoto Matsumoto to generate the matrix and my own code for LU factorization.

LU Factorization is used to solve a general matrix problem  $Ax=B$ . The goal of LU is to break down A into two components  $A = L*U$ . Where L is a lower triangular matrix with diagonals set to 1 usually and U is upper triangular matrix. Gaussian elimination is used to create the L and then U can be calculated.

## II. Theory

If you we assume that we have a matrix  $Ax = b$  and we have some matrix  $A = LU$  then  $(LU)x = B$

$$L(Ux) = b \quad (1)$$

$$Ly = b \quad (2)$$

$$Ux = y \quad (3)$$

$$(4)$$

Equation 2 can be written as,

$$\begin{bmatrix} l_{1,1} & 0 & 0 & \dots & 0 \\ l_{2,1} & l_{2,2} & 0 & \dots & 0 \\ l_{3,1} & l_{3,2} & l_{3,3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{m,1} & l_{m,2} & l_{m,3} & \dots & l_{m,n} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

$$\text{row-1: } l_{1,1}y_1 = b_1$$

$$\text{row-2: } l_{2,1}y_1 + l_{2,2}y_2 = b_2$$

$$\text{row-3: } l_{3,1}y_1 + l_{3,2}y_2 + l_{3,3}y_3 = b_3$$

$$\text{row-m: } l_{m,1}y_1 + l_{m,2}y_2 + l_{m,3}y_3 + \dots + l_{m,n}y_n = b_n$$

If we let  $l_{i,i}$  all the way though  $l_{i,i}$  for  $i = 1$  to  $m$  be equal to 1 then it is possible to find a value for  $y_j$  for  $j = 1$  to  $n$ .

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \dots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & \dots & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \dots & a_{m,n} \end{bmatrix}$$

The idea is create each element of the L matrix then proceed with gaussian elimination on matrix A. The upper triangular of matrix A will be U.

$$\begin{aligned}
 L_{2,1} &= a_{2,1}/a_{1,1} \\
 a_{2,2} &= a_{2,2} - L_{2,1}a_{1,2} \\
 a_{2,3} &= a_{2,3} - L_{2,1}a_{1,3} \\
 a_{2,n} &= a_{2,n} - L_{2,1}a_{1,n} \\
 L_{3,1} &= a_{3,1}/a_{2,2} \\
 &\text{etc..}
 \end{aligned}$$

### III. Numerical Approach

This code below can be used to compute the LU factorization for NxN matrix.

Listing 1. Descriptive Caption Text

```

1 void LU(double **a, int n, double **L, double **U)
  {
3     // No row exchanges
    // Square matrix
5     int k,i,j;

7     #pragma omp parallel shared(a,n,L,U) private(k,i,j)
    {
9         #pragma omp for schedule(static,10)
        for (k=0; k<n; k++) {
11             L[k][k]=1;
                for (i=k+1; i<n; i++) {
13                 L[i][k]=a[i][k]/a[k][k];
                    // a[i][k]=L[i][k];
15                 for (j=k+1; j<n; j++) {
                            a[i][j]=a[i][j]-L[i][k]*a[k][j];
17                 }
                    #pragma omp flush(a)
19             }
                #pragma omp nowait
21             for (j=k; j<n; j++) {
                        U[k][j]=a[k][j];
23             }
25         }
    }
}

```

This code will not work if the dimensions are not the same. If dimension are different say nxm. L would have to be nxn if n is less than m and U would have to be nxm. This is the only way it would work.