# LU Decomposition

By
Dilip Puri
Govind Meena
Hemant Kumar

October 24, 2016

# Problem

We have a matrix A, now we want to decompose matrix A into two matrices L(Lower Triangular Matrix) and U(Upper Triangular Matrix) such that

$$\mathbf{A = L * U}$$

where

$$L = \begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ b & c & 1 \end{bmatrix}, U = \begin{bmatrix} d & e & f \\ 0 & g & h \\ 0 & 0 & i \end{bmatrix}.$$

# Motivation

`So the first question come to mind that why we are doing this?`

so straight forward answer would be that it will simplify things.
How?
Most of the time in mathematics modeling we came up with system of linear equations in the form of

$$\mathbf{Ax = b}$$

so finding $A^{-1}$ is quite difficult so we will use LU decomposition

## How?

# Example

Lets we have system of eqations

$$
\begin{aligned}
[A]\{x\} &= \{b\} \\
[L][U]\{x\} &= \{b\} \\
(\therefore [A] &= [L][U]) \\
\{y\} &= [U]\{x\} \\
[L]\{y\} &= \{b\} \\
\because [] = matrix, \{\} &= vector
\end{aligned}
$$

This will make our system so simple to solve...

# Example

$$A = \begin{bmatrix} 1 & 2 & 4 \\ 3 & 8 & 14 \\ 2 & 6 & 13 \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, b = \begin{bmatrix} 3 \\ 13 \\ 4 \end{bmatrix}.$$

$$A = LU \Rightarrow \begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ b & c & 1 \end{bmatrix} * \begin{bmatrix} d & e & f \\ 0 & g & h \\ 0 & 0 & i \end{bmatrix} = \begin{bmatrix} d & e & f \\ ad & ae+g & af+h \\ bd & be+cg & bf+ch+i \end{bmatrix}.$$

$$\Rightarrow \begin{bmatrix} d & e & f \\ ad & ae+g & af+h \\ bd & be+cg & bf+ch+i \end{bmatrix} = \begin{bmatrix} 1 & 2 & 4 \\ 3 & 8 & 14 \\ 2 & 6 & 13 \end{bmatrix}$$

Now compare the values and get the values of elements of L and U.

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 2 & 1 & 1 \end{bmatrix}, U = \begin{bmatrix} 1 & 2 & 4 \\ 0 & 2 & 2 \\ 0 & 0 & 3 \end{bmatrix}.$$

The next step is to solve [L]{y}={b} for the vector {y} that we consider

$$Ly = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 2 & 1 & 1 \end{bmatrix} * \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 13 \\ 4 \end{bmatrix} = b$$

which can be solved by forward substitution {y} = $[3\ 4\ \text{-}6]^T$ now that we have found y we finish the procedure by solving

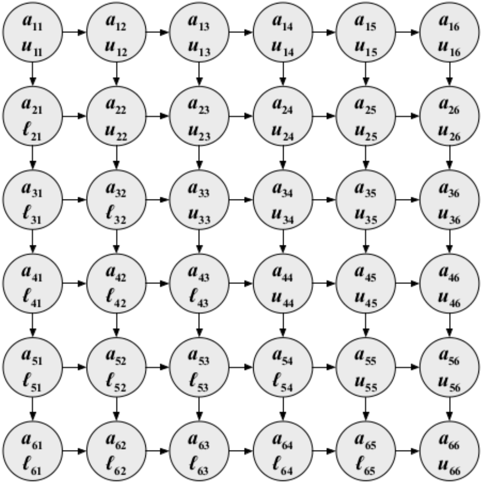
$$Ux = \begin{bmatrix} 1 & 2 & 4 \\ 0 & 2 & 2 \\ 0 & 0 & 3 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ -6 \end{bmatrix} = y$$

by using backward substitution we will get {x}.

# Algorithm

**for** $k = 1$ **to** $\min(i, j) - 1$ — { vert bcast }
  recv broadcast of $a_{kj}$ from task $(k, j)$ — { vert bcast }
  recv broadcast of $\ell_{ik}$ from task $(i, k)$ — { horiz bcast }
  $a_{ij} = a_{ij} - \ell_{ik}\, a_{kj}$ — { update entry
**end**
**if** $i \leq j$ **then**
  broadcast $a_{ij}$ to tasks $(k, j),\ \ k = i+1, \ldots, n$ — { vert bcast }
**else**
  recv broadcast of $a_{jj}$ from task $(j, j)$ — { vert bcast }
  $\ell_{ij} = a_{ij}/a_{jj}$ — { multiplier }
  broadcast $\ell_{ij}$ to tasks $(i, k),\ \ k = j+1, \ldots, n$ — { horiz bcast }
**end**

# Task Generation and Dependency Graph



Since the LU decomposition is completely done using elimination and substation, there are 3 for loops involved and also the cells of a particular row (for U) or particular column (for L) can be computed in parallel so we used LU decomposition block as a suitable area to implement parallelism. Each code has sufficient comments inside to describe the methods and important statements.

# SpeedUp

| Data | | | |
|---|---|---|---|
| Size | Sequential | OMP | Pthread |
| 10 | 0 | 0.000971 | 0 |
| 25 | 0 | 0.001993 | 0.000001 |
| 50 | 0 | 0.00483 | 0.02 |
| 100 | 0.02 | 0.01813 | 0.01 |
| 200 | 0.08 | 0.091132 | 0.03 |
| 300 | 0.15 | 0.252912 | 0.08 |
| 400 | 0.33 | 0.272793 | 0.2 |
| 500 | 0.77 | 0.658797 | 0.25 |
| 1000 | 5.8 | 2.806621 | 1.03 |

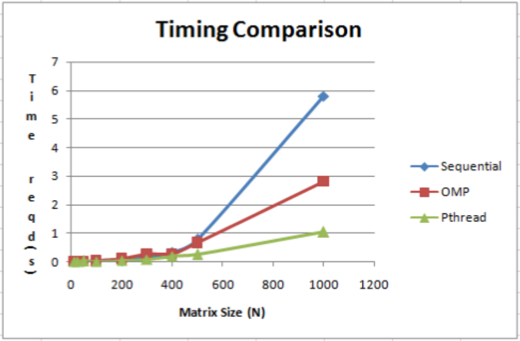



Figure : Output

# OpenMP

| OMP Speedup Data | | | | | |
| --- | --- | --- | --- | --- | --- |
| Size | Sequential | OMP | Speedup | Processors | Effieciency |
| 10 | 0 | 0.000971 | 0 | 2 | 0 |
| 25 | 0 | 0.001993 | 0 | 2 | 0 |
| 50 | 0 | 0.00483 | 0 | 2 | 0 |
| 100 | 0.02 | 0.01813 | 1.103144 | 2 | 0.55157198 |
| 200 | 0.08 | 0.091132 | 0.877848 | 2 | 0.43892376 |
| 300 | 0.15 | 0.252912 | 0.593092 | 8 | 0.07413646 |
| 400 | 0.33 | 0.272793 | 1.209708 | 2 | 0.60485423 |
| 500 | 0.77 | 0.658797 | 1.168797 | 4 | 0.29219927 |
| 1000 | 5.8 | 2.806621 | **2.066542** | 4 | 0.51663548 |

Figure : Output

# Pthread

| PThread Speedup Data | | | | | |
|---|---|---|---|---|---|
| Size | Sequential | Pthread | Speedup | Threads | Effieciency |
| 10 | 0 | 0 | 1 | 3 | 0.33333333 |
| 25 | 0 | 0 | 1 | 5 | 0.2 |
| 50 | 0 | 0 | 1 | 5 | 0.2 |
| 100 | 0.02 | 0.01 | 2 | 4 | 0.5 |
| 200 | 0.08 | 0.03 | 2.666667 | 2 | 1.33333333 |
| 300 | 0.15 | 0.08 | 1.875 | 4 | 0.46875 |
| 400 | 0.33 | 0.2 | 1.65 | 4 | 0.4125 |
| 500 | 0.77 | 0.25 | 3.08 | 2 | 1.54 |
| 1000 | 5.8 | 1.03 | 5.631068 | 4 | 1.40776699 |

Figure : Output

# Q & A

Thank You!