Author: Dr. Shi Guo Email: guoshi1984@hotmail.com

1 LU Decomposition

a. Motivation

When we solve a linear system equation $Ax = b$, image we are able to decompose A into L and U and $A = LU$, where L is a lower triangular matrix U is an upper triangular matrix. Then equation $Ax = b$ becomes $LUx = b$. Let $Ux = y$, as both L and U are triangular matrix, we can solve for y in $Ly = b$ first, then solve for x in $Ux = y$ second.

b. Formula

Decomposition form, $LU = A$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 & 0 \\ l_{41} & l_{42} & l_{43} & 1 & 0 \\ l_{51} & l_{52} & l_{53} & l_{54} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ 0 & u_{22} & u_{23} & u_{24} & u_{25} \\ 0 & 0 & u_{33} & u_{34} & u_{35} \\ 0 & 0 & 0 & u_{44} & u_{45} \\ 0 & 0 & 0 & 0 & u_{55} \end{pmatrix}$$

$$= \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{pmatrix}$$

we run the calculation starting from the 1st column to the last column, and for each column we update from the top row to the bottom row. For each element a_{ij}, we write it as a product of l's row and u's column. We start from a_{11}

$$\begin{aligned} a_{11} = u_{11} &\Rightarrow u_{11} \\ a_{21} = l_{21}u_{11} &\Rightarrow l_{21} \\ a_{31} = l_{31}u_{11} &\Rightarrow l_{31} \\ a_{41} = l_{41}u_{11} &\Rightarrow l_{41} \\ a_{51} = l_{51}u_{11} &\Rightarrow l_{51} \\ a_{12} = u_{12} &\Rightarrow u_{12} \\ a_{22} = l_{21}u_{12} + u_{22} &\Rightarrow u_{12} \\ a_{32} = l_{31}u_{12} + l_{32}u_{22} &\Rightarrow u_{22} \end{aligned}$$

1) for $i < j$, which are the upper triangular part, namely U part(except diagonal), for example, for i=3, j=2, the matrix multiplication form is the following

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ \mathbf{l_{21}} & 1 & 0 & 0 & 0 \\ . & . & 1 & 0 & 0 \\ . & . & . & 1 & 0 \\ . & . & . & . & 1 \end{pmatrix} \begin{pmatrix} . & . & \mathbf{u_{13}} & . & . \\ 0 & . & \mathbf{u_{23}} & . & . \\ 0 & 0 & . & . & . \\ 0 & 0 & 0 & . & . \\ 0 & 0 & 0 & 0 & . \end{pmatrix}$$

Generally,

$\sum_{k=1}^{min(i,j)-1} l_{ik}u_{kj} + u_{ij} = a_{ij}(1)$

We use the above equation to calculate u_{ij}
2) for $i = j$, which are the upper triangular part, namely U part(except diagonal), for example, for i=3, j=2, the matrix multiplication form is the following

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ . & 1 & 0 & 0 & 0 \\ \mathbf{l_{31}} & \mathbf{l_{32}} & 1 & 0 & 0 \\ . & . & . & 1 & 0 \\ . & . & . & . & 1 \end{pmatrix} \begin{pmatrix} . & . & \mathbf{u_{13}} & . & . \\ 0 & . & \mathbf{u_{23}} & . & . \\ 0 & 0 & \mathbf{u_{23}} & . & . \\ 0 & 0 & 0 & . & . \\ 0 & 0 & 0 & 0 & . \end{pmatrix}$$

Generally
$\sum_{k=1}^{j-1} l_{jk}u_{kj} + u_{jj} = a_{jj}(2)$
We use the above equation to calculate u_{jj}

$$u_{jj} = a_{jj} - \sum_{k=1}^{j-1} l_{jk}u_{kj} \tag{3}$$

3) for $i > j$, which are the lower triangular part, namely L part(except diagonal), for example, for i=3, j=2, the matrix multiplication form is the following

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ . & 1 & 0 & 0 & 0 \\ \mathbf{l_{31}} & \mathbf{l_{32}} & 1 & 0 & 0 \\ . & . & . & 1 & 0 \\ . & . & . & . & 1 \end{pmatrix} \begin{pmatrix} . & . & \mathbf{u_{13}} & . & . \\ 0 & . & \mathbf{u_{23}} & . & . \\ 0 & 0 & . & . & . \\ 0 & 0 & 0 & . & . \\ 0 & 0 & 0 & 0 & . \end{pmatrix}$$

Generally,
$\sum_{k=1}^{j-1} l_{ik}u_{kj} + l_{ij}u_{jj} = a_{ij}(4)$

$$l_{ij}u_{jj} = a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj} \tag{5}$$

$$l_{ij} = \frac{1}{u_{jj}}(a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}) \tag{6}$$

We use the above equation to calculate l_{ij}

c. Inplace Update
Why we can do in place update?
In the upate formula, each a_{ij} is used only once, and then we get either l_{ij} or u_{ij}, so we can store the l_{ij} or $u_{ij} in the exact same place where a_{ij}$ is stored. This is in-place update. For example, for a certain step i=4, j=3, our in-place updated matrix A looks like

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} & a_{14} & a_{15} \\ l_{21} & u_{22} & u_{23} & a_{24} & a_{25} \\ l_{31} & l_{32} & u_{33} & a_{34} & a_{35} \\ l_{41} & l_{42} & l_{43} & a_{44} & a_{45} \\ l_{51} & l_{52} & a_{53} & a_{54} & a_{55} \end{pmatrix}$$

d. Partial Pivoting
For each l_{ij}, we need to divided by u_{jj}, from equation 3

$$u_{jj} = a_{jj} - \sum_{k=1}^{j-1} l_{jk}u_{kj} \tag{7}$$

we never like dividing a really small number. So the trick is , for all rows below, we calculate the divider as if they were in the current row, and the exciting thing is we do this by the same formula because

$$l_{ij}u_{jj} = a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj} \tag{8}$$

The right hand side of the above equation is the same as the right hand side of equation 3. Let call $a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}$ the divider, we calculate all the dividers for i¿j in the j's column, choose the largest divider, then interchange the rows, divide all the rest rows by the largest divider.

e. Scaled Partial Pivoting

Notice the fact that if we multiply the row of the matrix by a very large number. The solution to the linear system equation never change. So in order to compare the divider's value $u_j j$, we can not compare the true value, we rather need to compare the scaled value, which is the value divided by the largest element in the row.

f. Implementation Details

When we using scaled partial pivoting, we basically change the order of rows. So the decomposed matrix is not the original matrix A, rather than a row permutation matrix of A. When we solve the equation $Ax = b$, since A has been change after LU decomposition, we need b to be changed in the same way of row permutation as A so that A and b are both consistent. This requires us to keep track of the order of the rows of matrix A. we need store the new row index into an array. We do not need to know which rows we exchange at every step, and knowing the order is enough. However, there is one more thing we need to take care of. A useful application of LU decomposition is to evaluate the determinant. When we interchange two rows of the matrix, the determinant gets a minus sign. So we need to store whether we exchange the rows even number of times or odd number of times, we call this parity.

g. Java Implementation

The java code is on github repo
https://github.com/guoshi1984/guoshi1984.github.io/math/java