Scribe for LAB - Session

① Explaination of Zero pivotting, Partial Pivotting and pivotting.

Zero pivoting

While performing zero pivotting, we exchange a_{ij} by a_{kj} where $k > i$, in order to make a_{ij} a pivot. For example matrix is.

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 3 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Thus for LU factorisation we need to pivot a_{22} by a_{32}.

Partial Pivoting is performed to make pivot maximum of all the numbers below it.

Full Pivoting

While full pivoting, numbers can be interchanged in rows as well as column to magnify pivot.

Example of Partial Pivoting

$$\begin{bmatrix} 2 & 0 & 1 \\ 9 & 4 & 0 \\ 3 & 6 & 0 \end{bmatrix} \longleftrightarrow \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 9 & 4 & 0 \\ 2 & 0 & 1 \\ 3 & 6 & 0 \end{bmatrix}$$

Example of Full Pivoting

$$\begin{bmatrix} 2 & 11 & 4 \\ 3 & 2 & 1 \\ 8 & 9 & 2 \end{bmatrix} \longleftrightarrow \begin{bmatrix} 11 & 2 & 4 \\ 2 & 3 & 1 \\ 9 & 8 & 2 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Why is partial pivoting done?

Example 1 :- To reduce errors.

Partial pivoting reduces the error.

For eg:- If we are working on the matrix.

$$\begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix}$$

If we do LU factorisation of such matrix without pivoting we will get

$$L = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 10^{-20} & 1 \\ 0 & 1-10^{20} \end{bmatrix}$$

as $\cancel{1-10^{20} \simeq -10^{20}}$

$$1-10^{20} \simeq -10^{20}$$

$$U = \begin{bmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{bmatrix}$$

On multiplying L and U, we get,

$$\begin{bmatrix} 10^{-20} & 1 \\ 1 & 0 \end{bmatrix}$$ which is a significant error

While if we use pivoting.

$$\begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 10^{-20} & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 10^{-20} & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 0 & 1-10^{-20} \end{bmatrix}$$

$$1-10^{-20} \simeq 1$$

$$= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 10^{-20} & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 10^{-20} & 1+10^{-20} \end{bmatrix}$$

which is not a very significant er

Secondly :-

If we are working on a matrix like

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix}$$

On having its LU factorisation. We will get

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 4 \\ 0 & 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 0 & 16 \end{bmatrix}$$

Thus matrix got enlarged in the order of 2^n which is a very large number to store for a high value of n.

But by using partial pivoting

we can't be prevented from such situati

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 \end{bmatrix}$$

But by using full pivoting, we can stop matrix from growing of order (2^n) which is not a easy job.

Thus we prefer to use partial pivoting in major cases.

For finding LU factorisation in R.

We need to do following steps.

Note:- R uses partial pivoting

① You need to install package ('Matrix')

② > library ('Matrix')

③ > expand (lu(A))

During partial pivoting, each time we need to calculate highest number in the column, which takes n ~~iteration~~ iteration.

Thus order is of $n^2 \{n+(n-1) \ldots\}$ which is not much as solving equation is $O(n^3)$

During partial pivoting, we get

$$L_3 P_3 L_2 P_2 L_1 P_1 A = U$$

But it can also be written a

$$L_3' L_2' L_1' P_3 P_2 P_1 A = U$$

where $L_3' = L_3 \quad L_2' = P_3 L_2 P_3^{-1}$

To solve this, we can use:-

Algorithm.

As we have got the permutation matrix

$$P = P_3 P_2 P_1$$

We only need to find L and U can be found by Gaussian elimin and by using

$$L = \begin{bmatrix} 1 & & & \\ l_{11} & 1 & & \\ l_{12} & l_{13} & 1 & \\ - & - & - & 1 \\ - & - & - & - & 1 \end{bmatrix}$$

We can solve

$$PA = LU$$

to get L !

Or can be simply said :-

1. Permute the rows according to P.
2. Apply Gaussian elimination without pivoting to PA.

Complete pivoting

During complete pivoting, we find the largest number in $(r \times r)$ matrix to be solved each times

Thus need r^2 iterations

$$O = \sum_{i=1}^{n} n^2 = O(n^3)$$

Due to high complexity order, we don't use it much.

In matrix form, complete pivoting precedes each elimination step with a permutation P_k of the rows applied on left and also a permutation Q_k of the columns applied on the right.

$$L_{m-1}P_{m-1}\cdots L_2P_2L_1P_1Q_1Q_2\cdots Q_{m-1}=U$$

Once again

$$(L'_{m-1}\cdots L'_1)(P_{m-1}\cdots P_2P_1)A(Q_1Q_2\cdots Q_{m-1})=U$$

$$L=(L'_{m-1}\cdots L'_1)^{-1}$$

$$P=P_{m-1}\cdots P_1$$

$$Q=Q_1\cdots Q_{m-1}$$

$$PAQ=LU$$

Thus in order of solve $Ax=b$ we have

$$PAQQ^{-1}x=Pb$$

$$LU(Q^{-1}x)=Pb$$

where Q^{-1} is permutation of $\begin{pmatrix}x_1\\ \vdots\\ x_m\end{pmatrix}$

~~If A is given matrix.~~

~~It provides you the matrix P, L, Q~~

For eg:-

If given matrix is

$$\begin{bmatrix} 2 & 4 & 5 & 6 \\ 3 & 7 & 9 & 1 \\ 1 & 8 & 2 & 4 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} & & \\ & I & \\ & & \end{bmatrix} \begin{bmatrix} 5 & 4 & 2 & 6 \\ 9 & 7 & 3 & 1 \\ 2 & 8 & 1 & 4 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 9 & 7 & 3 & 1 \\ 5 & 4 & 2 & 6 \\ 2 & 8 & 1 & 4 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(P, AQ_1)

While storing values, computer doesn't it in exact form.

For eg. as $\frac{n^2}{2}$ numbers of U and $\frac{n^2}{2}$ numbers of L are 0 and only n many numbers of P are rest 0. It only stores non zero values

Thus expand command is used to get real matrices.

Singular - Value Decomposition

Decomposition of any matrix A in the form of $A = \cancel{UV}$

$$U \Sigma V^T$$

where $U =$ orthogonal matrix of order $m \times m$

$\Sigma =$ diagonal matrix (order $m \times n$)

$V^T =$ orthogonal matrix (order $n \times n$)

$$V^T = V^{-1}$$

As V is orthogonal matrix

$\Rightarrow \quad UU^T = I_m \quad \Rightarrow \quad U^T = U^{-1}$

$VV^T = I_n$

$r =$ rank of matrix A.

First r rows of V^T are basis (orthogonal) of row space of A

First r columns of U are basis (orthogonal) of column space of B.

Thus, U and V^T are basically transformation from one basis to other basis.

First r diagonal elements of Σ are non-zero

$$\begin{bmatrix} | & | & | & & | \\ v_1 & v_2 & v_3 & \cdots & v_n \\ | & | & | & & | \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$x = \sum_{i=1}^{n} b_i v_i$$

Thus b is representation of x with respect to orthogonal basis of F^n (row space)

$b = \Sigma V^{-1}(x)$ leads to magnification of b in different directions. $\Sigma b = c$

$U \Sigma V^{-1}(x)$ converts c back to column space of A (using orthogonal basis of F^m)

Generation of random matrices using R

We are trying to guess relation between λ_i's with respect to matrix A.

Should it grow or diminish for high order matrices?

To analyse it, we can find $\frac{\max(\Sigma)}{\max(A)} = \rho$

and plot mean(ρ) and standard deviation (ρ) against n

where $n \times n$ is order of matrix

$$A = \begin{bmatrix} u_1 & u_2 & u_3 & \cdots & u_m \\ \vdots & \vdots & \vdots & & \vdots \end{bmatrix} \begin{bmatrix} \lambda_1 & & & & & \\ & \lambda_2 & & & & \\ & & \lambda_3 & & & \\ & & & \ddots & & \\ & & & & \lambda_r & \\ & & & & & 0 \\ & & & & & & 0 \end{bmatrix} \begin{bmatrix} - & v_1 & - \\ - & v_2 & - \\ - & v_3 & - \\ & \vdots & \\ - & v_n & - \end{bmatrix}$$

$$U \qquad \Sigma \qquad V^T$$

$$\lambda_1 \geq \lambda_2 > \lambda_3 > \ldots \lambda_r$$

u_i's are m orthogonal vectors $\in F_m$

v_i's are n orthogonal vectors $\in F_n$

Note :- We can check whether U and V are orthogonal or not by computing (UU^T) and (VV^T) in R.

Syntax :→

```
> U %*% t(U)
> V %*% t(V)
```

where $t(U)$ = transpose of matrix U.

Explanation of :-

$$U \Sigma V^T (x) = U \Sigma V^{-1} (x)$$

$V^{-1}(x)$ can be thought as representation of x in form of orthogonal basis of A (row space)

$$V^{-1}(x) = b$$

$$\Rightarrow \quad x = Vb$$

R- Syntax

```
> library(Matrix)
> row = numeric(100).
>exprow= numeric (50)
>sdrow = numeric (50)
> c= seq(10, 500, 50)
> for (n in c)
+ { for (i in 1:100)
+ { A = matrix (rnorm(n*n), n)
+ p= svd(A)$d
+ a = max(A)
+ d = max(p)
+ row[i] = a/d
+ }
+ exprow[n/10] = mean(raw)
+ sdrow [n/10] = sd(row)
+ }
>plot (c, expraw)
>plot (c, sdrow)
```

Plots are attached with the scribe.

Blow up factor

$$\rho = \text{Blow up factor} = \frac{Max|U|}{Max|A|}$$

where $A = LU$

In most of the cases, L does not blow up. So we need to have an upper bound on Max|U| and estimate it using experiments

Random Matrices Generation and Calculating (ρ)

```
> library(Matrix)
> row = numeric(100)
> c = seq(10,1000,10)
> for (n in c)
+ {
+ A = matrix(rnorm(n*n),n)
+ d = expand(lu(A))
+ a = max((A))
+ u = max((d$U))
+ row[n/10] = u/a
+ }
> plot(c, row)
```