

**Numerical Methods in Civil Engineering**  
**Prof. Arghya Deb**  
**Department of Civil Engineering**  
**Indian Institute of Technology, Kharagpur**


**Lecture - 6**  
**Linear Systems: Error Bounds**

A series in numerical methods in Civil Engineering, we are going to talk about Error Bounds for linear systems.

(Refer Slide Time: 00:24)

**Choleski Decomposition**

- A particularly useful compact form for positive definite matrices is given by Choleski decomposition.
- It is based on the following theorem which states: "For a symmetric positive definite matrix there is a unique upper triangular matrix with positive diagonal elements such that  $A=R^T R$ "
- From the LU theorem, we have  $A=LU$  where:  
$$u_{11} = a_{11} > 0$$
$$u_{22} = a_{22}^{(2)} = a_{22} - \frac{a_{12} a_{21}}{a_{11}} = \frac{a_{11} a_{22} - a_{12} a_{21}}{a_{11}} = \frac{\det A_2}{\det A_1}$$



However before doing so, we want to talk about one other method for solving linear systems in a chief and efficient manner. A method which is particularly suited for positive definite matrices, and as we know in civil engineering applications, we encounter a wide number of positive definite symmetric matrices, in a numerous applications. Particularly useful compact form for positive definite matrices is given by Choleski decomposition.

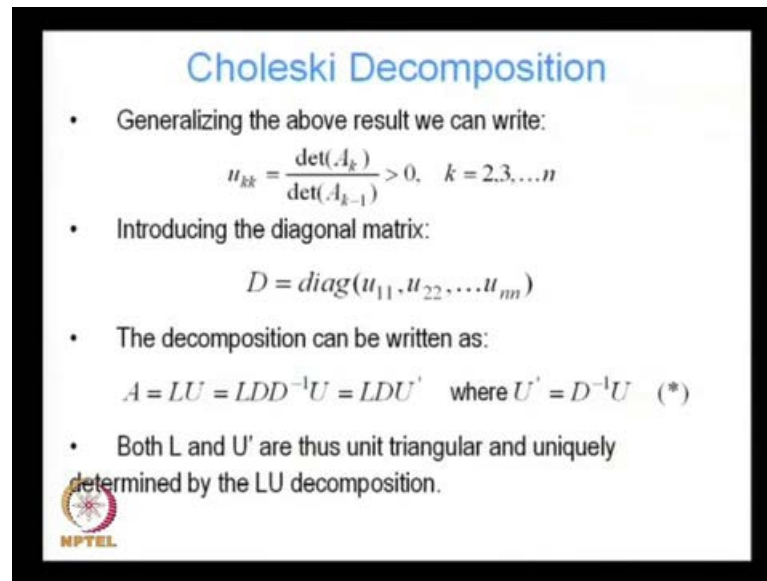
It is based on the following theorem which says that for a symmetric positive definite matrix, there is a unique upper triangular matrix with positive diagonal elements such that, A is equal to R transpose R. Let us recall our L U decomposition, where we said that

A can be written as the product of a lower triangular matrix and an upper triangular matrix. This is much more narrow than this, this criteria is much more narrow than this, it is says that A is equal to R transpose R, so basically what we are saying that a can be written in terms of an upper triangular matrix.

And the lower triangular matrix is just the transpose of the upper triangular matrix, recall from the L U theorem we have a is equal to L U, where  $u_{11}$  is equal to  $a_{11}$ , which is positive.  $u_{22}$  is equal to  $a_{22} - a_{21}a_{11}^{-1}a_{12}$  and using the transformation rule for L U decomposition or for gauss elimination for that matter,  $u_{22}$  is basically  $a_{22}$  minus  $a_{21}$  by  $a_{11}^{-1}$  by  $a_{12}$ , which can be written as determinant of A 2 by determinant of A 1.


Where we recall determinant of a k of with k can vary from 1 to n a k is the sub matrix form by the intersection of the first k rows, and the first k columns of A. So, A 2 is the sub matrix of A which is form by the intersection of the first two rows and the first two columns of A. So,  $u_{22}$  in the second diagonal element is nothing but, determinant of A two divided by determinant of A 1, and for positive definite matrices we recall that all the sub matrices for k equal to 1 to n, that is all these sub matrices A 1, A 2, A 3 up to A n, they must all have positive determinants. So, what does this means, so it means that  $u_{22}$  must always also be positive, since determinant of A 2 will always be greater than 0 and determinant of A 1 will always be greater than 0.

(Refer Slide Time: 03:29)



**Choleski Decomposition**

- Generalizing the above result we can write:
$$u_{kk} = \frac{\det(A_k)}{\det(A_{k-1})} > 0, \quad k = 2, 3, \dots, n$$
- Introducing the diagonal matrix:
$$D = \text{diag}(u_{11}, u_{22}, \dots, u_{nn})$$
- The decomposition can be written as:
$$A = LU = LDD^{-1}U = LDU' \quad \text{where } U' = D^{-1}U \quad (*)$$
- Both L and U' are thus unit triangular and uniquely determined by the LU decomposition.



Similarly, we can show that for any  $k$ ,  $u_{kk}$  the  $k$ th element on the diagonal of the upper triangular matrix, can be written as determinant of  $A_k$  divided by determinant of the sub matrix of  $A$  of order  $k-1$ , that is  $A_{k-1}$ . And since this is a positive definite matrix both determinant of  $A$  and determinant of  $A_{k-1}$  are positive, so that guarantees that  $u_{kk}$  will always be positive.

So, next let us introduce the diagonal matrix  $D$ , whose diagonal elements comprise the elements which we just computed  $u_{11}, u_{22}$  through  $u_{nn}$ , which we just showed how to compute just before this. In that case the decomposition, the  $LU$  decomposition can be written as  $A = LU$ , we can write it as  $LDD^{-1}U$  which is identical to  $LDU'$ . And then, write  $L = LDU'$ , where we define an intermediate matrix  $U'$   $U' = D^{-1}U$ , it is clear that both  $L$  and  $U'$  are unit triangular,  $L$  by definition is as got  $U'$  one on it is diagonal.


So,  $L$  by definition is unit triangular and now when we have defined  $U'$   $U' = D^{-1}U$ ,  $U'$  is also going to have unit values on it is diagonal, because  $D^{-1}$  contains the inverse of the diagonal elements, so  $D^{-1}U$  will always going to have one on it is diagonal, so  $U'$  is going to be an upper triangular matrix, only difference with  $U$  is that it has got 1 on it is diagonals. So, now we can write

A is equal to L D U prime, where both L and U prime are unit triangular and also they are uniquely determined by the L U decomposition. According to a theorem earlier which we encountered, which say that the L U decomposition when it exist, it has to be unique.

(Refer Slide Time: 05:48)

### Choleski Decomposition

- Since A is symmetric, we can write:
 
$$A = A^T = (U')^T D L^T \quad (**)$$
- Comparing equations (\*) and (\*\*) we get:
 
$$L^T = U' = D^{-1}U$$
- Denoting,
 
$$R = D^{-1/2}U \quad \text{where } D^{-1/2} \text{ has positive diagonalelements } u_{kk}^{-1/2}$$




Since A is symmetric, we can write A is equal to A transpose which means from ((Refer Time: 05:55)) this expression on the right from the last equation which means that, A transpose is equal to U prime transpose D L transpose. So, comparing this equation with the equation A equal to L D U prime you can by comparing terms, we can show that L transpose is equal U prime which is equal to D inverse of U.

Next we denote R another matrix with as D minus half U, where D minus half has positive diagonal elements  $u_{kk}^{-1/2}$ . So, R is D minus half U, how do you found D minus half, D minus half is we already know how to found D, we have from D taking the diagonal elements of U. And D minus half we just take the square root of the diagonal elements, it is just the square root of the diagonal elements of course, one by the square root of the diagonal elements and if we define R is equal to D minus half U.

(Refer Slide Time: 07:08)

### Choleski Decomposition

- Then:  
$$R^T R = U^T D^{-1/2} D^{-1/2} U = U^T D^{-1} U$$
- But:  
$$D^{-1} U = L^T \quad \therefore R^T R = U^T L^T$$
- Taking transpose of both sides, we therefore get:  
$$R^T R = L U = A$$
- Thus for symmetric positive definite matrices it is always possible to write:  
$$U = L^T$$




Then, we can write  $R^T R$  is equal to  $U^T D^{-1/2} D^{-1/2} U$ , which is identical to  $U^T D^{-1} U$ , since  $D^{-1/2}$  is a diagonal matrix,  $D^{-1/2} D^{-1/2} U$  which is equal to  $U^T D^{-1} U$ . ((Refer Time: 07:31)) But, let us recall  $L^T$  is equal to  $D^{-1} U$  which we have obtained earlier, therefore  $D^{-1} U$  is equal to  $L^T$ , therefore  $R^T R$  is nothing but,  $U^T L^T$  and taking the transpose of both sides, we get  $R^T R$  is equal to  $L U$  which is equal to  $A$ . Thus this shows that it is possible to write  $A$  as a product of a matrix  $R$  it is trans and it is transpose; and this also shows that, it is always possible to write  $U$  is equal to  $L^T$  for positive definite symmetric matrices.

(Refer Slide Time: 08:17)

### Choleski Decomposition

- Recalling that for the  $k^{\text{th}}$  step of LU decomposition we use the following equation:
 
$$a_{kj} = \sum_{p=1}^k m_{kp} u_{pj}, \quad j \geq k \quad a_{ik} = \sum_{p=1}^k m_{ip} u_{pk}, \quad i > k$$
- Then we can write:
 
$$a_{kj} = m_{kk} u_{kj} + \sum_{p=1}^{k-1} m_{kp} u_{pj} \quad j = k, k+1, \dots, n$$

Hence,  $m_{kk} u_{kj} = a_{kj} - \sum_{p=1}^{k-1} m_{kp} u_{pj} \quad (*)$



Let us go back to the LU decomposition and recall that for the  $K^{\text{th}}$  step of the LU decomposition, we use the following equation  $a_{kj}$  is equal to sum over  $p$  equal to 1 to  $k$   $m_{kp} u_{pj}$ , where  $j$  is greater than or equal to  $K$ , that we are constructing the part which is above the principle diagonal at or above the principle diagonal. And  $a_{ik}$  is equal to sigma  $p$  equal to 1 to  $k$   $m_{ip} u_{pk}$   $i$  greater than  $K$ , which is the part which is below the principle diagonal.

We can write this as  $a_{kj}$  is equal to  $m_{kk} u_{kj}$  plus sigma  $p$  equal to 1 to  $k$  minus 1  $m_{kp} u_{pj}$ , basically we have combining both these equations and writing it like that, where we have taken out the term, the  $K^{\text{th}}$  term and we have kept the rest of the terms within the sum. So, we can write to combine both the equations and write it like that, so we can write in that case  $m_{kk} u_{kj}$  just by moving the terms to the left hand side,  $m_{kk} u_{kj}$  to the left hand side,  $m_{kk} u_{kj}$  is equal to  $a_{kj}$  minus sigma  $p$  equal to 1 to  $k$  minus 1  $m_{kp} u_{pj}$ .

So, actually let me go back a little bit and maybe I should clarify ((Refer Time: 10:12)) this equation is nothing but, this equation and this equation we are going to handle in the next slide. So, it is not that it is combined I am missed spoke, but  $a_{kj}$  is just this equation, where have taken out the term  $K^{\text{th}}$  term outside the summation sign.

(Refer Slide Time: 10:35)


### Choleski Decomposition

Also,  $a_{ik} = m_{ik}u_{kk} + \sum_{p=1}^{k-1} m_{ip}u_{pk}$

$$m_{ik} = \frac{a_{ik} - \sum_{p=1}^{k-1} m_{ip}u_{pk}}{u_{kk}}, \quad i = k+1, k+2, \dots, n (**)$$

- If  $U=L^T$ , then  $u_{kk} = m_{kk}$  and  $u_{pk} = m_{kp}$
- From (\*) for  $j=k$ :
 
$$(m_{kk})^2 = a_{kk} - \sum_{p=1}^{k-1} m_{kp}u_{pk} = a_{kk} - \sum_{p=1}^{k-1} (m_{kp})^2$$

$$m_{kk} = [a_{kk} - \sum_{p=1}^{k-1} (m_{kp})^2]^{\frac{1}{2}}$$



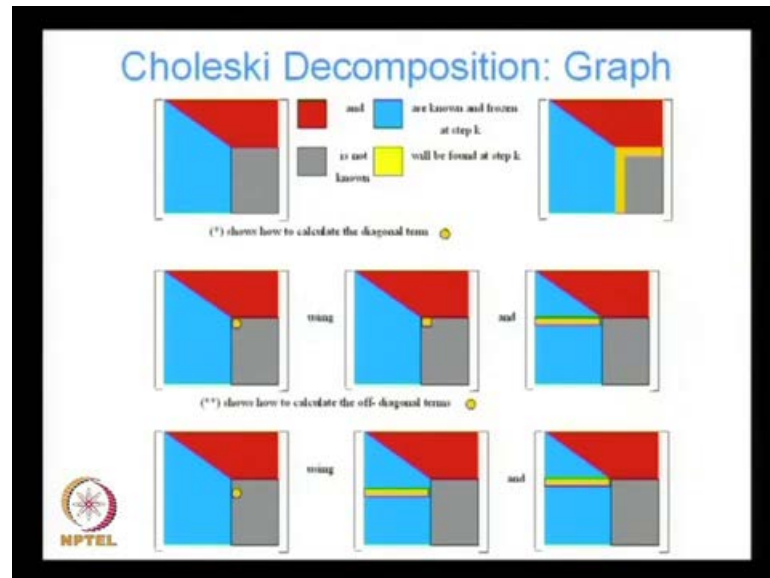
In the next slide, we have the second equation which is  $a_{ik} = m_{ik}u_{kk} + \sum_{p=1}^{k-1} m_{ip}u_{pk}$ , which is basically ((Refer Time: 10:48)) this equation, after I have taken out the  $k$ th term outside the summation sign. And we get that, so again rearranging terms we can write  $m_{ik}$  is equal to  $a_{ik}$  minus  $\sum_{p=1}^{k-1} m_{ip}u_{pk}$  divided by  $u_{kk}$ , when now  $i$  is equal to  $k+1, k+2$  through  $n$ .

If  $U$  is equal to  $L$  transpose, then  $u_{kk}$  is equal to  $m_{kk}$ , so  $m$  are the diagonal elements have to be the same, while the off diagonal elements suggest the transpose of each other. So,  $u_{pk}$  is equal to  $m_{kp}$ , since  $U$  is equal to  $L$  transpose recall the elements of  $m$  comprise the elements of  $L$ ,  $U$  comprises the elements of  $U$ . So, since  $U$  is equal to  $L$  transpose that means, the diagonal elements have to be equal  $L u_{kk}$  must be equal to  $m_{kk}$  and  $u_{pk}$  is equal to  $m_{kp}$ .

From the previous equation from star  $j$  is equal to  $k$ , we can write it as  $m_{kk}u_{kk}$ , which is and we know  $u_{kk}$  is equal to  $m_{kk}$ , so the left hand side becomes  $m_{kk}^2$ . And the right hand side becomes  $a_{kk} - \sum_{p=1}^{k-1} m_{kp}u_{pk}$ , let us go back and see what it look like, it look like  $m_{kp}u_{pk}$ . And we have utilize that in said  $m_{kp}u_{pk}$  and that is equal to  $a_{kk} - \sum_{p=1}^{k-1} m_{kp}^2$

square, because  $u_{pk}$  is equal to  $m_{kp}$  from the symmetry from  $U$  is equal to  $L$  transpose. So, we can write  $m_{kk}$  is equal to  $a_{kk}$  minus  $\sum_{p=1}^{k-1} u_{pk}^2$  to the power half, what does this tell us, this basically tells us how we can go about our algorithm.

(Refer Slide Time: 13:19)



So, what this tells us is becomes clear of we look at the next slide, so this is our Choleski decomposition graph, so this is the particular stage in a Choleski decomposition, suppose of at the  $K$  th step. At the  $K$  th of the Choleski decomposition, where the red part is already known that is the upper triangular part which is already we found, the blue part consist of the multiplies which have already being formed.

And the grey part is not known, not known means it is still got be modified it is already the original matrices known, but it has not yet reach it is final firm after Choleski decomposition. So, what this says is that at the  $K$  th step what we are going to do is, we are going to change this yellow part, the yellow strip corresponding to this column and the yellow strip is corresponds to the part of this row, this is going to be changed. So, at the end of the step, this grey region will become smaller it will become like this and this part is going to, we are going to find the new values of this part.



So, how do we go about doing this, well the previous equation ((Refer Time: 14:37)) this equation tells us how we are going to change the diagonal element, this tells us how we are going to change  $m_{k,k}$  this term, which is basically the diagonal element here. And it is says that how are you going to form the diagonal element, well the diagonal element is going to deformed by  $a_{k,k}$ , which is the existing value here minus  $m_{k,p}^2$ , where  $p$  is equal to  $1$  to  $k-1$ . So, this  $m_{k,p}^2$  terms are the terms here, the terms on this row up to  $k-1$ , so up to the diagonal term.


So, as you can see the information that it means for updating this element is already known, because this we already know the final form of this, and this is just the diagonal the current value of the diagonal. So, it uses these two values, this minus this square to form the diagonal element, so next let us look at how it updates the off diagonal elements. So, we know how it forms this element, we know how we forms the diagonal element, so if we can makes find out how we can update this column, we are all said.

Because, since this is symmetric matrix if we have updated the column, we have also updated the row, so you only need to find out how to update this part of this column. And that is given by the second equation, the second equation which is double star  $m_{i,k}$  ((Refer Time: 16:32)) this equation, this equation tells me how to update that column and let us see how we do that.

(Refer Slide Time: 16:39)

**Choleski Decomposition**

- From (\*\*):
$$m_{ik} = \frac{a_{ik} - \sum_{p=1}^{k-1} m_{ip} m_{kp}}{m_{kk}} \quad i = k+1, \dots, n$$
- The right hand side of these expressions contain known quantities from the previous k-1 steps
- The method is known as Choleski's method or the square root method.

 However computing square roots is expensive.

This equation can be rearranged as  $m_{ik}$ , basically I am rewriting ((Refer Time: 16:48)) this equation  $m_{ik}$  is equal to  $a_{ik}$  minus sigma  $p$  equal to 1 to  $k$  minus 1  $m_{ip} m_{kp}$  divided by  $m_{kk}$   $i$  equal to  $k+1$  through  $n$ . And you can see from a previous picture, so for instance if we want to update this element on the column, we just need to take the product of this vector, this vector with that vector. So, this times that I mean as a column vector, this is a row times that becomes a column vector row times this column gives me this updated value.

So, again to compute the updated value, we only read terms in the blue or red regions of the matrix, that is we only deal with terms which already known which is the significant advantage of this method. The right hand side of these expressions contain known quantities from the previous  $k$  minus 1 steps, so this minimizes storage this is more ((Refer Time: 18:01)). And this method is known as Choleski's method or the square root method, why is it call this square root, because to compute the diagonal terms we have to take the square root of this expression.


But, it is well known I mean it is that computing square roots is expensive, square root computation in any numerical, any computer the square root is at the most expensive, probably the one of the most expensive numerical operations that you can perform, other

than for instance taking trigonometric functions such as sin, cosine and so on, and so forth. So, we would like to avoid taking square roots if possible, in order to will increase the efficiency of our computer program.

(Refer Slide Time: 18:47)

### Avoiding square roots

- It may be preferable to obtain an upper triangular matrix  $U$  by symmetric Gaussian elimination and then solve the system  $Ax = b$  using the relation:
 
$$A = R^T R = U^T D^{-1/2} D^{-1/2} U$$
- This allows decomposition of the system into the triangular system:
 
$$U^T y = b, \quad Ux = Dy$$
- Recall that Gaussian elimination for symmetric positive definite systems requires no pivoting at all.



So, what is normally done is that, people decide to instead of going through this processes, instead of doing Choleski decomposition, exactly as I discussed earlier. What they do is to go ahead with Gaussian elimination and then, obtained the triangular matrix  $U$  and then, compute the  $R$  matrix, so we use this relation  $A$  is equal to  $R$  transpose  $R$  is equal to  $U$  transpose  $D$  minus half  $D$  minus half  $U$ .

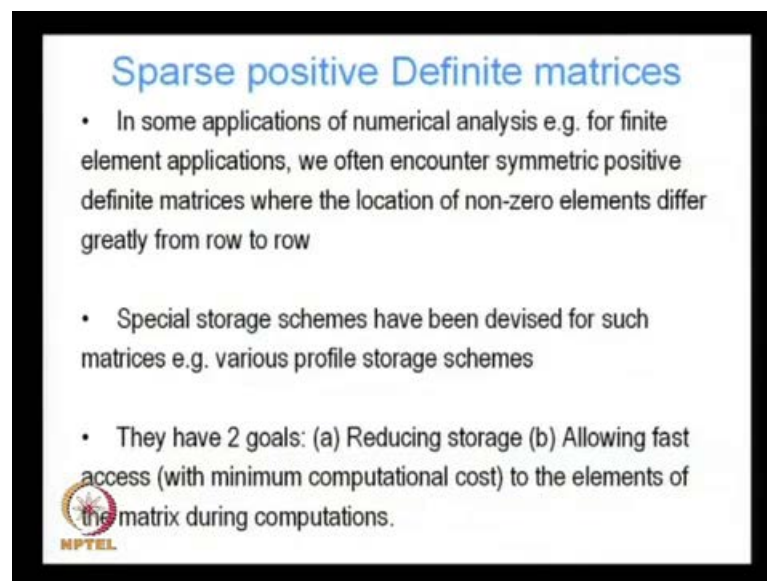
So, compute  $R$  in terms of  $D$  and  $U$  and this allows us to write the decomposition of the system, as  $A$  is equal to  $U$  transpose  $D$  minus 1  $U$ , so  $A$  is equal to  $U$  transpose  $A$   $x$  is equal to  $U$  transpose  $D$  minus 1  $U$   $x$  and then, we write  $D$  minus 1  $U$   $x$  as  $y$ . So, we write this system becomes  $U$  transpose  $y$  is equal to  $b$  and then, after we have solved for  $y$  we solved for  $x$  by solving the system  $U$   $x$  equal to  $D$   $y$ . So, again we solve two triangular systems  $U$  transpose  $y$  equal to  $b$  and  $U$   $x$  equal to  $d$   $y$ , again it is noticeable here that  $L$  is no where formed.

So, we are taking the advantage of the fact that for a symmetric matrix, symmetric

positive definite matrix you can write instead of  $A$  is equal to  $LU$ , you can write  $A$  is equal to  $R^T R$ . So, we are taking advantage of the fact without having to compute this square root, which makes the Choleski decomposition expensive, why do not we have to compute the square root, because we are never explicitly computing  $D$  minus half, we are always computing  $D$  only in this way.

So, this takes advantage of the fact that for a symmetric positive definite matrix, you can write  $A$  as  $R^T R$ , but it does not require taking the square root. And let us recall that, but the prerequisite for this is to perform the Gaussian elimination, because we have to calculate  $U$ . But, recall that for Gaussian elimination for positive symmetric definite matrices, we do not require any pivoting at all; so Gaussian elimination for positive symmetric definite matrices is relatively assumed.

(Refer Slide Time: 21:26)



**Sparse positive Definite matrices**

- In some applications of numerical analysis e.g. for finite element applications, we often encounter symmetric positive definite matrices where the location of non-zero elements differ greatly from row to row
- Special storage schemes have been devised for such matrices e.g. various profile storage schemes
- They have 2 goals: (a) Reducing storage (b) Allowing fast access (with minimum computational cost) to the elements of the matrix during computations.

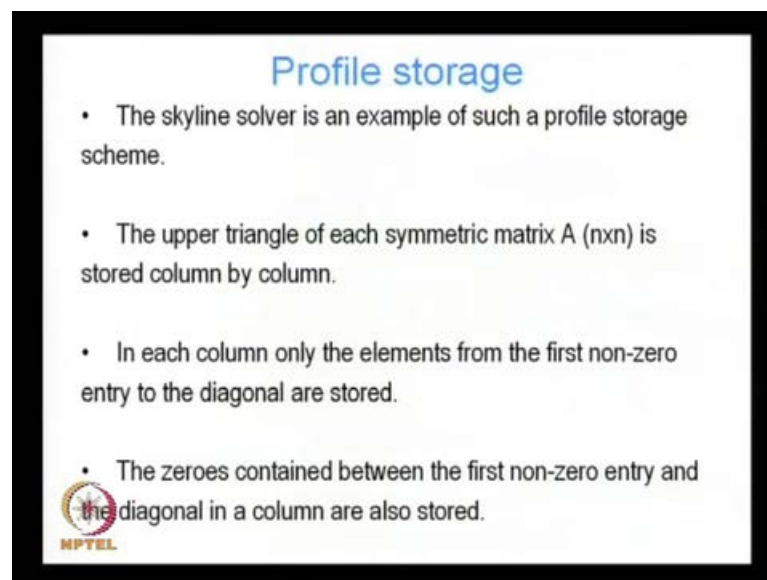
MPTEL

Next let us switch to slightly different topic which is related, but which is a very great importance and when you are trying to solve practical problems, which have got very large coefficient matrices. We often encounter symmetric positive definite matrices, for instance in finite elements by the location of the non-zero elements differ greatly from row to row. For instance, in one row there might be, the size of the row is  $n$ , you might have it may be very fully populated or it may be almost fully populated.

While in another row or column, it might have just a few non zero elements which are probably located localize near the diagonal. So, for these matrices, we want to take a advantage or certain special storage scheme, because if there are lot of zeros in a matrix, we do not want to store the zeros. Because, they occupy memory, they increase the size of the matrix, if we can figure out clever way of reducing the storage we have and are if the efficiency of algorithm increases significantly.

So, special storage schemes have been device for such matrices for example, various profile storage schemes, which have two goals, first of all the reduce storage, they reduce the requirement to store all the zero elements. So, the reduce the storage and number 2 they allow fast access to the elements of the matrix during the computations, when we are processing in a matrix, we want to access the elements of the matrix, we want to able to do it as fast as possible. In order to reduce the computational cost, so these special storage schemes have these two goals, so reducing storage and allowing fast access.

(Refer Slide Time: 23:25)



**Profile storage**

- The skyline solver is an example of such a profile storage scheme.
- The upper triangle of each symmetric matrix  $A$  ( $n \times n$ ) is stored column by column.
- In each column only the elements from the first non-zero entry to the diagonal are stored.
- The zeroes contained between the first non-zero entry and the diagonal in a column are also stored.

MPTEL

One of the most well known storage schemes for this is known as a skyline solver, and in the skyline solver the upper triangle of each symmetric matrix is stored column by column. But, the entire column is not stored, in each column only the elements from the first non zero entry to the diagonal are stored, for each column all zero entries from the


from the first entry up to the non zero entry, they are not stored. So, only the diagonal and the first non zero entries in that column are store, the zeroes contained between the first non zero entry and the diagonal are of course store, but any zero is above the first non zero entry are not stored.

(Refer Slide Time: 24:17)

### Skyline solver

- A matrix A is thus represented by a vector:
 
$$\mathbf{s} = (s_1, s_2, \dots, s_{\mu(n)})$$
- A pointer vector contains the location of the diagonal elements:  $\boldsymbol{\mu} = (\mu(1), \mu(2), \dots, \mu(n))$  where  $\mu(i)$  gives the location in  $\mathbf{s}$  of the diagonal elements of the  $i^{\text{th}}$  column of A
- Hence if we want to extract the element  $a_{ij}$  of the matrix where  $j > i$  then the following algorithm is adopted:
 

If  $(j-i) < \mu(j) - \mu(i)$ , i.e. the element  $a_{ij}$  lies within the "skyline", then  $a_{ij} = s_p$ , where  $p = \mu(i) + j - 1$



So, matrix is thus represented by a vector S is equal to S 1, S 2 through to S mu n, and in addition we need a pointer vector which contains the location of the diagonal elements. This pointer vector is mu which has got n entries with mu 1 representing the location in S, where the first diagonal elementary sides of a S, mu 1 is always got to be 1, because for the first column the diagonal element is the first entry. So, mu 1 is equal to 1, but mu 2 is gives me the location in that row vector S, where the second diagonal element resides.

Similarly, mu 3 gives the location in that vector S where the third diagonal elementary sides and mu n, gives the location in S, where the n th diagonal elementary sides which always has to be the last entry in the S vector. So, if we want to extract element  $a_{ij}$  of the matrix where j is greater than i, then we can use this following rule and which says that if j minus 1 is less than mu j minus mu i which means that, the element  $a_{ij}$  lies within the skyline, then  $a_{ij}$  is equal to  $S_p$ , where p is equal to mu i plus j minus 1.

(Refer Slide Time: 25:52)

**Skyline solver**


If  $(j-i) > \mu(j) - \mu(i)$ , i.e. the element  $a_{ij}$  lies outside the "skyline". Since all elements outside the skyline are zero, then  $a_{ij} = 0$ .

- Example

$$A = \begin{bmatrix} 25 & 3 & 0 & 0 & 0 & 0 \\ 3 & 21 & 2 & 4 & 0 & 0 \\ 0 & 2 & 23 & 0 & 0 & 0 \\ 0 & 4 & 0 & 22 & 0 & 0 \\ 0 & 0 & 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 & 0 & 35 \end{bmatrix}$$

$s = (25, 3, 21, 2, 23, 4, 0, 22, 20, 35)$

$\mu = (1, 3, 5, 8, 9, 10)$



So, this will become clear from an example which we are going to look at earlier, if on the other hand, if  $j - i$  is greater than  $\mu(j) - \mu(i)$ , that is the element  $a_{ij}$  lies outside the skyline, then we do not need to store it (Refer Time: 26:04). Since, outside the skyline we can be by definition in that  $a_{ij}$  is equal to 0, for instance let us look at this matrix  $A$  equal to this.

And you can see that in this matrix, we are going to store only the elements in the upper triangular part, this is the symmetric matrix of course, and we are only going to store the elements in the upper triangular matrix, which are below the first non zero term in a column. So, in the first column we are going to store 25, in the second column we are only going to store 3 and 21, since 3 is non 0, but in the third column we are not going to store this leading 0. We are only going to store 2 and 23, we are only going to store the all the elements at or below the first non zero entry in the upper triangular part of the matrix, in this column.

So, the first column we are going to store 25, in the second column we are going to store 3 as well as 21, since 3 is non 0, but in the third column we are not going to store this leading 0. We are only going to store 2 and 23, we are only going to store the all the elements at or below the first non zero entry in the upper triangular part of the matrix, in this column.


So, we are going to store 2 and 23, in the fourth column again we are not going to store this 0, we are going to store 4, but we have also that to store 0, because 4 0 is occurring after the first non zero entry in the column. So, I going to store 4 0 and 22, but in the fifth column, we are going to not going to store all these 0's, so I going to save a lot of storage. And then, we are just going to stored the diagonal element which is 20 same thing for the last column which is a sixth column, we are not going to store all these leading 0's and we are just going to store 35.

So, my S vector will be this, so we have reduced the storage significantly however, we have also need to store this mu vector, this mu vector tells me which is the position in this column which has the first diagonal matrix. So, the first diagonal element is at the first entry in this columns, so it is stores 1, the second diagonal element which is 21 is at the third location in this column, so it is stores 3. The third diagonal element 23 is at the 1, 2, 3, 4, 5th location in that column, so it is stores 5, so it is stores the diagonal pointers and it stores this skyline like this. So, the total storage of course, is given by the length of this S vector and the length of the mu vector.

(Refer Slide Time: 28:57)

### Skyline solver

- The total storage = length (s) + length (μ)=16
- Hence saving in storage =  $\frac{n^2}{2} - 16 = \frac{6^2}{2} - 16 = 2$
- Savings are much larger for larger sparse matrices e.g. a 100 x 100 matrix with 400 non-zero elements
- Even if a compact storage scheme is adopted, it is of limited usefulness if successive stages of a numerical algorithm introduces a large number of non-zero elements.



And in this case we can see this length of the S vector is probably 10 and the length of the mu vector is 6, so the total length of this storage is 16. However, how much have be



saved storage by if we actually stored the full upper triangular matrix, you would have to store  $n^2$  by two terms, so we have save this storage by  $n^2$  by 2 minus 16, which in this cases  $36$  by  $2$  that is  $18$  minus  $16$  is equal to  $2$  which the saving is comparatively less for the small matrix.

But, for real problems real world problems where you can have thousand by thousand or may be hundred thousand by hundred thousand matrices, you save a lot of storage by this profile using this skyline solver. However, even if a compact storage scheme is adopted it is of limited usefulness, if successive stages of a numerical algorithm introduces a large number of non zero elements.

The efficiency of ((Refer Time: 30:22)) this scheme is because there are lot of zero elements that is why we can save storage, but suppose we start with an initial matrix which has got a lot of zero elements. But, then during a Gaussian elimination a lot of, because of pivoting a lot of those zero elements become non zero, in that case again now storage is going to increase. So, it is advisable to try a pivoting scheme which does not increase the number of non zero elements.


(Refer Slide Time: 31:01)

### Sparseness preservation

- Hence it is desirable to use an algorithm which is both numerically stable and sparseness preserving
- Partial pivoting for Gaussian elimination e.g. is not normally advantageous from the sparseness preservation point of view.
- An alternative pivoting scheme known as "threshold" pivoting can be adopted. In this scheme the pivot element is chosen to be:
 

$$a_{kk}^{(k)} \text{ if } \left| a_{kk}^{(k)} \right| \geq \tau \max_{i>k} \left| a_{ik}^{(k)} \right|$$

where  $\tau$  is a pre-set threshold value lying between zero and one



Hence, it is desirable to use an algorithm, which is both numerically stable and

sparseness preserving, why because numeric stable, because we know that unless we do pivoting for non positive definite matrices, we can now in count a numerical in stability. So, some amount of pivoting is absolutely essential, but we want to also achieve the competing goal, which is that we want to preserve sparseness not only we want to ensure numerical stability, but we also want to preserve sparseness.

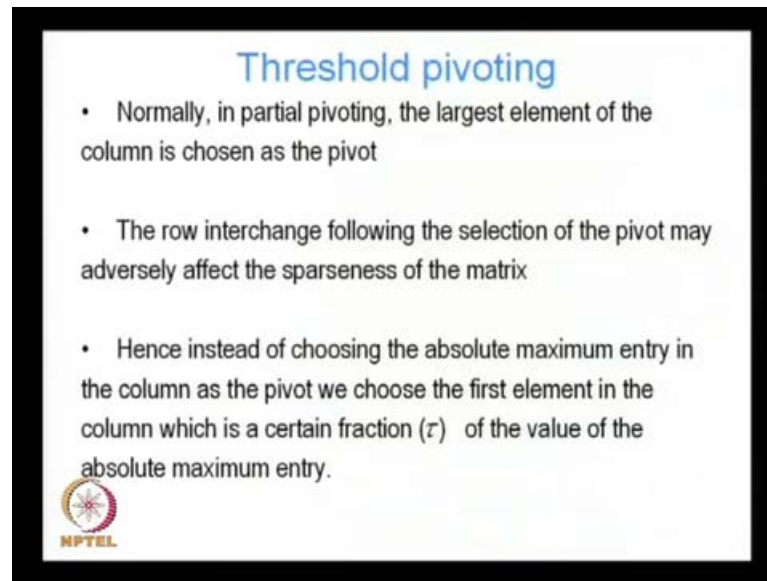
Partial pivoting for Gaussian elimination is not normally advantageous from the sparseness preservation point of view, an alternative pivoting scheme known as threshold pivoting scheme can be adopted. In this pivoting scheme the pivot element is chosen to be  $a_{ik}$  at the  $k$ th step the pivoting element is chosen to be  $a_{ik}$  prime  $k$ , if  $a_{ik}$  prime  $k$  is greater than  $\tau$  times maximum of  $|a_{ik}|$ ,  $i$  is greater than  $k$  and  $\tau$  is a pre set threshold value lying between 0 and 1.

So, for come for partial pivoting for full partial pivoting, if we choose as the pivot the element on the column in the  $k$ th column, which has the largest absolute value right, for partial pivoting we choose as the  $k$  as the pivot the element on the  $k$ th column which has the largest absolute values. So, basically for partial pivoting we will choose maximum of  $|a_{ik}|$  greater than  $k$  that is what we would choose for full partial pivoting.

Therefore, threshold pivoting we are saying that we do not have to choose this maximum value we can choose a value which is may be, if  $\tau$  is equal to half which is half this maximum value. If my pivot if  $a_{ik}$  prime which is which is of course, less than  $i$  the assumption is  $k$  prime is less than  $i$ . So, I am looking down, I am going down from the  $k$ th row and I am moving downwards and I am going to choose if I am going to do full partial pivoting I will move downwards throughout the entire column. And choose the element in that column which has the maximum positive value as the pivot.


But now if I am not being full partial pivoting I do not have to choose the absolute maximum value I can choose as pivot an element which is may be half the maximum value. In if my largest maximum value is at distance of 5 from  $k$  and if my half the maximum values at distance of 3 from  $k$ . Then if I choose 3 as the pivot rather than if I choose a  $a_{3k}$  as the pivot rather than a  $a_{5k}$  I am the possibility of my increasing the of my reducing the sparseness becomes less, because I am going to do less switching.

(Refer Slide Time: 34:22)



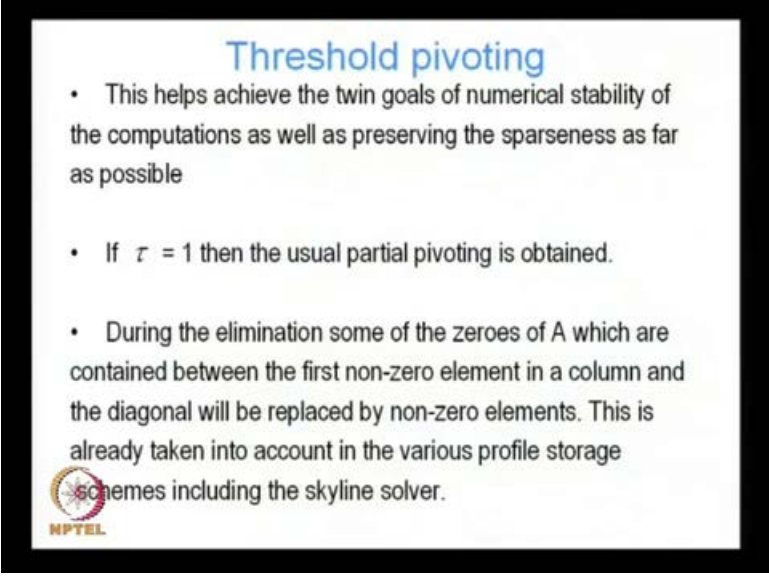
### Threshold pivoting

- Normally, in partial pivoting, the largest element of the column is chosen as the pivot
- The row interchange following the selection of the pivot may adversely affect the sparseness of the matrix
- Hence instead of choosing the absolute maximum entry in the column as the pivot we choose the first element in the column which is a certain fraction ( $\tau$ ) of the value of the absolute maximum entry.



Normally in partial pivoting the largest element of the column is chosen as the pivot; however, the row interchange following the selection of the pivot may adversely affect the sparseness of the matrix. Hence instead of choosing the absolute maximum entry in the column as the pivot, we choose the first element of the column which is a certain fraction of the value of the absolute maximum entry; which it is seen this algorithm helps in preserving the sparseness, it is better preserving sparseness than doing full partial pivoting.

(Refer Slide Time: 35:01)



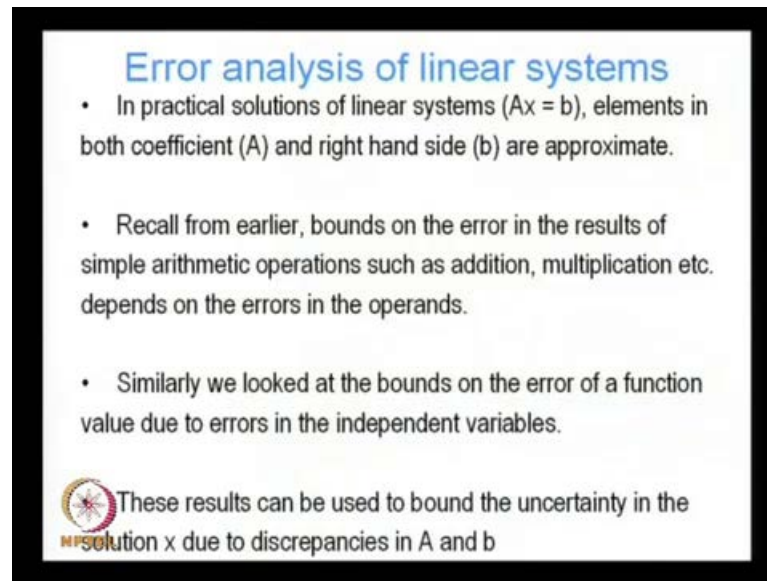
**Threshold pivoting**

- This helps achieve the twin goals of numerical stability of the computations as well as preserving the sparseness as far as possible
- If  $\tau = 1$  then the usual partial pivoting is obtained.
- During the elimination some of the zeroes of  $A$  which are contained between the first non-zero element in a column and the diagonal will be replaced by non-zero elements. This is already taken into account in the various profile storage schemes including the skyline solver.

NPTEL


This helps achieves the twin goals of numerical stability of the computations, because we are doing a some pivoting at least as well as preserving the sparseness as far as possible. If tau is equal to 1 then of course, the usual partial pivoting is obtained during elimination some of the zeroes of a which are contained between the first non zero element in a column and the diagonal will be replaced by non zero elements. This is not a problem, because our profile storage scheme accounts for that because we stored all the elements which are below at or below the first non zero element in a column. So, if there is zeroes below the first non zero element will automatically stored, so that is not a problem.

(Refer Slide Time: 35:51)



**Error analysis of linear systems**

- In practical solutions of linear systems ( $Ax = b$ ), elements in both coefficient ( $A$ ) and right hand side ( $b$ ) are approximate.
- Recall from earlier, bounds on the error in the results of simple arithmetic operations such as addition, multiplication etc. depends on the errors in the operands.
- Similarly we looked at the bounds on the error of a function value due to errors in the independent variables.

 These results can be used to bound the uncertainty in the solution  $x$  due to discrepancies in  $A$  and  $b$

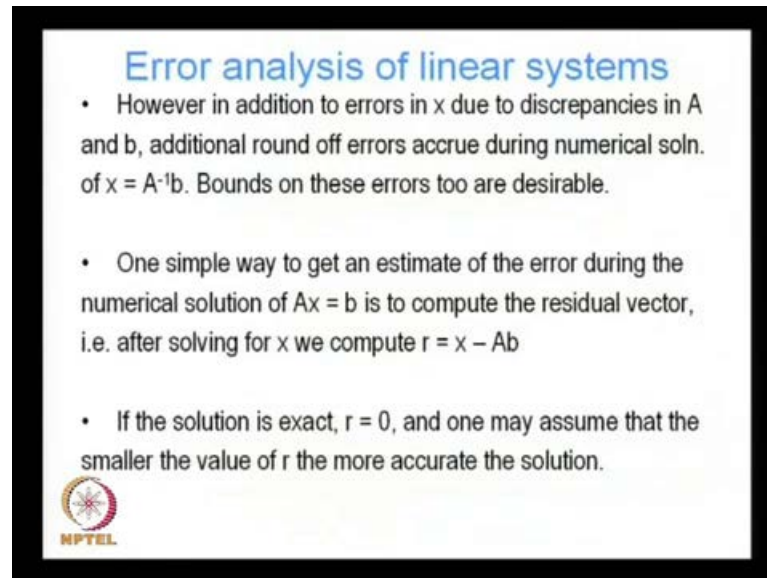
Next, so with all that discussion next we move on to a main focus in this lecture which is error analysis of linear systems. In practical solutions of linear systems  $Ax = b$  elements in both the coefficient and the right hand side  $b$  are approximate particularly when  $a$  and  $b$  represents certain physical quantities they are always there is always going to be may be  $a$  and  $b$  has been obtained from experimental data, so they will always be approximate.

Recall from earlier bounds on the error in the results of simple arithmetic operations such as addition multiplication etcetera depend on the errors in the operands for instance we found bounds on the errors on in addition. And we said that that bound on the error, the error in addition is bounded by the some of the error in the operands right; similarly we found errors in the bounds on the errors in multiplication and division

Also we looked at bounds on the error of a function due to errors in the independent variable. So, if  $y$  is equal to function of  $x_1, x_2, x_3$  to  $x_n$  then we found a bound on the error in  $y$ , if we know the errors in the independent variables  $x_1$  through  $x_n$ . So, we derive that relationship, these results can be used to bound the uncertainty in the solution  $x$  due to discrepancies in  $A$  and  $b$ . So, due to discrepancies in  $A$  and  $b$  because of errors we want to find out what will be the maximum possible error in the in  $x$ . So, suppose we


know that error in A, suppose we know the error in b we want to find out what will be the maximum possible error in my solution x.

(Refer Slide Time: 37:53)



**Error analysis of linear systems**

- However in addition to errors in x due to discrepancies in A and b, additional round off errors accrue during numerical soln. of  $x = A^{-1}b$ . Bounds on these errors too are desirable.
- One simple way to get an estimate of the error during the numerical solution of  $Ax = b$  is to compute the residual vector, i.e. after solving for x we compute  $r = x - Ab$
- If the solution is exact,  $r = 0$ , and one may assume that the smaller the value of r the more accurate the solution.

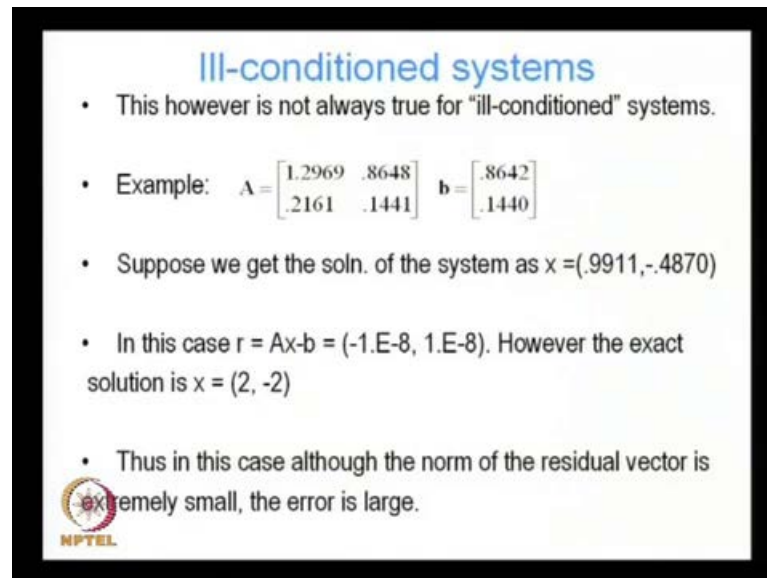
 NPTEL

In addition to the errors in a due to discrepancies in A and b additional round of errors accrue during numerical solution of x is equal to A inverse b bounds on these errors too are desirable. So, there are two sorts of errors in linear systems the first sort of error is due to discrepancies in the values of A and b when there is some error in A and b are not known exactly because may be they are coming from experimental data's. So, there are errors in the elements of a in the elements of b, but then there are additional errors which we have during the Gaussian elimination, whichever solution proceed we adopt to solve the x is equal to A inverse b additional errors we will on during the solution. So, we want to know bounds on those errors too.

One simple way to get an estimate of the error during the numerical solution of A x is equal to b is to compute the residual vector, that is after we solve for x we compare r is equal to x minus A b actually this should be a inverse b x minus A inverse b; because we have solve for x by taking a inverse b. So, we compare that x is equal to A inverse b and we get our residual. If the solution is exact the residual is exact r is equal to 0 and one we assume that the smaller the value of r the more accurate the solution right, so the residual


is 0, so then we can say that my solution is more correct.

(Refer Slide Time: 39:46)



**Ill-conditioned systems**

- This however is not always true for "ill-conditioned" systems.
- Example:  $A = \begin{bmatrix} 1.2969 & .8648 \\ .2161 & .1441 \end{bmatrix}$   $b = \begin{bmatrix} .8642 \\ .1440 \end{bmatrix}$
- Suppose we get the soln. of the system as  $x = (.9911, -.4870)$
- In this case  $r = Ax - b = (-1.E-8, 1.E-8)$ . However the exact solution is  $x = (2, -2)$
- Thus in this case although the norm of the residual vector is extremely small, the error is large.




However this is not always true for ill condition systems let us considered the system a with a coefficient matrix A given by the matrices square brackets there and b by the column vector given there. So, suppose we get the solution of the system as x is equal to the 0.9911 and minus 0.4870 in this case r is equal to A x minus b is given by minus 1 into e to the power minus 8 and 1, e to the power minus 8. So, the residual is very small; however, the exact solution for this problem is x is equal to 2 minus 2. So, even though the residual is small a x where x has been computed from by solving that linear system a x is differing from b by a very small magnitude by minus of magnitude by column vector whose largest value is largest entry is 1 into 10 to the power minus 8; however, even though the residual is small the exact solution is way off, thus in this case although the norm of the residual vector extremely small the error is large why is this. So, this is, so because my matrix a is extremely ill condition.

(Refer Slide Time: 41:17)

**Ill-conditioned systems**

- This occurs because the system is extremely ill-conditioned.
- This is evident if we look at the system after eliminating  $x_1$  when we get:  
$$a_{22}^{(2)} x_2 = b_2^{(2)} \text{ where}$$
$$a_{22}^{(2)} = .1441 - .8648 \times \frac{.2161}{1.2969} = .1441 - .1440999923$$
$$= 10^{-8}$$
- Since the pivot is extremely small: very small changes in  $a_{22}^{(2)}$  will lead to large changes in  $x_2$ .

 The size of residual is not indicative of error in such cases.

So, this will become evident if we look at the Gaussian elimination of the system where at the second after eliminating  $x_1$  at the second step we get  $a_{22}^{(2)} x_2 = b_2^{(2)}$  and if we calculate  $a_{22}^{(2)}$  it becomes  $10^{-8}$ . So, this system by definition has got very poor numerical stability. Since the pivot is extremely small very small changes in  $a_{22}^{(2)}$  will lead to large changes in  $x_2$ . So, basically my solution for this system by Gaussian elimination is going to be arbitrary, minor change is minor perturbation will give very large differences in the solution. So, for ill conditioned systems the size of the residual is not indicative of the error at all.



(Refer Slide Time: 42:17)

**Matrix and vector norms**

- To quantitatively discuss errors in the solution of linear systems, we have to decide on measures of magnitude of error
- Recall the  $L_2$  norm of a vector  $x$  and a matrix  $A$ :
$$\|x\|_2 = \sqrt{x \cdot x} \quad \|A\|_2 = \sqrt{a_{ij} a_{ij}}$$
- The infinite norm are similarly given by:
$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i| \quad \|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$$
- In general, if a matrix and vector norm satisfies  $\|Ax\| \leq \|A\| \|x\|$  for any  $A$  and  $x$ , then the norm is said to be consistent.

NPTEL

So, that is for ill condition systems, therefore we have to figure out some better way to decide how to figure out the norm of the error and then to bound the error to decide what are the postulate bounds on those errors. To quantitatively discuss errors in the solution of linear systems, we have to decide on measures of magnitude of error; recall the  $L_2$  norm of a vector and a matrix which we discussed earlier is given by norm of a vector  $x$  is given by root of  $x$  dotted with  $x$ . And the norm of a matrix  $L_2$  norm of a matrix is given by this square root of the inner product of  $A$  with itself that is  $\sqrt{a_{ij} a_{ij}}$ .

The infinite norm is given by for a vector  $x$  infinity is equal to maximum of  $x_i$  maximum of the absolute value of  $x_i$  for  $i$  is equal to 1 through  $n$  and the infinite norm of  $A$  is given by maximum of  $\sum_{j=1}^n |a_{ij}|$  for  $i$  is equal to 1 through  $n$ . So, basically we are summing the absolute values of all the entries in a row and then we are taking the maximum over the column maximum over each column. So, for basically what we are doing is that we are looking at it row by row and we are summing the absolute values of the entries of all the elements in that row right; and then we are taking the maximum of those values over all the rows, so that gives me my infinite norm of a matrix.


In general if a matrix and vector norm satisfies this relation which is  $\|Ax\| \leq \|A\| \|x\|$

than or equal to norm of a times norm of x for any a and x then the norm is said to be consistent. So, both my both my 1 2 norm and my infinite norm, i consistent norms because the satisfies this relation norm of a x is norm of a x means the norm of the vector which is given by a x is lesser than or equal to the norm of a which is a matrix norm times the norm of x which is a vector norm. So, if any norms satisfy this relationship, then for any matrix and any vector a and x then the norms are said to be consistent norms.

(Refer Slide Time: 45:01)

**Bound on error due to coeff matrix**

- We will use these norms to develop error bounds for the linear system assuming A and b are known and then trying to estimate the effect of perturbations in A and b
- Let  $\delta b$  be the perturbation in b. Then
 
$$A(x + \delta x) = b + \delta b. \quad \text{Hence, } \delta x = A^{-1} \delta b$$
- Then for any consistent norm, in particular for the infinite norm:
 
$$\|\delta x\| \leq \|A^{-1}\| \|\delta b\|$$

 MPTEL

So, we will use these norms to develop error bounds for the linear system assuming a and b are known and then try to estimate the effect of perturbations in a and b, first let us assume that we have perturbations in b only. So, we do not know exactly where b is, we know b plus delta b, where delta b is the perturbation in b, because there is a perturbation and b the solution to my linear system is no longer going to be x, it is going to be x plus delta x.

So, a x I am going to solve the system a x plus delta x is equal to b plus delta b, so this gives me a x is since a x is equal to b this gives me delta x is equal to a inverse of delta b and if for any consistent norm recall how we have define consistent norm here. So, for a consistent norm this has to be true, so delta norm of delta x must be lesser than or equal

to norm of a inverse delta b. And this norm of delta x must be equal to norm of a inverse delta b and norm of a inverse delta b must be lesser than or equal to norm of a inverse times norm of delta b, since that since my norms are consistent norms.

(Refer Slide Time: 46:26)

### Bound on error due to coeff matrix

- Next consider perturbations in A i.e.  $\delta A$ 


$$(A + \delta A)(x + \delta x) = b.$$

$$\therefore A\delta x + \delta A(x + \delta x) = 0.$$

$$\therefore \delta x = -A^{-1} \delta A(x + \delta x)$$
- Thus again for any consistent norm:
 
$$\|\delta x\| \leq \|A\|^{-1} \|\delta A\| \|x + \delta x\|$$

$$\therefore \frac{\|\delta x\|}{\|x + \delta x\|} \leq \|A\| \|A\|^{-1} \frac{\|\delta A\|}{\|A\|} = K(A) \frac{\|\delta A\|}{\|A\|} \quad (*)$$

where  $K(A) = \|A\| \|A\|^{-1}$



Next let us consider the effect of perturbations in a, that is delta a, so a is not known correctly. So, suppose we just know a plus delta a again that solution is going to change this solution is no longer going to be x it is going to be x plus delta x. So, now, we have the equation a plus delta a, x plus delta x equal to b, so we have a delta x plus delta a x plus delta x is equal to 0, because again a x is equal to b. So, a x and b cancels out from both sides this gives me delta x is equal to minus a inverse delta a x plus delta x.


Again for any consistent norm delta x is equal to norm of a inverse delta a x plus delta x, but norm of a inverse delta a x plus delta x must be lesser than norm a inverse times norm of delta a x plus delta x, which must again we less than norm of a inverse times norm of delta a times norm of x plus delta x. So, we get this expression and dividing both sides by norm of x plus delta x, we have this expression norm of a norm of a inverse norm of delta a by norm a, norm of a norm of a of course, this is cancelling now norm of a inverse norm of delta a.

And this quantity is what is known as the condition number of a matrix this is basically, this quantity is known as the condition number of the matrix norm of a times norm of a inverse and actually there is a printing error here. So, this is actually norm of a times norm of a inverse, the inverse should be inside the norm. So, norm of a times norm of a inverse times norm of delta a by norm of a. So, this is the condition number times norm of delta a by norm of a.

(Refer Slide Time: 48:46)

### Bound on error due to RHS

- $K(A)$  is the condition number of the matrix  $A$  with respect to the current norm.
- If the condition number is large, then from (\*) small relative perturbations in  $A$  will produce large changes in  $x$ .
- Recall  $\|\delta x\| \leq \|A^{-1}\| \|\delta b\|$   
 $\therefore \frac{\|\delta x\|}{\|x\|} \leq \frac{\|A^{-1}\| \|\delta b\|}{\|x\|}$



$K(A)$  is the condition number of the matrix  $A$  with respect to the current norm if the condition number is large then from star small relative perturbations in  $A$  will produce large changes in  $x$  let us look at this expression again. So, if my condition number is large small relative perturbations in  $A$  that is small values of if normal delta  $A$  by norm of  $A$  is small, but my condition number is large, then my solution the change in the solution will also be large. So, this is very important the condition number is the crucial factor. So, even for small changes in  $A$  in when norm of delta  $A$  is small, I am going to get a large change my error is going to be large for a large condition number.

So, again let us recall norm of delta  $x$  is lesser than or equal to  $A$  inverse norm of delta  $b$  which we have obtained earlier, norm of delta  $x$  is lesser than or equal to norm of  $A$  inverse norm of delta  $b$ . So, again dividing both sides by norm of  $x$  norm of delta  $x$  by

norm of x is lesser than or equal to norm of A inverse norm of delta b by norm of delta x

(Refer Slide Time: 50:20)

### Bound on error due to coeff matrix

- Next consider perturbations in A i.e.  $\delta A$ 


$$(A + \delta A)(x + \delta x) = b.$$

$$\therefore A\delta x + \delta A(x + \delta x) = 0.$$

$$\therefore \delta x = -A^{-1} \delta A(x + \delta x)$$
- Thus again for any consistent norm:
 
$$\|\delta x\| \leq \|A\|^{-1} \|\delta A\| \|x + \delta x\|$$

$$\therefore \frac{\|\delta x\|}{\|x + \delta x\|} \leq \|A\| \|A\|^{-1} \frac{\|\delta A\|}{\|A\|} = K(A) \frac{\|\delta A\|}{\|A\|} \quad (*)$$

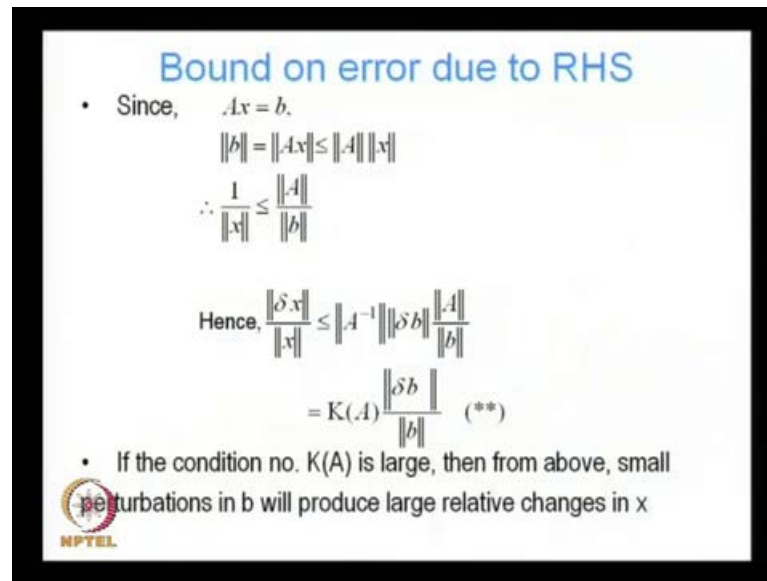
where  $K(A) = \|A\| \|A\|^{-1}$



Since  $Ax = b$  we can write norm of b is equal to norm of x and again because the norms are consistent we can write this is less than or equal to norm of A times norm of x. So,  $\|b\| \leq \|A\| \|x\|$  from this expression  $\|x\| \leq \|A\|^{-1} \|b\|$  is lesser than or equal to norm of A inverse times norm of b, then because of this norm of delta x by norm of x is lesser than or equal to norm of A inverse times norm of delta b by norm of b, how do we get this.

Well we use the previous relation which we got here which says that the delta x by norm of x is lesser than or equal to norm of A inverse norm of delta b by norm of x. So, we use that relation here to get this expression and again we identify norm of A inverse times norm of A is A here it is right. So, it is a norm of A inverse not A norm of A inverse. So, norm of A inverse times norm of A that is if the condition number of A, this is equal to norm of delta b by norm of b. So, again this again shows that if the condition number of k A is that condition number, k A is large then small perturbations in b is going to give me large relative changes in x. So, in both cases condition number is crucial.

(Refer Slide Time: 51:52)




**Bound on error due to RHS**

- Since,  $Ax = b$ ,  
$$\|b\| = \|Ax\| \leq \|A\| \|x\|$$
$$\therefore \frac{1}{\|x\|} \leq \frac{\|A\|}{\|b\|}$$

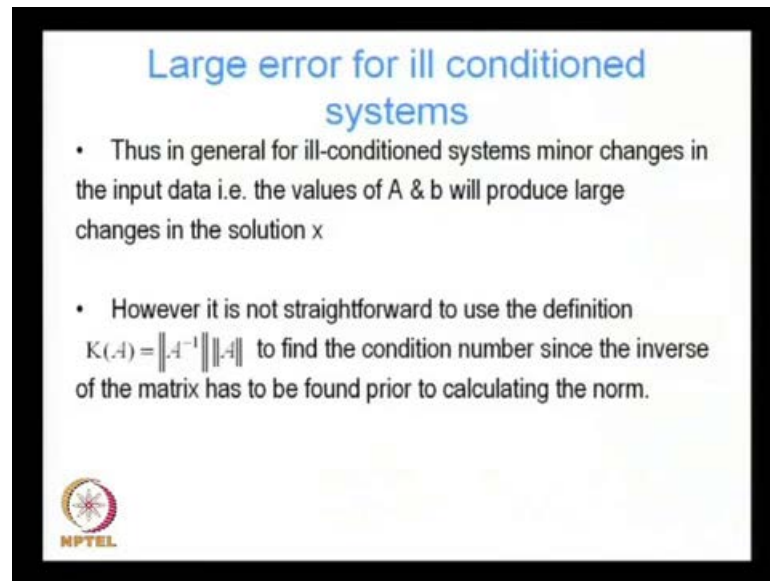
Hence, 
$$\frac{\|\delta x\|}{\|x\|} \leq \|A^{-1}\| \|\delta b\| \frac{\|A\|}{\|b\|}$$
$$= K(A) \frac{\|\delta b\|}{\|b\|} \quad (**)$$

- If the condition no.  $K(A)$  is large, then from above, small perturbations in  $b$  will produce large relative changes in  $x$




So, in the previous case also we saw that if the condition number is large then small changes in the coefficient matrix are going to produce large changes in the solution. Similarly, here we show that if the condition number is large small changes in the right hand side is also going to give lead to large relative errors in the solution. So, the condition number of a matrix is a very crucial quantity in determine how much is the error going to be if either my coefficient matrix or my right hand side is slightly off has got errors.

(Refer Slide Time: 52:34)



**Large error for ill conditioned systems**

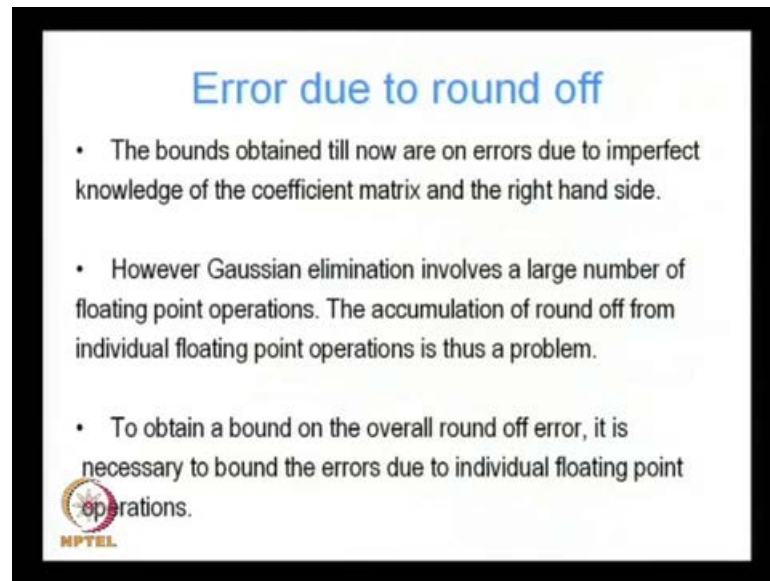
- Thus in general for ill-conditioned systems minor changes in the input data i.e. the values of A & b will produce large changes in the solution x
- However it is not straightforward to use the definition  $K(A) = \|A^{-1}\| \|A\|$  to find the condition number since the inverse of the matrix has to be found prior to calculating the norm.

 NPTEL

Thus in general for ill conditioned systems minor changes in the input data that is the values of A and b will produce large changes in the solution x; however, it is not straight forward to calculate the condition number why because if we have to compute the condition number as the norm of a inverse stands norm of a we have to first invert a and the inversion of a is a highly expensive operation. So, it is not always use or it is not always practicable to compute the condition number.


So, we have to come up with certain other ways of computing the bounds on the error in a linear system, but condition number if known that gives me very detailed information about the error bound if there is an error in the coefficient matrix or in the right hand side.

(Refer Slide Time: 53:44)



**Error due to round off**

- The bounds obtained till now are on errors due to imperfect knowledge of the coefficient matrix and the right hand side.
- However Gaussian elimination involves a large number of floating point operations. The accumulation of round off from individual floating point operations is thus a problem.
- To obtain a bound on the overall round off error, it is necessary to bound the errors due to individual floating point operations.

 MPTEL

So, in the next part of this lecture we are going we have planning to talk about errors due to round off. So, up till now we have talked on errors due to in perfect knowledge of the coefficient matrix and the right hand side; however, Gaussian elimination. So, we will suppose we know on coefficient matrix in a right hand side and then we do the Gaussian elimination. But, Gaussian elimination involves a large number of floating point operations the accumulation of round off from this individual floating point operation is also going to delete to more errors it is going to accumulate errors are going to accumulate. Therefore it is necessary to obtain a bound on the overall round off error due to Gaussian elimination, but in order to do that we have to bound the errors due to the individual floating point operations and we are going to talk about this in the next lecture in this series.

Thank you very much.