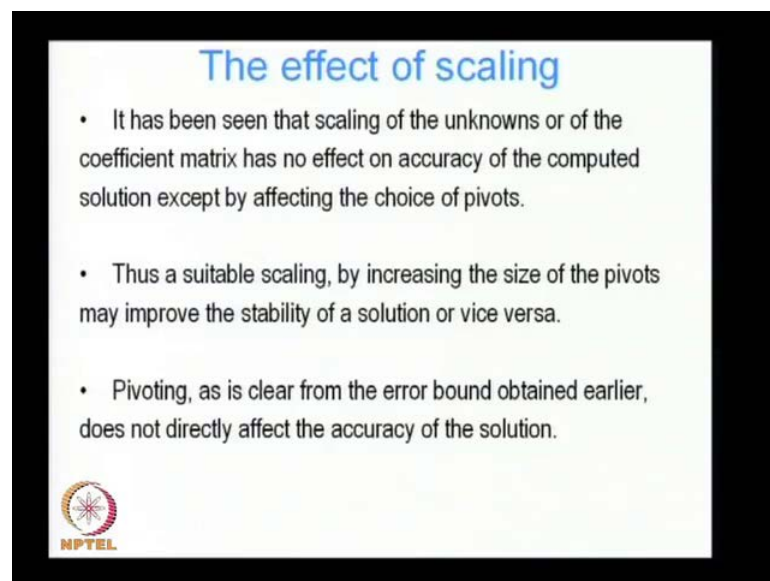


**Numerical Methods in Civil Engineering**  
**Prof. Arghya Deb**  
**Department of Civil Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture - 8**  
**Interactive Methods for Solving Linear Systems**

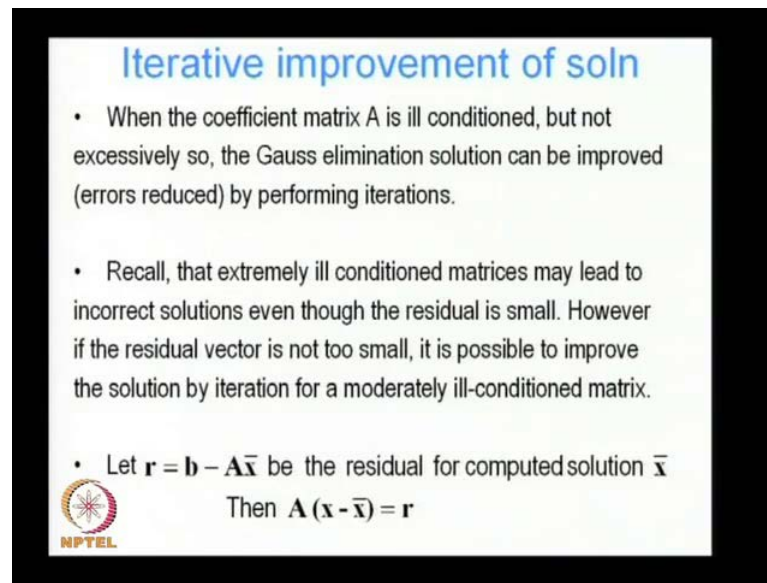
Lecture on series and numerical methods in civil engineering we are going to focus on iterative methods for solving linear systems.

(Refer Slide Time: 00:26)




In our previous lecture, we talked about error analysis of linear systems and we showed the scaling of the unknowns. The coefficient matrix has no effect on the accuracy of the computed solution except by affecting the choice of pivots. By choosing a suitable scaling, we can improve the stability characteristics of the equation of the solution procedure, but it affects the accuracy only indirectly.

(Refer Slide Time: 01:00)



**Iterative improvement of soln**

- When the coefficient matrix  $A$  is ill conditioned, but not excessively so, the Gauss elimination solution can be improved (errors reduced) by performing iterations.
- Recall, that extremely ill conditioned matrices may lead to incorrect solutions even though the residual is small. However if the residual vector is not too small, it is possible to improve the solution by iteration for a moderately ill-conditioned matrix.
- Let  $\mathbf{r} = \mathbf{b} - \mathbf{A}\bar{\mathbf{x}}$  be the residual for computed solution  $\bar{\mathbf{x}}$   
Then  $\mathbf{A}(\mathbf{x} - \bar{\mathbf{x}}) = \mathbf{r}$

 NPTEL

When the coefficient matrix is ill conditioned, but it is not excessively ill conditioned the Gauss elimination solution can be improved by performing iterations. What do we mean basically we can improve the accuracy of the Gauss elimination solution by performing relatively cheap iterations on the solution; let us recall that if the matrix is extremely ill conditioned. Then, we may get incorrect solutions even though there we have a very small residual. Otherwise, if the residual is reduced, the residual we improve the accuracy of the solution, so long as the coefficient matrix is moderately ill conditioned by reducing the residual we can improve we automatically improve the accuracy of the solution.

However, if the coefficient matrix is extremely ill conditioned, then we have seen earlier that even for a totally incorrect solution, we can get a really small residual in that case. If the coefficient matrix is extremely ill conditioned, it does not make any sense to try to improve the residual because we are not guaranteed that we are improving the solution by reducing the residual. Otherwise, if the coefficient matrix is not too ill conditioned by reducing the residual, we can improve the accuracy of the solution. What do we mean by extremely ill conditioned or moderately ill conditioned, we will talk about those definitions later on in this lecture.

Let us denote our residual  $\mathbf{r}$  as  $\mathbf{b} - \mathbf{A}\bar{\mathbf{x}}$  where  $\bar{\mathbf{x}}$  is the computed solution by Gauss elimination, if  $\bar{\mathbf{x}}$  were equal to  $\mathbf{x}$ , which is the exact solution, then our residual


would be 0, but since it is not, we have a residual which is given by  $b - Ax$ . We can therefore write  $Ax - \bar{x} = r$ , basically we are re writing this equation, where we are replacing  $b$  by  $Ax$  the exact solution and we are then we can write the residual as  $r = Ax - \bar{x}$ .

(Refer Slide Time: 03:26)

**Iterative improvement of soln**

- Suppose that the decomposition  $\bar{L}\bar{U}$  of  $\bar{A}$  is known.
- Since the error due to the decomposition is bounded from Eqn. (\*) and provided partial pivoting has been performed can a priori be shown to be small, the correction  $\delta x = x - \bar{x}$  can be computed as:  $\bar{L}\bar{U}\delta x = r$
- We can solve this equation by solving two triangular systems sequentially:
 
$$\bar{L}y = r$$

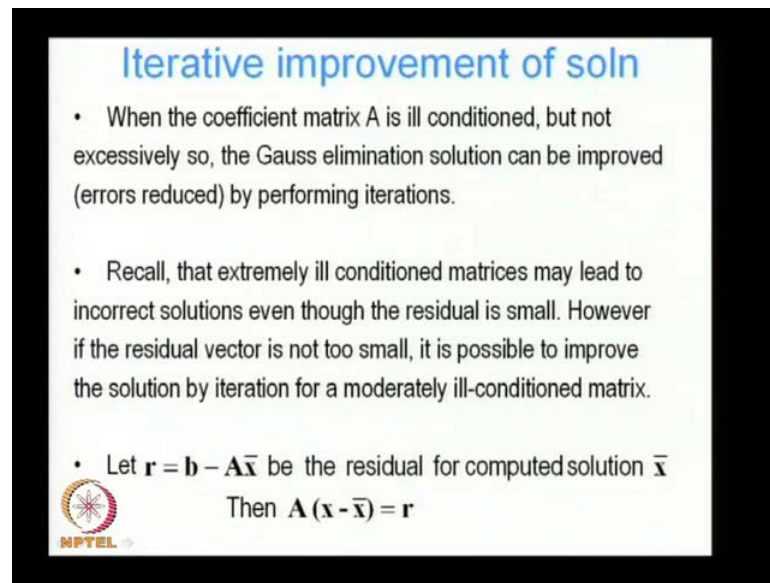
$$\bar{U}\delta x = y$$



Now, suppose we have the decomposition of a bar into L bar and U bar is known, this is sort of because we are we are trying to improve the accuracy of the solution which has been already computed by Gaussian elimination. So, that means that we have the L U decomposition of the coefficient matrix the L U decomposition of the coefficient matrix by Gauss elimination is L bar U bar so the assumption is that L bar U bar is known L bar U bar being the approximate means that it is there is an error there because of round off. We have seen the extent of round off, but L bar and U bar is the correct decomposition of a up to the round of error.


Since, the error due to the decomposition is bounded; we have seen that the error due to the decomposition is bounded provided partial pivoting has been performed. It can be bounded up priori and we have shown that that error is small the correction the L x which is the true solution minus  $\bar{x}$  the solution that we have that we have obtained by Gaussian elimination, we can write it as  $\bar{L}\bar{U}\delta x = r$  basically.

(Refer Slide Time: 04:49)



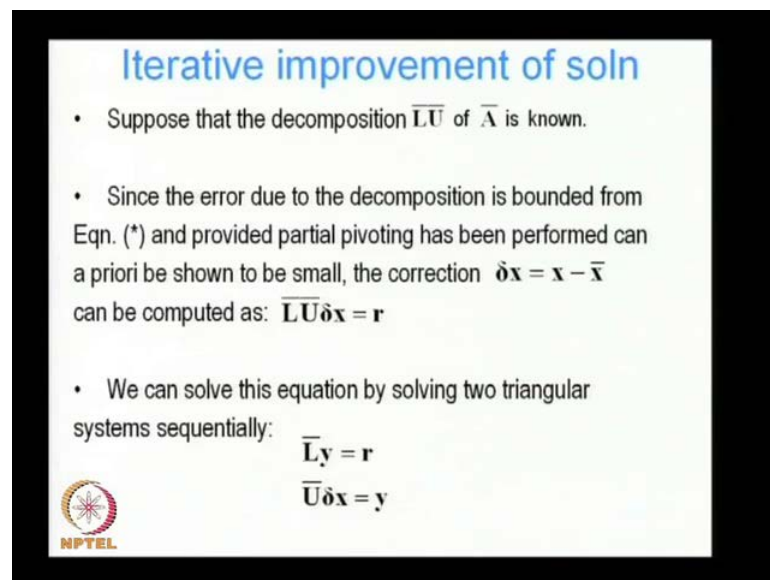
**Iterative improvement of soln**

- When the coefficient matrix  $A$  is ill conditioned, but not excessively so, the Gauss elimination solution can be improved (errors reduced) by performing iterations.
- Recall, that extremely ill conditioned matrices may lead to incorrect solutions even though the residual is small. However if the residual vector is not too small, it is possible to improve the solution by iteration for a moderately ill-conditioned matrix.
- Let  $\mathbf{r} = \mathbf{b} - \mathbf{A}\bar{\mathbf{x}}$  be the residual for computed solution  $\bar{\mathbf{x}}$   
Then  $\mathbf{A}(\mathbf{x} - \bar{\mathbf{x}}) = \mathbf{r}$




Let us go back to the previous equation, we have  $\mathbf{A}(\mathbf{x} - \bar{\mathbf{x}}) = \mathbf{r}$  we have defined  $\delta\mathbf{x} = \mathbf{x} - \bar{\mathbf{x}}$  and  $\mathbf{A}$ , we are replacing it by its decomposition by its  $L U$  decomposition.

(Refer Slide Time: 05:07)



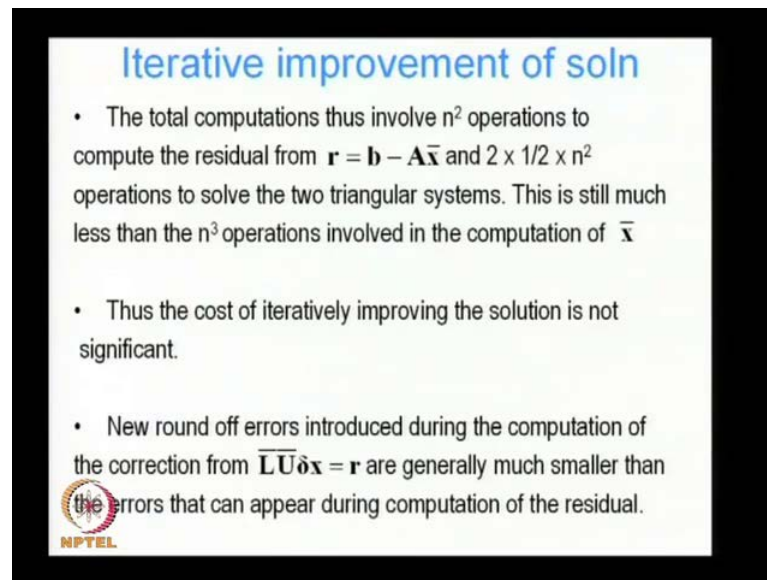
**Iterative improvement of soln**

- Suppose that the decomposition  $\bar{L}\bar{U}$  of  $\bar{A}$  is known.
- Since the error due to the decomposition is bounded from Eqn. (\*) and provided partial pivoting has been performed can a priori be shown to be small, the correction  $\delta\mathbf{x} = \mathbf{x} - \bar{\mathbf{x}}$  can be computed as:  $\bar{L}\bar{U}\delta\mathbf{x} = \mathbf{r}$
- We can solve this equation by solving two triangular systems sequentially:  
$$\bar{L}\mathbf{y} = \mathbf{r}$$
$$\bar{U}\delta\mathbf{x} = \mathbf{y}$$



So, we get  $\bar{L}\bar{U}\delta\mathbf{x} = \mathbf{r}$ , so we can solve this system sequentially by solving two triangular systems by denoting  $\bar{U}\delta\mathbf{x}$  as an intermediate vector  $\mathbf{y}$ , we first we solve for  $\mathbf{y}$   $\bar{L}\mathbf{y} = \mathbf{r}$ , then once we know  $\mathbf{y}$ , we can solve for  $\delta\mathbf{x}$ .

(Refer Slide Time: 05:32)



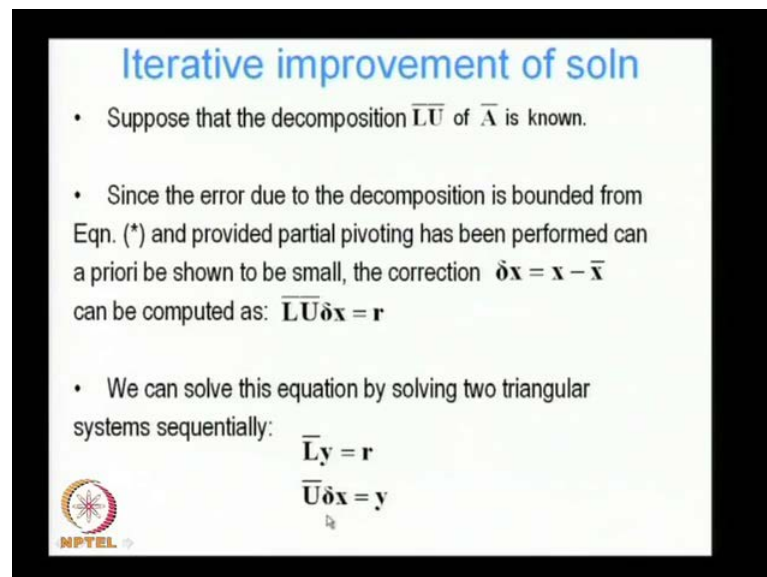
**Iterative improvement of soln**

- The total computations thus involve  $n^2$  operations to compute the residual from  $\mathbf{r} = \mathbf{b} - \mathbf{A}\bar{\mathbf{x}}$  and  $2 \times 1/2 \times n^2$  operations to solve the two triangular systems. This is still much less than the  $n^3$  operations involved in the computation of  $\bar{\mathbf{x}}$
- Thus the cost of iteratively improving the solution is not significant.
- New round off errors introduced during the computation of the correction from  $\overline{\mathbf{L}}\overline{\mathbf{U}}\delta\mathbf{x} = \mathbf{r}$  are generally much smaller than the errors that can appear during computation of the residual.

NPTEL

The total computations in this improvement to the original solution involve  $n$  square operations due to the computation of the residual in order to compute the residual. We have to compute  $\mathbf{b} - \mathbf{A}\bar{\mathbf{x}}$ ,  $\mathbf{A}$  is an  $n$  by  $n$  matrix  $\bar{\mathbf{x}}$  is a  $n$  by  $1$  vector, so this involves  $n$  square operations.

(Refer Slide Time: 06:00)



**Iterative improvement of soln**

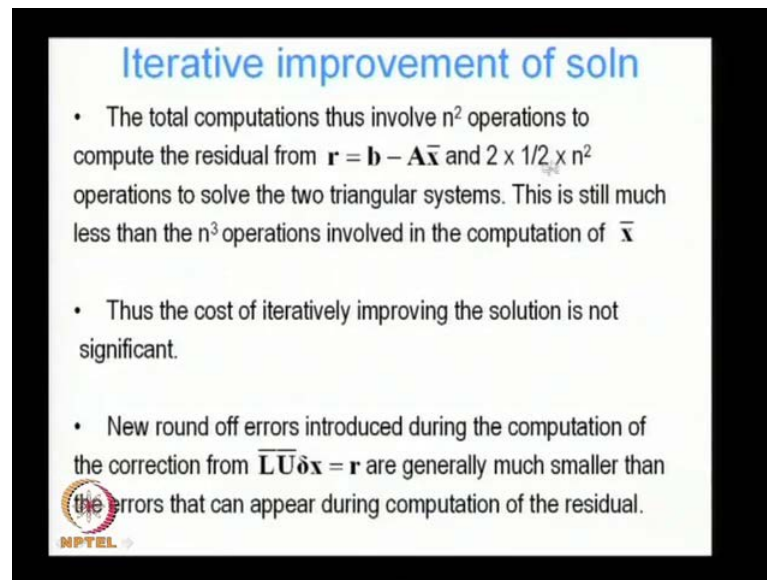
- Suppose that the decomposition  $\overline{\mathbf{L}}\overline{\mathbf{U}}$  of  $\overline{\mathbf{A}}$  is known.
- Since the error due to the decomposition is bounded from Eqn. (\*) and provided partial pivoting has been performed can a priori be shown to be small, the correction  $\delta\mathbf{x} = \mathbf{x} - \bar{\mathbf{x}}$  can be computed as:  $\overline{\mathbf{L}}\overline{\mathbf{U}}\delta\mathbf{x} = \mathbf{r}$
- We can solve this equation by solving two triangular systems sequentially:

$$\overline{\mathbf{L}}\mathbf{y} = \mathbf{r}$$
$$\overline{\mathbf{U}}\delta\mathbf{x} = \mathbf{y}$$

NPTEL

This  $n$  square operations plus we need to solve the two triangular systems  $\overline{\mathbf{L}}\mathbf{y} = \mathbf{r}$  and  $\overline{\mathbf{U}}\delta\mathbf{x} = \mathbf{y}$ .

(Refer Slide Time: 06:11)



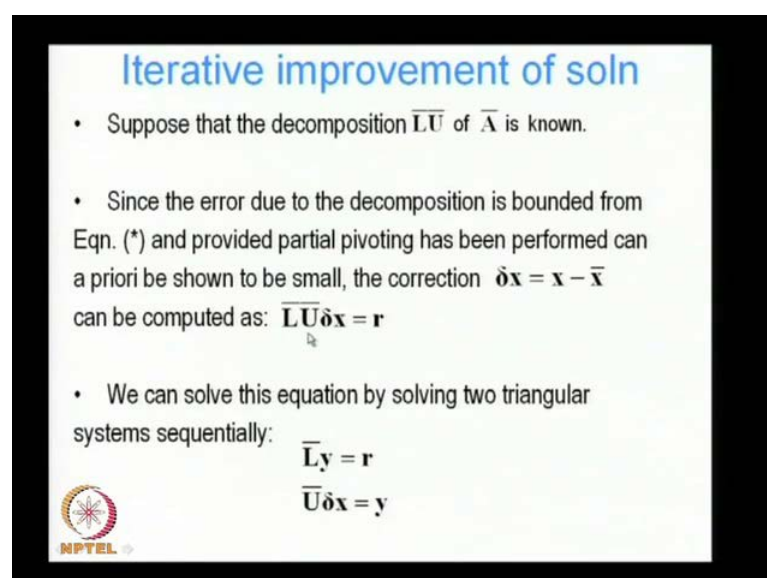
**Iterative improvement of soln**

- The total computations thus involve  $n^2$  operations to compute the residual from  $\mathbf{r} = \mathbf{b} - \mathbf{A}\bar{\mathbf{x}}$  and  $2 \times \frac{1}{2} \times n^2$  operations to solve the two triangular systems. This is still much less than the  $n^3$  operations involved in the computation of  $\bar{\mathbf{x}}$
- Thus the cost of iteratively improving the solution is not significant.
- New round off errors introduced during the computation of the correction from  $\overline{\mathbf{L}}\overline{\mathbf{U}}\delta\mathbf{x} = \mathbf{r}$  are generally much smaller than the errors that can appear during computation of the residual.

NPTEL

So, each of these solutions of this triangular systems involves half times  $n$  square operations we have seen that earlier. So, 2 times half  $n$  square that is equal to  $n$  square, so the operations required to improve the solution as of are of the order of the  $n$  square, but recall what was the total expense in the Gaussian elimination. That was of the order of  $n$  cube, so the computations the cost of the computation the cost of the number of operations. Hence, the cost of the computations involved in improving the solution is a small fraction of the total cost obtained in getting the solution, which is of the order of  $n$  cube, thus the cost of the iteratively improving the solution is not significant.

(Refer Slide Time: 07:01)



**Iterative improvement of soln**

- Suppose that the decomposition  $\overline{\mathbf{L}}\overline{\mathbf{U}}$  of  $\overline{\mathbf{A}}$  is known.
- Since the error due to the decomposition is bounded from Eqn. (\*) and provided partial pivoting has been performed can a priori be shown to be small, the correction  $\delta\mathbf{x} = \mathbf{x} - \bar{\mathbf{x}}$  can be computed as:  $\overline{\mathbf{L}}\overline{\mathbf{U}}\delta\mathbf{x} = \mathbf{r}$
- We can solve this equation by solving two triangular systems sequentially:  
$$\overline{\mathbf{L}}\mathbf{y} = \mathbf{r}$$
$$\overline{\mathbf{U}}\delta\mathbf{x} = \mathbf{y}$$

NPTEL



One may argue that new round of errors are introduced during the computation of the correction from  $\bar{L}\bar{U}\bar{x} = \mathbf{r}$  is equal to  $\mathbf{r}$ , but those errors are not significant because that is mostly back substitution type operations. We have seen that back substitution the errors are negligible compared to the errors during the Gaussian elimination, so those errors are not significant.

(Refer Slide Time: 07:22)

**Iterative improvement of soln**

- The total computations thus involve  $n^2$  operations to compute the residual from  $\mathbf{r} = \mathbf{b} - \mathbf{A}\bar{\mathbf{x}}$  and  $2 \times 1/2 \times n^2$  operations to solve the two triangular systems. This is still much less than the  $n^3$  operations involved in the computation of  $\bar{\mathbf{x}}$
- Thus the cost of iteratively improving the solution is not significant.
- New round off errors introduced during the computation of the correction from  $\bar{L}\bar{U}\delta\mathbf{x} = \mathbf{r}$  are generally much smaller than the errors that can appear during computation of the residual.

NPTEL

So, the new round of errors involved in which occur during the computation of the correction  $\delta\mathbf{x}$  are generally much smaller than the error that can appear during the computation of the residual. So, basically this process is two step, first step is computing the residual like this and then solving this this equation to find  $\delta\mathbf{x}$ . The main error during this improvement can occur during this operation and why that is it because of this product is just because of this subtraction. We are subtracting  $\mathbf{A}\bar{\mathbf{x}}$  from  $\mathbf{b}$  and both  $\mathbf{A}\bar{\mathbf{x}}$  and  $\mathbf{b}$  are approximately of the same magnitude.


Since, they are of the same magnitude, there will be cancellation errors and the chances of errors occurring here are significant. We have seen that in probably the second or third lecture that when we subtract two for instance two scalars in this case we are subtracting two vectors with all of whose all of whose components are approximately of the same magnitude. In case of scalars, if we subtract two scalars, which are approximately of the same magnitude, the chances of errors are very high.

So, similarly the main error during this during this improvement to the solution occurs during the computation of the residual  $\mathbf{r} = \mathbf{b} - \mathbf{A}\bar{\mathbf{x}}$ , relatively insignificant errors occur during this operation during the actual computation of the correction.

(Refer Slide Time: 09:00)

**Iterative improvement of soln**

- The residual computation  $\mathbf{r} = \mathbf{b} - \mathbf{A}\bar{\mathbf{x}}$  is prone to cancellation errors because it involves taking the difference of two vectors each of whose components are very close.
- Errors can be reduced if the terms of the product
 
$$\mathbf{A}\bar{\mathbf{x}} = \sum_{k=1}^n a_{ik} \bar{x}_k$$
 are accumulated in double precision (2t digit) storage even though  $a_{ik}$  and  $\bar{x}_k$  are stored as t digit numbers.

 This gives a more accurate value of  $\mathbf{r}$ .

The residual computation  $\mathbf{r}$  is equal to  $\mathbf{b}$  minus  $\mathbf{A}\bar{\mathbf{x}}$  is prone to cancellation errors. It involves taking the difference of two vectors each of whose components are very close errors can be reduced if the terms of this product  $\mathbf{A}\bar{\mathbf{x}}$  are accumulated in double precision storage even though  $a_{ik}$  and  $\bar{x}_k$  are stored as t digit numbers. So, we can improve this the error in this computation by storing by doing this computation  $a_{ik}$  and  $\bar{x}_k$  are already stored in the computer.

They are stored as single precision numbers normal precision, they are stored with up to t digits, but when we compute  $\mathbf{A}\bar{\mathbf{x}}$  we are storing it as two t digits. So, we try to preserve increase the accuracy of this operation by storing it as two digit, two t digit numbers and this gives me a more accurate value of the residual.

Since, I am computing this more accurately,  $\mathbf{r}$  is the residual is going to be more accurate because  $\mathbf{b}$  is known  $\mathbf{b}$  is known  $\mathbf{b}$  is a given vector and since I am computing this product more accurately my residual will also be more accurate.



(Refer Slide Time: 10:24)


**Iterative improvement of soln**

- Successive iterations can be performed to improve the computed solution in the above manner. The iterative scheme is as follows:  $x^{(1)} = \bar{x}$

Compute  $x^{(s)}$ ,  $s = 2, 3, \dots$  as:

$$r^{(s)} = b - Ax^{(s)}$$
$$\bar{L}(\bar{U}\delta x^{(s)}) = r^{(s)}$$
$$x^{(s+1)} = x^{(s)} + \delta x^{(s)}$$

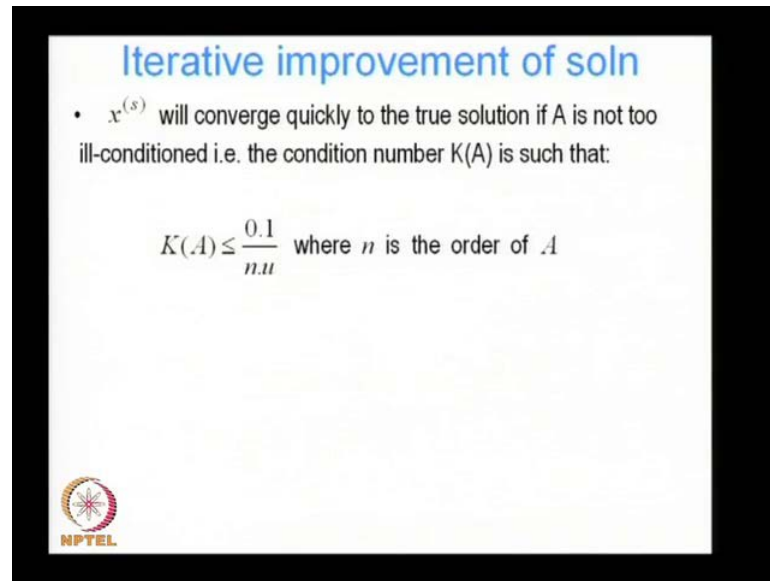
- Only the computation  $r^{(s)} = b - Ax^{(s)}$  requires double precision storage



So, this process can be carried over for a number of iterations we can perform successive iterations in this manner to improve the computed solution, how do we do this the scheme is as follows. So, we said in the first iteration we said  $\bar{x}^{(1)}$  is equal to the computed solution from Gaussian elimination. Then, we compute for subsequent iterations we compute  $x^{(s)}$ ,  $s$  goes from 2, 3 to as many iterations we want to as  $r^{(s)}$  is equal to  $b - Ax^{(s)}$ . So, for the first iteration we compute  $r^{(1)}$  is equal to  $b - Ax^{(1)}$  just as we saw previously and then we compute our correction  $\bar{L}(\bar{U}\delta x^{(s)}) = r^{(s)}$  is equal to  $r^{(s)}$ . We update our solution the corrected solution  $x^{(s+1)}$  is equal to  $x^{(s)} + \delta x^{(s)}$ , so  $s = 1$  for  $x^{(2)}$ , we get  $x^{(1)} + \delta x^{(1)}$ .

Then, go back again and go through this whole thing again until we are satisfied that our solution is sufficiently accurate. How do we know that our solution is sufficiently accurate when my residual is sufficiently small, so in this case the magnitude of the residual gives me a measure of the accuracy of the solution as we discussed in the beginning. This is only going to be true when the matrix  $A$  is not extremely ill conditioned and we note again that only the computation of the residual  $r^{(s)} = b - Ax^{(s)}$  only that requires double precision storage. So, there is not too much excessive computational cost due to double precision storage, since only this operation requires double precision storage.


(Refer Slide Time: 12:20)



**Iterative improvement of soln**

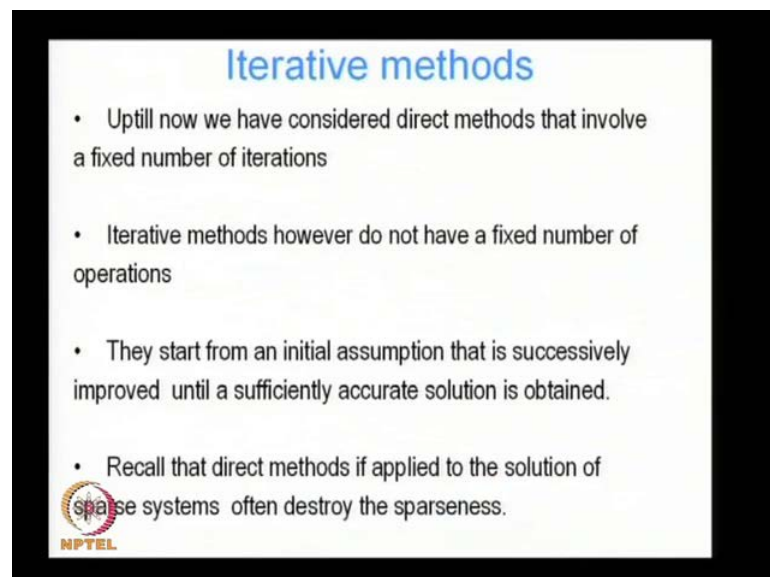
- $x^{(s)}$  will converge quickly to the true solution if  $A$  is not too ill-conditioned i.e. the condition number  $K(A)$  is such that:

$$K(A) \leq \frac{0.1}{n.u} \text{ where } n \text{ is the order of } A$$




It is found that  $x^{(s)}$  will converge quickly to the true solution if  $A$  is not too ill conditioned, what do we mean by too ill conditioned? That is the condition number of  $A$  that which we defined couple of classes back the condition number of  $A$  is lesser than  $0.1$  divided by  $n$  by  $U$ , where  $n$  is the order of the matrix  $A$  it is the number of rows and number of columns in  $A$ .

(Refer Slide Time: 13:21)



**Iterative methods**

- Uptill now we have considered direct methods that involve a fixed number of iterations
- Iterative methods however do not have a fixed number of operations
- They start from an initial assumption that is successively improved until a sufficiently accurate solution is obtained.
- Recall that direct methods if applied to the solution of sparse systems often destroy the sparseness.



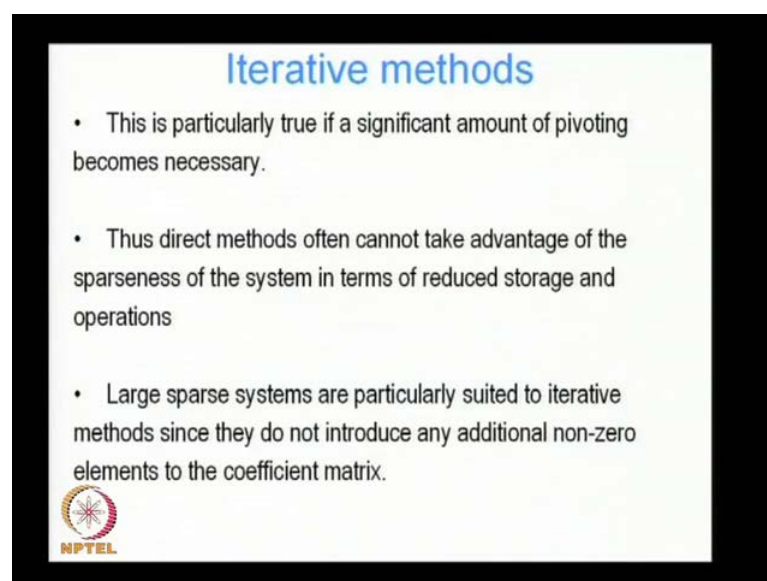
Here,  $U$  is my machine precision, if my condition number is less than this value, then we can say that the matrix is not too ill conditioned and by doing this iterative procedure we

can improve the solution obtained by Gauss elimination. We can improve the solution, why do we need to improve the solution, because the solution has got round of errors, so this is how we can try to reduce the round of errors.

Next, we want to talk about iterative methods in proper the last the iterative scheme, which we talked about just previously was improving the solution direct improving the solution from a direct method. That is Gauss elimination iteratively, but now we are going to talk about iterative methods proper which are which do not involve any metrics inversions at all. Now, we have considered direct methods that involve a fixed number of iterations, so we perform a fixed number, I am sorry this is this should not be iterations this is operations. Now, we have considered direct methods that involve a fixed number of operations iterative methods, however do not have a fixed number of operations.


They start from an initial assumption that is successively improved until a sufficiently accurate solution is obtained. So, we start with a guess to the solution and then we keep on iterating and then we can show that if our if our iterative scheme is well posed, then eventually we get a solution which is almost very close to the true solution. Why do we need iterative methods because recall that direct methods if applied to the solution of sparse systems often destroy the sparseness. We have seen particularly when there is pivoting required then direct methods are going to destroy the sparseness of the system.

(Refer Slide Time: 15:02)



**Iterative methods**

- This is particularly true if a significant amount of pivoting becomes necessary.
- Thus direct methods often cannot take advantage of the sparseness of the system in terms of reduced storage and operations
- Large sparse systems are particularly suited to iterative methods since they do not introduce any additional non-zero elements to the coefficient matrix.

 NPTEL

So, thus direct methods cannot take advantage of the sparseness of the system in terms of reduced storage and operation. So, if we have a sparse matrix, you would like to limit the number you would like to take advantage of this sparseness of the system by reducing the number of operations, but direct methods because of pivoting because of pivoting the sparseness gets lost. We cannot take advantage of the sparseness of the system that is why iterative methods are useful.

Large sparse systems are particularly suited to iterative methods, since they do not introduce any additional non 0 elements to the coefficient matrix, why is that because in iterative methods the coefficient matrix as we will see does not change. It remains constant throughout the iterations for all the iterations the coefficient matrix remains the same. So, the coefficient matrix does not change, so there is no increase in the sparseness of the coefficient matrix, so we can take advantage of the sparse structure of the coefficient matrix.


(Refer Slide Time: 16:11)

### Iterative methods

- Considering the system  $Ax = b$  and supposing all diagonal terms of the coefficient matrix are non-zero i.e.  $a_{ii} \neq 0 \forall i$  we can write the solution of the system as:

$$x_i = -\frac{\sum_{\substack{j=1 \\ j \neq i}}^n a_{ij}x_j + b_i}{a_{ii}} \quad i = 1, \dots, n$$

- One of the most straightforward and widely used iterative methods, Jacobi's method, uses the above expression to compute a sequence of operations  $x^{(1)}, x^{(2)}, \dots$



Let us consider the system  $Ax = b$  and for the time being, let us suppose that all diagonal terms of the coefficient all elements on the principal diagonal are non 0 that is  $a_{ii} \neq 0$  for all  $i$ . In that case, we can write the solution of the system as  $x_i$  is equal to any let us constitute any equation. In this system of  $n$  equations and that equation we can write  $x_i$  as minus sigma  $a_{ij}x_j$   $j$  is equal to 1 to  $n$  except  $j$  is not equal

to  $a_{ii}x_i + b_i$  divided by  $a_{ii}$ . Basically, this is the same equation this is exactly if we bring the  $a_{ii}$  to the left hand side, we will get  $a_{ii}x_i$ .

If you bring this the rest of the terms to the right hand side, we will just get  $a_{ij}x_j$  is equal to  $b_i$ . So, that is my usual any row in that equation in that system of equations except that I have moved one of the terms  $x_i$  to the left hand side and divided the right hand side by its coefficient. So, this is identical to my usual any equation in that system of equations corresponds to that. One of the most straightforward and widely used iterative methods Jacobi's method uses the above expression to find the sequence of operations  $x_1, x_2$  and so on and so forth. So, what does it do well what it does is that it starts with a particular assumption it starts with a particular assumption and then it keeps on improving the solution.


(Refer Slide Time: 18:04)

**Iterative methods**

- Considering the system  $Ax = b$  and supposing all diagonal terms of the coefficient matrix are non-zero i.e.  $a_{ii} \neq 0 \forall i$  we can write the solution of the system as:

$$x_i = -\frac{\sum_{j=1, j \neq i}^n a_{ij}x_j + b_i}{a_{ii}} \quad i = 1, \dots, n$$

- One of the most straightforward and widely used iterative methods, Jacobi's method, uses the above expression to compute a sequence of operations  $x^{(1)}, x^{(2)}, \dots$

 NPTEL

So, it starts with an assumption for the initial values of  $x_1, x_2, x_3$  up to  $x_n$ , it starts with an assumption of the initial value which we call which we denote with a super script 0. We start with that assumption and then for each variable  $x_1, x_2$  and so on and so forth, we solve these equations, we solve these equations to obtain updated values of these quantities.

So, basically look at compare this equation to the previous equation, all we have done is that to the left hand side we have added a subscript of  $k + 1$  and to the right hand side.

We have put, sorry we have added a superscript of k plus 1 and to the right hand side, we have added a superscript of k.

(Refer Slide Time: 19:00)

The slide is titled "Iterative methods" in blue text. It displays the following equation for the iteration:

$$x_i^{(k+1)} = \frac{\sum_{j=1}^n a_{ij} x_j^{(k)} + b_i}{a_{ii}}$$

Below the equation, there are two bullet points:

- The starting approximation  $\mathbf{x}^{(0)}$  can be chosen as any value – if chosen close to the solution it leads to faster convergence i.e. it takes a lesser number of iterations to arrive at the solution
- In the absence of any knowledge of the true solution we can start with  $\mathbf{x}^{(0)} = \mathbf{0}$

At the bottom left of the slide, there is a circular logo with the word "star" inside, and the text "NPTEL" below it.

So, all we have we are saying that if we know the solution from the previous iteration which I am denoting with a super script k and I put them on the left on the right hand side. Then, I can get the new as new values of iterate from the left hand side because I am writing this equation by isolating it. I am writing re-writing each equation by isolating one of the independent variables and putting it on the left hand side. So, the one on the left hand side that is going to give me the new value of the iterate, the one on the on the left hand side is going to give me the new value of the iterate in terms of the old values of the iterate, which appear on the right hand side.

So, we start with an initial guess to the solution we start with an initial guess to all the components of x and denote it by x 0. We put it on the right hand side and then we find out improved guesses to each of the components of x on the left hand side and denote it by x 1 and so on and so forth. We go on doing it over and over again, so once we know x 1, we put that on the right hand side and we compute x 2 and again we do the same thing for x 2 and so on and so forth. So, how long will it take for me to converge the accurate solution, it depends on what starting guess I choose for x 0, if I choose x 0 to be close to my true solution, I will converge in relatively fewer number of iterations.




(Refer Slide Time: 20:42)

**Iterative methods**

$$x_i^{(k+1)} = -\frac{\sum_{j=1, j \neq i}^n a_{ij}x_j^{(k)} + b_i}{a_{ii}}$$

- The starting approximation  $\mathbf{x}^{(0)}$  can be chosen as any value – if chosen close to the solution it leads to faster convergence i.e. it takes a lesser number of iterations to arrive at the solution
- In the absence of any knowledge of the true solution we can start with  $\mathbf{x}^{(0)} = 0$



However, in most linear System solutions for linear systems, we do not know what is a good guess what a good starting guess is. So, a safe starting guess is to assume that  $\mathbf{x} = 0$  equal to 0, so we start with assumptions that all the components of my solution are 0, put that on the right hand side of this equation find out my improved guesses for each of the variables and do this over and over again.

(Refer Slide Time: 21:11)


**Iterative methods**

- In Jacobi's method, the updated solution is not used until after a complete iteration e.g. Jacobi's iteration goes as follows:

$$x_1^{(1)} = -\frac{(a_{12}x_2^{(0)} + a_{13}x_3^{(0)} - \dots - a_{1n}x_n^{(0)}) + b_1}{a_{11}}$$

However, even though  $x_1^{(1)}$  is known by the time  $x_2^{(1)}$  is evaluated, it is not used:

$$x_2^{(1)} = -\frac{(a_{21}x_1^{(0)} - a_{23}x_3^{(0)} - \dots - a_{2n}x_n^{(0)}) + b_2}{a_{22}}$$
$$x_2^{(1)} \neq -\frac{(a_{21}x_1^{(1)} - a_{23}x_3^{(0)} - \dots - a_{2n}x_n^{(0)}) + b_2}{a_{22}}$$



So, in Jacobi's method, the updated solution is not used until after a complete iteration, for example Jacobi's iteration goes as follows for instance we start with all the 0 values

we know  $x_1^{(0)}$ ,  $x_2^{(0)}$  so on and so forth  $x_3^{(0)}$  up to  $x_n^{(0)}$ . So, we compute  $x_1^{(1)}$  from this equation, but now we know  $x_1^{(1)}$ , so when the time comes to compute  $x_2^{(1)}$ , we could have put in this here we could have put  $x_1^{(1)}$ , but we actually put the initial the values from the previous iteration. So, we persist with the value of  $x_1^{(0)}$  rather than using  $x_2^{(1)}$  is equal to minus  $a_{21}$ ,  $x_1^{(1)}$ , the value that is known, but  $x_3$  we do not know yet the updated value for iteration.

Once we use the  $x_3^{(0)}$  value, so we could have used this scheme the bottom scheme. We could have used this scheme instead of this scheme, but Jacobi's iterations says that we use the same the same values for all the variables dividing the iterations. The values from the previous iteration we have used the values from the previous iteration to get the updated values of the variables for the new iterations, we do not use any of the updated values until the whole iteration is complete.

(Refer Slide Time: 22:48)


### Iterative methods

- On the other hand in Gauss-Seidel's method, the updated values of the variable are used as soon as they are computed. The advantage is that at any point of time only one value of  $x_1, x_2, \dots, x_n$  need to be stored.

$$\text{Hence, } x_i^{(k+1)} = \frac{-\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)} + b_i}{a_{ii}}, \quad i = 1, \dots, n$$

Thus now the algorithm is:

$$x_1^{(1)} = \frac{-(a_{21}x_2^{(0)} - a_{13}x_3^{(0)} - \dots - a_{2n}x_n^{(0)}) + b_1}{a_{11}}$$

$$x_2^{(1)} = \frac{-a_{21}x_1^{(1)} - (a_{23}x_3^{(0)} \dots + a_{2n}x_n^{(0)}) + b_2}{a_{22}}$$


On the other hand, there is something known as Gauss Seidel's method in the Gauss Seidel's method the updated variables are used as soon as they are found. So, once we know  $x_1^{(1)}$  we are going to use  $x_1^{(1)}$  in the equation for  $x_2^{(1)}$ , so if we go back to the previous slide for Gauss Seidel says we are going to use this while Jacobi says that we are going to use this. The advantage is that any at any point of time only one in Gauss Seidel at any point of time we are storing only one value of these variables  $x_1$  through  $x_n$ . So, either we have  $x_1^{(0)}$  or we have  $x_1^{(1)}$ , so we can we can overwrite that value of  $x$

$x_{1,0}$  by the value  $x_{1,1}$  as soon as we compute it because once we have computed  $x_{1,1}$  we are not going to use  $x_{1,0}$  anymore in our subsequent computations.

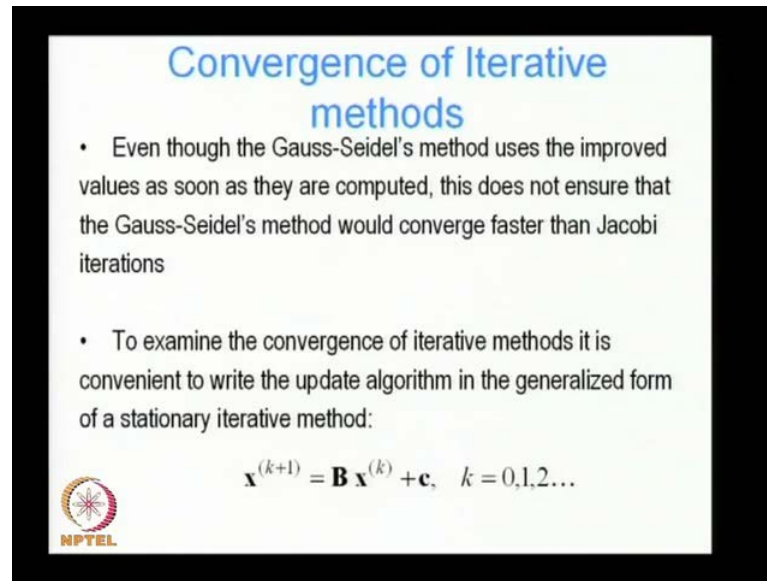
We are always going to use  $x_{1,1}$ , so we can overwrite  $x_{1,0}$  by  $x_{1,1}$ , so this is an advantage of Gauss Seidel. We do not need additional storage, but in case of Jacobi we have to carry along two copies because unless the whole iteration is over we cannot throw out the old values. So, we have to carry along two copies of each variable, so that is the advantage of Gauss Seidel.

So, for this Gauss Seidel iteration the algorithm is as follows  $x_{i,k+1}$  is equal to  $\frac{1}{a_{ij}} (b_j - \sum_{j=1}^{i-1} a_{ij} x_{j,k+1} - \sum_{j=i+1}^n a_{ij} x_{j,k})$ . So, for this part this part involves for instance we have this is I am sorry, this is  $i$ , so  $x_{i,k+1}$ , suppose we are interested in finding the value of  $x_{i,k+1}$  and suppose  $i$  is equal to 5. Suppose, my hat and I am interested in finding the value of  $k+1$   $i$ th iteration, so I am interested in finding  $x_{5,k+1}$ , but by the time I am trying to find  $x_{5,k+1}$  I have already found  $x_{1,k+1}$   $x_{2,k+1}$   $x_{3,k+1}$  and  $x_{4,k+1}$ .

I have no idea what is  $x_{6,k+1}$   $x_{7,k+1}$ ,  $x_{8,k+1}$  up to  $x_{n,k+1}$ , so for up to  $x_{4,k+1}$ , I can use the updated value the  $k+1$  value, but be for  $x_6$  and so forth I have to use the old value the  $k$  value. So, this part I use the updated values this part the terms which are below the principal diagonal I use at that row below. I do not mean below the principal diagonal term of that row, we use the old value the terms which are above the principal, sorry below the principal diagonal, we use the new values above the principal diagonal we use the updated values.

Thus, now the new algorithm is  $x_{1,1}$  for  $x_1$  we have we have to use the old iteration values, but for  $x_2$  for one  $x_{1,1}$  we are going we are using  $x_{1,1}$ . So, for  $x_1$ , we are using the updated value, but for  $x_3$  through  $X_n$  we are still using the old values.


(Refer Slide Time: 26:32)



**Convergence of Iterative methods**

- Even though the Gauss-Seidel's method uses the improved values as soon as they are computed, this does not ensure that the Gauss-Seidel's method would converge faster than Jacobi iterations
- To examine the convergence of iterative methods it is convenient to write the update algorithm in the generalized form of a stationary iterative method:

$$\mathbf{x}^{(k+1)} = \mathbf{B} \mathbf{x}^{(k)} + \mathbf{c}, \quad k = 0, 1, 2, \dots$$

 NPTEL

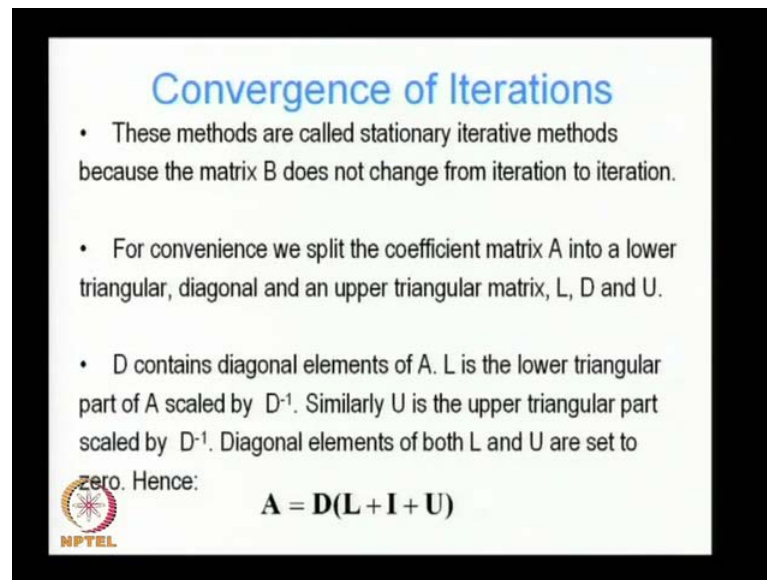
So, even though the Gauss Seidel algorithm uses improved values as soon as they are computed, this does not ensure that Gauss Seidel's method would converge faster than Jacobi iterations. It may turn out that for particular problem, the Gauss Seidel may actually need more iterations than Jacobi. Only advantage of Gauss Seidel obviously is because it is stored computational storage because in Gauss Seidel. We only need to carry along one copy of my independent variables while in Jacobi; I have to carry along two copies of my independent variables at any iteration so that is the major advantage.

So, any iterative method n as you can see if it is not true that if you perform any iteration we are going to converge to the true solution. Eventually, that is not true, there are only certain conditions each iterative algorithm must satisfy certain conditions in order to converge. So, after n iterations I know that my solution is going to approach the true solution only if only if my algorithm satisfies certain conditions and what are those conditions.

Those are the convergence requirements of iterative methods and in order to obtain those convergence requirements. It is convenient to write update algorithm in the generalised form of a stationery iterative method. So, this is a generalised form of a stationery iterative method which says that my improved estimate to x. I obtained at the k plus 1 th iteration is equal to sum operate at b operating my old estimate x k plus some constant vector c and k goes from 0 to 2. So, in order to obtain convergence characteristics, we

write these iterative methods in this general form all of these iterative methods Jacobi Gauss Seidel. They fit into this general form, we shall see we shall show how they exactly fit into it, but they all satisfy this form.

(Refer Slide Time: 28:48)



**Convergence of Iterations**

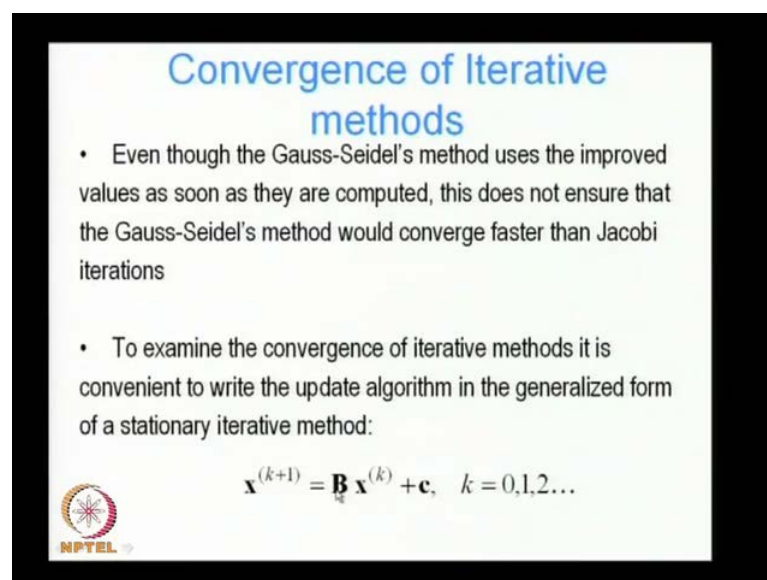
- These methods are called stationary iterative methods because the matrix B does not change from iteration to iteration.
- For convenience we split the coefficient matrix A into a lower triangular, diagonal and an upper triangular matrix, L, D and U.
- D contains diagonal elements of A. L is the lower triangular part of A scaled by  $D^{-1}$ . Similarly U is the upper triangular part scaled by  $D^{-1}$ . Diagonal elements of both L and U are set to zero. Hence:

$$\mathbf{A} = \mathbf{D}(\mathbf{L} + \mathbf{I} + \mathbf{U})$$

NPTEL

Why are these methods called stationery iterative methods because the matrix B, which we have indicated here does not change from iteration to iteration.

(Refer Slide Time: 28:55)



**Convergence of Iterative methods**

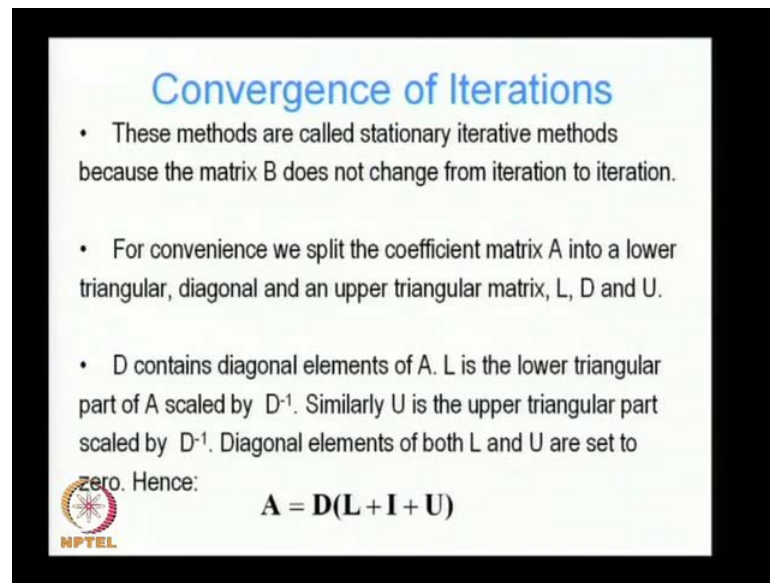
- Even though the Gauss-Seidel's method uses the improved values as soon as they are computed, this does not ensure that the Gauss-Seidel's method would converge faster than Jacobi iterations
- To examine the convergence of iterative methods it is convenient to write the update algorithm in the generalized form of a stationary iterative method:

$$\mathbf{x}^{(k+1)} = \mathbf{B} \mathbf{x}^{(k)} + \mathbf{c}, \quad k = 0, 1, 2, \dots$$

NPTEL

That is why it is called stationery iterative method.


(Refer Slide Time: 29:04)



**Convergence of Iterations**

- These methods are called stationary iterative methods because the matrix B does not change from iteration to iteration.
- For convenience we split the coefficient matrix A into a lower triangular, diagonal and an upper triangular matrix, L, D and U.
- D contains diagonal elements of A. L is the lower triangular part of A scaled by  $D^{-1}$ . Similarly U is the upper triangular part scaled by  $D^{-1}$ . Diagonal elements of both L and U are set to zero. Hence:

$$A = D(L + I + U)$$

 NPTEL

For convenience, we split the coefficient matrix a into a lower triangular a diagonal and an upper triangular matrix L d and U d contains the diagonal elements of my original matrix a L is the lower triangular part of a scaled by d inverse. So, b d 1 by I mean basically b inverse contains the diagonal terms inverse of the diagonal terms on its principal diagonal. So, a scale the lower triangular part of L lower triangular part of a by d inverse to obtain L, similarly to obtain U i scale the upper triangular part by d inverse and I put the diagonal elements of both L and U to 0. So, L contains the lower triangular part of a except for the fact that its diagonal elements are 0 and each term is scaled by the inverse of d, so it is scaled by d inverse.

Similarly, for U in that case I can write a is equal to d L plus i plus U you can see how why this is true because L is actually d inverse of L if L is the true real lower triangular part of A, then L is actually d inverse of L. So, D inverse cancels gives me i, so I have this d L is going to give me the lower triangular part of a with the principal with 0 s on the diagonal D i is going to give me the diagonal terms.


Then, d U is going to give me d U is actually d inverse of U where U is the upper triangular part of A. So, D D inverse of U that is just going to give me the upper triangular part a with that 0 s on the diagonal.



(Refer Slide Time: 31:13)

### Convergence of Iterations

- Then Jacobi's method can be written as:
 
$$\mathbf{x}^{(k+1)} = -(\mathbf{L} + \mathbf{U})\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{b}$$
- Thus B for Jacobi's method is:  $-(\mathbf{L} + \mathbf{U})$ .
- The Gauss-Seidel's method is written as:
 
$$\mathbf{x}^{(k+1)} = -\mathbf{L}\mathbf{x}^{(k+1)} - \mathbf{U}\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{b}$$
- Hence B for Gauss-Seidel's methods is:  $-(\mathbf{I} + \mathbf{L})^{-1}\mathbf{U}$
- Assume B has eigen values  $\lambda_1, \lambda_2, \lambda_3 \dots \lambda_n$  and linearly independent eigen vectors  $u_1, u_2, \dots, u_n$  corresponding to those eigen values.




So, we if we split it like that then we can write Jacobi's method as  $\mathbf{x}^{(k+1)} = -(\mathbf{L} + \mathbf{U})\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{b}$ , if we go back to the statement of Jacobi's algorithm, you will see that this is true.

(Refer Slide Time: 31:37)

### Iterative methods

$$x_i^{(k+1)} = -\frac{\sum_{j=1, j \neq i}^n a_{ij}x_j^{(k)} + b_i}{a_{ii}}$$

- The starting approximation  $\mathbf{x}^{(0)}$  can be chosen as any value – if chosen close to the solution it leads to faster convergence i.e. it takes a lesser number of iterations to arrive at the solution
- In the absence of any knowledge of the true solution we can start with  $\mathbf{x}^{(0)} = \mathbf{0}$



This is Jacobi's algorithm, so this is  $x_i^{(k+1)}$ , this is  $\sum_{j=1, j \neq i}^n a_{ij}x_j^{(k)} + b_i$ , so it is basically  $\mathbf{L} + \mathbf{U}$  times  $\mathbf{x}^{(k)}$  plus  $\mathbf{D}^{-1}\mathbf{b}$ . The diagonal term is not there, so  $\mathbf{D}^{-1}\mathbf{b}$  comes from this  $b_i$  by  $a_{ii}$  is the diagonal term, let me go back and look at it again.

(Refer Slide Time: 32:20)


### Iterative methods

- In Jacobi's method, the updated solution is not used until after a complete iteration e.g. Jacobi's iteration goes as follows:  

$$x_1^{(1)} = -\frac{(a_{12}x_2^{(0)} - a_{13}x_3^{(0)} - \dots - a_{1n}x_n^{(0)}) + b_1}{a_{11}}$$

However, even though  $x_1^{(1)}$  is known by the time  $x_2^{(1)}$  is evaluated, it is not used:

$$x_2^{(1)} = -\frac{(a_{21}x_1^{(0)} - a_{23}x_3^{(0)} - \dots - a_{2n}x_n^{(0)}) + b_2}{a_{22}}$$

$$x_2^{(1)} \neq -\frac{(a_{21}x_1^{(1)} - a_{23}x_3^{(0)} - \dots - a_{2n}x_n^{(0)}) + b_2}{a_{22}}$$



So, this D inverse B is coming from this b i by a i i.

(Refer Slide Time: 32:25)

### Iterative methods

$$x_i^{(k+1)} = -\frac{\sum_{j=1, j \neq i}^n a_{ij}x_j^{(k)} + b_i}{a_{ii}}$$

- The starting approximation  $\mathbf{x}^{(0)}$  can be chosen as any value – if chosen close to the solution it leads to faster convergence i.e. it takes a lesser number of iterations to arrive at the solution
- In the absence of any knowledge of the true solution we can start with  $\mathbf{x}^{(0)} = 0$



This sigma a i j x j divided by a i i is nothing but d inverse times L plus U because a i j x j is L plus a i j a i j x j k is nothing but L plus u.


(Refer Slide Time: 32:45)

### Convergence of Iterations

- Then Jacobi's method can be written as:  

$$\mathbf{x}^{(k+1)} = -(\mathbf{L} + \mathbf{U})\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{b}$$
- Thus B for Jacobi's method is:  $-(\mathbf{L} + \mathbf{U})$ .
- The Gauss-Seidel's method is written as:  

$$\mathbf{x}^{(k+1)} = -\mathbf{L}\mathbf{x}^{(k+1)} - \mathbf{U}\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{b}$$
- Hence B for Gauss-Seidel's methods is:  $-(\mathbf{I} + \mathbf{L})^{-1}\mathbf{U}$
- Assume B has eigen values  $\lambda_1, \lambda_2, \lambda_3 \dots \lambda_n$  and linearly independent eigen vectors  $u_1, u_2, \dots, u_n$  corresponding to those eigen values.



So, I can write the Jacobi's equation in this form  $x_{k+1}$  is equal to minus L plus U  $x_k$  plus D inverse of b. Comparing it with our standard form for stationary iterative methods, it is clear that b is equal to minus L plus U for the Jacobi's method. The Gauss-Seidel method can be written as  $x_{k+1}$  is equal to minus L  $x_{k+1}$  all the terms which are below the principal diagonals go gets scaled by L the terms above the principal diagonal gets scaled by U. These involve the values from the previous iteration, well these have already been found from the new iterations, so that is  $x_{k+1}$  of course, there is the term D inverse of B.

(Refer Slide Time: 33:44)


### Iterative methods

- On the other hand in Gauss-Seidel's method, the updated values of the variable are used as soon as they are computed. The advantage is that at any point of time only one value of  $x_1, x_2, \dots, x_n$  need to be stored.

$$\text{Hence, } x_i^{(k+1)} = \frac{-\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)} + b_i}{a_{ii}}, \quad i = 1, \dots, n$$

Thus now the algorithm is:

$$x_1^{(1)} = \frac{-(a_{21}x_2^{(0)} - a_{13}x_3^{(0)} - \dots - a_{2n}x_n^{(0)}) + b_1}{a_{11}}$$

$$x_2^{(1)} = \frac{-a_{21}x_1^{(1)} - (a_{23}x_3^{(0)} \dots + a_{2n}x_n^{(0)}) + b_2}{a_{22}}$$


So, again if you go back and look at our expression for Gauss Seidel, we will see that this is how it looks like this is  $Lx^{(k+1)} + Dx^{(k+1)} - Ux^{(k)} + b$  and everything I am dividing by a  $D^{-1}$ , so there is a  $D^{-1}$  inverse, but this is already built into the definition of  $L$ .

(Refer Slide Time: 34:10)

**Convergence of Iterations**

- Then Jacobi's method can be written as:  

$$x^{(k+1)} = -(L+U)x^{(k)} + D^{-1}b$$
- Thus  $B$  for Jacobi's method is:  $-(L+U)$ .
- The Gauss-Seidel's method is written as:  

$$x^{(k+1)} = -Lx^{(k+1)} - Ux^{(k)} + D^{-1}b$$
- Hence  $B$  for Gauss-Seidel's methods is:  $-(I+L)^{-1}U$
- Assume  $B$  has eigen values  $\lambda_1, \lambda_2, \lambda_3 \dots \lambda_n$  and linearly independent eigen vectors  $u_1, u_2, \dots, u_n$  corresponding to those eigen values.

MPTEL

So, that does not show up, so minus  $Lx^{(k+1)} + Dx^{(k+1)} - Ux^{(k)} + b$  for Gauss Seidel's method is minus  $D^{-1}(L+U)x^{(k+1)} + D^{-1}b$  why is that  $x^{(k)}$  what is the general form of the stationary iterative method it is  $x^{(k+1)} = Bx^{(k)} + c$ . So, we bring this all the terms involving  $x^{(k+1)}$  on the left hand side, so we get  $(I+L)x^{(k+1)} = -Ux^{(k)} + D^{-1}b$  we take the inverse of that  $(I+L)^{-1}$  inverse, we get  $x^{(k+1)} = -(I+L)^{-1}Ux^{(k)} + (I+L)^{-1}D^{-1}b$ . So, that  $x^{(k+1)}$  is equal to this times  $x^{(k)}$  plus this, so  $B$  for Gauss Seidel's method is of this form. Now, let us assume that our matrix  $B$  which is this matrix for Jacobi. This is the matrix for Gauss Seidel has Eigen values  $\lambda_1, \lambda_2, \lambda_3$  to  $\lambda_n$  and it has got linearly independent Eigen vectors  $U_1, U_2, U_3$  to  $U_n$  corresponding to those Eigen values.

Since, these Eigen vectors are independent  $U_1, U_2, U_n$  form a basis for the space of solutions  $x$  what do we mean by basis because since these are independent vectors any vector of dimension  $n$  can be written as a linear combination of these  $U_1$  through  $U_n$ . So, these are the basis these form a basis for that space of solutions, thus the initial vector  $x_0$  minus  $x$  which is a vector of dimension  $n$  can be written as some constant  $\alpha_1$  times  $U_1$  plus  $\alpha_2 U_2$  though  $\alpha_n U_n$ . So,  $x_0 - x$  where  $x_0$  is my initial

assumption  $\mathbf{x}$  is the true solution my initial error my initial error I can write in terms of my as a linear combination of the Eigen vectors of  $\mathbf{B}$ .

(Refer Slide Time: 35:25)


### Convergence of Iterations

- Then  $u_1, u_2, \dots, u_n$  form a basis for the space of solutions  $\mathbf{x}$
- Thus the initial vector  $(\mathbf{x}^{(0)} - \mathbf{x})$  can be written as:
 
$$\mathbf{x}^{(0)} - \mathbf{x} = \alpha_1 \mathbf{u}_1 + \alpha_2 \mathbf{u}_2 + \alpha_3 \mathbf{u}_3 + \dots + \alpha_n \mathbf{u}_n \quad (*)$$
 However we can write :
 
$$\mathbf{x}^{(k+1)} - \mathbf{x} = \mathbf{B}\mathbf{x}^{(k)} + \mathbf{c} - \mathbf{x}$$
 But the true solution satisfies  $\mathbf{x} = \mathbf{B}\mathbf{x} + \mathbf{c}$ 

$$\therefore \mathbf{x}^{(k+1)} - \mathbf{x} = \mathbf{B}(\mathbf{x}^{(k)} - \mathbf{x}) = \mathbf{B}(\mathbf{B}\mathbf{x}^{(k-1)} + \mathbf{c} - \mathbf{x})$$

$$= \mathbf{B}^2 \mathbf{x}^{(k-1)} - \mathbf{B}^2 \mathbf{x} = \mathbf{B}^2 (\mathbf{x}^{(k-1)} - \mathbf{x})$$

$$= \dots = \mathbf{B}^{k+1} (\mathbf{x}^{(0)} - \mathbf{x})$$



We can write  $\mathbf{x}^{(k+1)} - \mathbf{x}$  is equal to  $\mathbf{B}(\mathbf{x}^{(k)} - \mathbf{x})$  because this is the form of my stationary iterative method except I have that I have subtracted  $\mathbf{x}$  from both sides. So, stationary iterative method is  $\mathbf{x}^{(k+1)} = \mathbf{B}\mathbf{x}^{(k)} + \mathbf{c}$  I subtract  $\mathbf{x}$  from both sides. Let us recall that the true solution satisfies  $\mathbf{x}$  is equal to  $\mathbf{B}\mathbf{x} + \mathbf{c}$  because that satisfies that is my true solution. Therefore, we can write  $\mathbf{x}^{(k)}$  by plus 1 minus  $\mathbf{x}$  is equal to  $\mathbf{B}(\mathbf{x}^{(k-1)} - \mathbf{x})$ .

So, we replace  $\mathbf{x}^{(k+1)}$  by  $\mathbf{B}\mathbf{x}^{(k)} + \mathbf{c}$  and then we subtract that, so  $\mathbf{B}\mathbf{x}^{(k)} - \mathbf{x}$  and this I can write as equal to  $\mathbf{B}(\mathbf{x}^{(k-1)} - \mathbf{x})$  I can again write as  $\mathbf{B}(\mathbf{B}\mathbf{x}^{(k-2)} + \mathbf{c} - \mathbf{x})$ . So,  $\mathbf{x}^{(k)}$ , I am replacing by  $\mathbf{B}\mathbf{x}^{(k-1)} + \mathbf{c} - \mathbf{x}$ , so I eventually get  $\mathbf{B}^2(\mathbf{x}^{(k-2)} - \mathbf{x})$  and again if I replace  $\mathbf{x}^{(k-1)}$  by  $\mathbf{B}\mathbf{x}^{(k-2)} + \mathbf{c}$ , I can do this operation again. Eventually, I am going to get  $\mathbf{B}^{k+1}(\mathbf{x}^{(0)} - \mathbf{x})$  by performing this repeatedly, I can going to get every time I perform I reduce this  $k$  by 1, I reduce this  $k$  by 1, but I add a power to this  $\mathbf{B}$ , eventually I am going to get  $\mathbf{B}^{k+1}(\mathbf{x}^{(0)} - \mathbf{x})$ .

(Refer Slide Time: 38:29)

**Convergence of Iterations**

- Hence using (\*):
$$\mathbf{x}^{(k)} - \mathbf{x} = \mathbf{B}^k (\mathbf{x}^{(0)} - \mathbf{x}) = \mathbf{B}^k (\alpha_1 \mathbf{u}_1 + \alpha_2 \mathbf{u}_2 + \dots + \alpha_n \mathbf{u}_n)$$
$$= \alpha_1 \lambda_1^k \mathbf{u}_1 + \alpha_2 \lambda_2^k \mathbf{u}_2 + \alpha_3 \lambda_3^k \mathbf{u}_3 + \dots + \alpha_n \lambda_n^k \mathbf{u}_n$$
- From the above equation it is clear that if  $|\lambda_i| < 1, i = 1, 2, \dots, n$ :
$$\mathbf{x}^{(k)} - \mathbf{x} \rightarrow 0 \text{ as } k \text{ increases}$$
- Thus the iterations will converge from an arbitrary starting approximation  $\mathbf{x}^{(0)}$  if and only if  $|\lambda_i| < 1, i = 1, 2, \dots, n$
- This conclusion holds even if the eigen vectors  $\mathbf{u}$  are not linearly independent

NPTEL

Using this equation here, so  $\mathbf{x}^{(0)} - \mathbf{x}$ , we know we can write like this, so substituting this value here we get  $\mathbf{B}^k$  operating on  $\alpha_1 \mathbf{u}_1 + \alpha_2 \mathbf{u}_2 + \dots + \alpha_n \mathbf{u}_n$ . Remember that  $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n$  are the eigen vectors of  $\mathbf{B}$ , so  $\mathbf{B}^k$  operating on  $\mathbf{u}_1$  is going to give me  $\lambda_1^k \mathbf{u}_1$ , what is it going to give me  $\mathbf{B}^k$  operating on  $\mathbf{u}_1$  is going to give me the Eigen.

Recall  $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n$  are the Eigen vectors of  $\mathbf{B}$ , so  $\mathbf{B}$  operating on  $\mathbf{u}_1$  is going to give me the corresponding Eigen value times  $\mathbf{u}_1$  corresponding Eigen value. If I denote by  $\lambda_1$ , so  $\mathbf{B}$  operating on  $\mathbf{u}_1$  is going to give me  $\lambda_1 \mathbf{u}_1$ , similarly  $\mathbf{B}^k$  operating on  $\mathbf{u}_1$  is going to give me  $\lambda_1^k \mathbf{u}_1$ . Similarly,  $\mathbf{B}^k$  operating on  $\mathbf{u}_2$  is going to give me  $\lambda_2^k \mathbf{u}_2$  and so on and so forth. So, eventually we can write  $\mathbf{x}^{(k)} - \mathbf{x}$  is equal to  $\mathbf{B}^k (\mathbf{x}^{(0)} - \mathbf{x})$  is actually  $\mathbf{B}^k$  operating on this and  $\mathbf{B}^k$  operating on  $\mathbf{u}_1$  is equal to  $\lambda_1^k \mathbf{u}_1$ ,  $\mathbf{B}^k$  operating on  $\mathbf{u}_2$  is  $\lambda_2^k \mathbf{u}_2$  and so on and so forth.

From the above equation, it is clear that if we can ensure that each of my Eigen values have modulus less than one in that case this term is going to go to 0 as  $k$  increases. So, each of these  $\lambda$ s are less than 1 that  $k$  increases, since this is  $k$  power  $\lambda_1$  to the power  $k$ ,  $\lambda_2$  to the power  $k$  as  $k$  increases. Then, this term the right hand side is going to go to 0 it is going to become smaller and smaller and smaller as my  $k$  increases.



So, we can be assured that if my lambda i's are less than 1, then  $x^{(k)} - x$  is going to go to 0 as k increases.

What does this mean that means as I increase the number of iterations my solution is going to become closer and closer to my true solution provided my b matrix has lambda i which are all less than 1. Thus, the iterations will converge from an arbitrary starting approximation arbitrary starting approximation right my starting approximation can be arbitrary can be any value may be very different from my true solution maybe of the orders of magnitude different from my true solution. Even in that case, I am assured that I am going to get to my true solution after certain number of iterations provided my B matrix has got a spectral radius which is less than 1.


What do I mean by spectral radius, it means that my largest Eigen value must be less than 1 my largest if my largest Eigen value is less than 1 all my other Eigen values are by definition less than 1. So, in that case I am going to get convergence to the true solution. Now, we have obtained this this condition under the assumption that b has got linearly independent Eigen vectors. However, it can be shown that even if the Eigen vectors of b are not linearly independent, this condition still holds, what is that condition? The condition is that if the largest Eigen value of B is less than 1, then my iterative method is bound to converge to the true solution.

(Refer Slide Time: 42:51)

### Convergence of Iterations

- Thus we have : "A necessary and sufficient condition for a stationary iterative method  $\mathbf{x}^{(k+1)} = \mathbf{B}\mathbf{x}^{(k)} + \mathbf{c}$  to converge is that  $\rho(\mathbf{B}) = \max_{1 \leq i \leq n} |\lambda_i(\mathbf{B})| < 1$  where  $\rho(\mathbf{B})$  is the spectral radius of B"
- From  $\mathbf{x}^{(0)} - \mathbf{x} = \alpha_1 u_1 + \alpha_2 u_2 + \alpha_3 u_3 + \dots + \alpha_n u_n$   
 $\mathbf{x}^{(k)} - \mathbf{x} = \alpha_1 \lambda_1^k u_1 + \alpha_2 \lambda_2^k u_2 + \alpha_3 \lambda_3^k u_3 + \dots + \alpha_n \lambda_n^k u_n$

it is clear that to reduce the amplitude of the error component  $\alpha_j u_j$  in  $\mathbf{x}^{(0)} - \mathbf{x}$  by a factor of  $10^{-m}$  we have to make k iterations where k is the smallest number such that:



$$|\lambda_j|^k \leq 10^{-m} \quad \text{or} \quad k \geq \frac{m}{-\log|\lambda_j|}$$

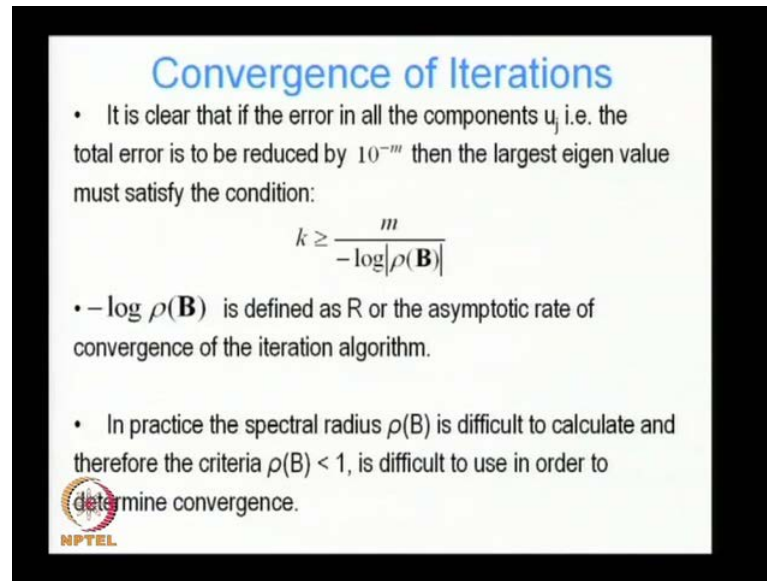
Thus, we have a necessary and sufficient condition for a stationary iterative method  $x_{k+1} = Bx_k + c$  to converge. The spectral radius of  $B$ , which is basically the largest Eigen value  $\max_{1 \leq i \leq n} |\lambda_i|$  for  $i = 1$  through  $n$  is less than one  $\rho(B) < 1$ . The spectral radius of  $B$  from  $x_0 - x^*$  is equal to  $\alpha_1 |U_1|^k + \alpha_2 |U_2|^k + \alpha_3 |U_3|^k + \dots + \alpha_n |U_n|^k$ . We have seen that we have since the  $U_1$  is formed a basis for this space, we can write any vector in terms of this. Similarly,  $x_k - x^*$  can be written as  $\alpha_1 \lambda_1^k U_1 + \alpha_2 \lambda_2^k U_2 + \dots + \alpha_n \lambda_n^k U_n$ . This we have just seen  $x_k - x^*$  can be written like that it is clear that to reduce the amplitude of the error component in error component  $\alpha_j |U_j|^k$ .

These are my error components right because this is my error vector and these are my error components. So, to reduce the error the amplitude of the error component  $\alpha_j |U_j|^k$  and  $x_0 - x^*$  by a factor of  $10^{-m}$ . We have to make  $k$  iterations where  $k$  is the smallest number such that  $|\lambda_j|^k$  is lesser than or equal to  $10^{-m}$ .

Basically, for instance I want to reduce this error component by a factor of  $10^{-m}$   $\alpha_j |U_j|^k$  can that can only happen if  $|\lambda_j|^k$  this term  $|\lambda_j|^k$  to the power  $k$  is lesser than or equal to  $10^{-m}$ . So, this means that  $k$  must be greater than or equal to  $m - \log_{10} |\lambda_j|$ , this is the same thing except that we have taken the log to the base 10 on both sides. So, in that case we get  $k$  greater than or equal to  $m - \log_{10} |\lambda_j|$ .

It is clear that if the error in all the components  $U_j$  that is the total error is to be reduced by  $10^{-m}$ . Then, the largest Eigen value the largest Eigen value must satisfy this equation this is for Eigen value  $\lambda_j$ , but if this is satisfied for the largest Eigen value. Then, we can be sure that the error in all the components has been reduced by a factor of  $10^{-m}$  and this factor  $m - \log_{10} \rho(B)$  is defined as the asymptotic rate of convergence of the iteration algorithm. This is defined as the asymptotic rate of convergence of the iteration algorithm.

(Refer Slide Time: 45:25)



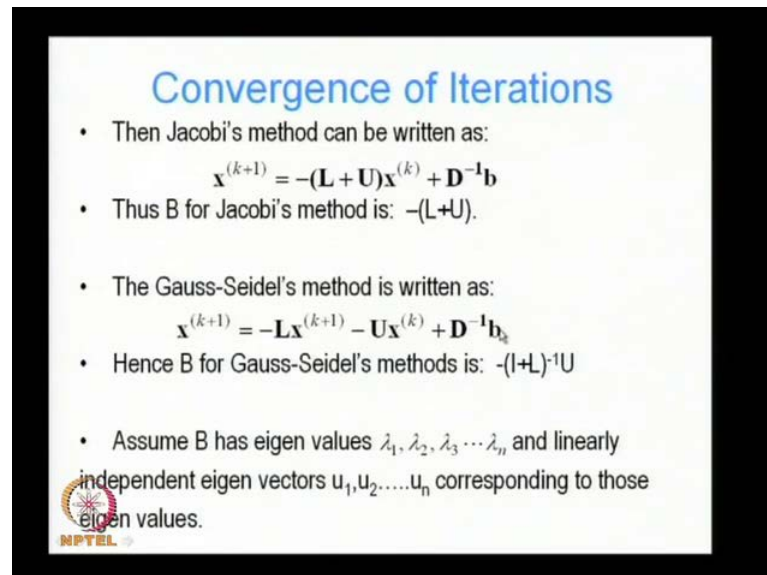
**Convergence of Iterations**

- It is clear that if the error in all the components  $u_j$  i.e. the total error is to be reduced by  $10^{-m}$  then the largest eigen value must satisfy the condition:
$$k \geq \frac{m}{-\log|\rho(\mathbf{B})|}$$
- $-\log \rho(\mathbf{B})$  is defined as R or the asymptotic rate of convergence of the iteration algorithm.
- In practice the spectral radius  $\rho(B)$  is difficult to calculate and therefore the criteria  $\rho(B) < 1$ , is difficult to use in order to determine convergence.

NPTEL

So, what have we seen up to here, we have seen that if a stationary iterative method is going to converge to its largest Eigen value must be less than 1, but a priori given a set of a linear system of equations. Given that, we have no idea what is the spectral radius of  $B$  right first we have to find out what is  $B$ .

(Refer Slide Time: 46:33)



**Convergence of Iterations**

- Then Jacobi's method can be written as:
$$\mathbf{x}^{(k+1)} = -(\mathbf{L} + \mathbf{U})\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{b}$$
- Thus  $B$  for Jacobi's method is:  $-(\mathbf{L} + \mathbf{U})$ .
- The Gauss-Seidel's method is written as:
$$\mathbf{x}^{(k+1)} = -\mathbf{L}\mathbf{x}^{(k+1)} - \mathbf{U}\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{b}$$
- Hence  $B$  for Gauss-Seidel's methods is:  $-(\mathbf{I} + \mathbf{L})^{-1}\mathbf{U}$
- Assume  $B$  has eigen values  $\lambda_1, \lambda_2, \lambda_3 \dots \lambda_n$  and linearly independent eigen vectors  $u_1, u_2, \dots, u_n$  corresponding to those eigen values.

NPTEL

We have to find out  $B$  from this expression if we are doing Jacobi, we can find out  $B$  from this expression. If we are doing Gauss Seidel, then we have to do a spectral decomposition, we have to find the Eigen values of this  $B$  matrix to find out what is the

largest Eigen value. We can probably use Rayleigh's method or we are going to talk about those methods, later on we can use some iterative methods to find out the largest Eigen value, but that is all quite expensive. So, a priori before hand, we have no idea what is the largest Eigen value B and whether that Eigen value is less than 1, which will which will make sure that my iterative method is going to converge. So, we need some other indicator of convergence which is relatively simpler to obtain than actually doing going ahead and doing a spectral decomposition of b.


(Refer Slide Time: 47:27)

### Convergence of Iterations

- It is clear that if the error in all the components  $u_j$  i.e. the total error is to be reduced by  $10^{-m}$  then the largest eigen value must satisfy the condition:

$$k \geq \frac{m}{-\log|\rho(\mathbf{B})|}$$

- $-\log \rho(\mathbf{B})$  is defined as R or the asymptotic rate of convergence of the iteration algorithm.
- In practice the spectral radius  $\rho(B)$  is difficult to calculate and therefore the criteria  $\rho(B) < 1$ , is difficult to use in order to determine convergence.



(Refer Slide Time: 47:43)


### Convergence of Iterations

- Hence we need a convergence criterion that is simpler to use
- Recall that  $\mathbf{x}^{(k)} - \mathbf{x} = \mathbf{B}^k (\mathbf{x}^{(0)} - \mathbf{x})$

Hence,

$$|\mathbf{x}^{(k)} - \mathbf{x}| \leq |\mathbf{B}^k| |\mathbf{x}^{(0)} - \mathbf{x}| \leq |\mathbf{B}|^k |\mathbf{x}^{(0)} - \mathbf{x}|$$

- Thus  $|\mathbf{B}| < 1$  in some consistent norm is a sufficient condition for convergence
- For Jacobi's method:



$$B_j = -(L + U). \text{ Hence, } b_{ij} = -\frac{a_{ij}}{a_{ii}}, i \neq j, b_{ii} = 0$$

In practice, the spectral radius  $\rho(B)$  is difficult to calculate and therefore, the criteria  $\rho(B) < 1$  is difficult to use in order to determine convergence, so we have an alternative criteria. Hence, we need a convergence criteria that is simpler to use, let us recall that we can write  $x^{(k)} - x = B^k(x^{(0)} - x)$  we have seen that before where we have seen that we have seen that here  $x^{(k)} - x = B^k(x^{(0)} - x)$ .

(Refer Slide Time: 48:13)

**Convergence of Iterations**

- Hence we need a convergence criterion that is simpler to use
- Recall that  $x^{(k)} - x = B^k(x^{(0)} - x)$   
Hence,  

$$\|x^{(k)} - x\| \leq \|B^k\| \|x^{(0)} - x\| \leq \|B\|^k \|x^{(0)} - x\|$$
- Thus  $\|B\| < 1$  in some consistent norm is a sufficient condition for convergence
- For Jacobi's method:  

$$B_J = -(L + U). \text{ Hence, } b_{ij} = -\frac{a_{ij}}{a_{ii}}, i \neq j, b_{ii} = 0$$

So, hence we can write  $\|x^{(k)} - x\| \leq \|B\|^k \|x^{(0)} - x\|$  from the definition, which we have discussed in probably the second lecture, we can write this and  $\|B\|^k$  is nothing but  $\|B\|^{k-1} \|B\|$  and so on. So, I can write this is again going to be less than  $\|B\|^k$ , because  $\|B\|^k$  is going to be less than  $\|B\|^{k-1} \|B\|$ , because I can write  $\|B\|^k$  as  $\|B\| \|B\|^{k-1}$ . So,  $\|B\|^k$  is less than or equal to  $\|B\| \|B\|^{k-1}$ . Similarly,  $\|B\|^{k-1}$ , I can split like that and so this thing is going to be less than that.

Thus, if  $\|B\|$  in some norm in some norm is less than 1, then what do we see we see that  $\|x^{(k)} - x\|$  is less than or equal to  $\|B\|^k \|x^{(0)} - x\|$ , so as we increase if  $\|B\|$  is less than 1 as we increase the number of iterations. This term is going to become smaller and smaller this term is going to become smaller and smaller therefore,  $\|x^{(k)} - x\|$  is going to become smaller and smaller also because  $\|B\|^{k-1}$  must be less than this term.


This term is becoming smaller and smaller with the number of iterations, so  $x_k$  is going to approach my true solution  $x$ , thus if we can show that  $\rho(B)$  is less than 1 in some consistent norm this is a sufficient condition for convergence. This is not a necessary and sufficient condition the previous condition  $\rho(B) < 1$  was a necessary and sufficient condition, what is the necessary and sufficient condition. It means that it is only going to converge if  $\rho(B) < 1$ , it is only going to converge if  $\rho(B) < 1$  if and only if  $\rho(B) < 1$ , but this is a sufficient condition for convergence.

This is not a necessary condition, so long as this is less than 1, we are going to converge, but it is going to converge only if  $\rho(B) < 1$ , so that is these are two different things this is sufficient condition that is a necessary and sufficient condition. So, for Jacobi's method we know that  $b_{ij}$  is equal to  $-L_{ij} + U_{ij}$ , therefore,  $b_{ij}$  can write as  $-L_{ij} + U_{ij}$  depending on how have  $L$  depending on the definition of  $L$  and  $U$ , I can write  $b_{ij}$  is equal to  $-L_{ij} + U_{ij}$  because remember what we have defined  $L$  and  $U$  to be  $D^{-1}$  times the actual  $L$  and  $U$  from  $A$ .

(Refer Slide Time: 52:02)

**Convergence of Jacobi's method**

- Recall that  $b_{ii} = 0$  since  $L$  and  $U$  are triangular matrices with zero diagonal terms.
- Hence  $\|B\|_{\infty} = \max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^n \frac{|a_{ij}|}{|a_{ii}|}$
- For a diagonally dominant matrix,  $|a_{ii}| \geq \sum_{j=1, j \neq i}^n |a_{ij}|$ ,  $i = 1, 2, \dots, n$
- Hence if the coefficient matrix is diagonally dominant, we can be assured that  $\|B\|_{\infty} < 1$  and Jacobi's method converges.



So,  $b_{ij}$  is equal to this for  $i \neq j$  and  $b_{ii}$  is equal to 0 because  $a_{ii}$  because  $L$  and  $U$  are the diagonal terms 0. So,  $b_{ii}$  must be equal to 0  $b_{ii}$  is equal to 0 since  $L$  and  $U$  are triangular matrices with 0 diagonal terms. Hence, if I want to take mod of  $b_{ij}$  take the norm of  $B$  in the infinite norm what is this equal to? This is equal to  $\sum_{j=1, j \neq i}^n |a_{ij}|$

sigma mod of a i j a i i i sum it over all the all the columns for each row and then take the maximum over all the rows that is my definition of my infinite norm.

So, I have mod of b infinity is given by that, but for a diagonally dominant matrix which we also defined earlier a diagonally dominant matrix each diagonal term is greater than the sum of the absolute values of all the off diagonal terms. So, this is a definition of a diagonally dominant matrix. So if my b matrix is diagonally dominant if my b if my if not my b matrix sorry i go back if my coefficient matrix a is diagonally dominant.

That means that this is always going to be less than one because a i i is always going to be less than this. So, each diagonal term in absolute value absolute value of each diagonal term is going to be greater than the sum of the absolute values of each of the off diagonal terms. So, this term is we are guaranteed that this b j infinity is going to be less than one and since b j infinity is less than one what was our sufficient condition.

(Refer Slide Time: 53:28)

**Convergence of Iterations**

- Hence we need a convergence criterion that is simpler to use
- Recall that  $\mathbf{x}^{(k)} - \mathbf{x} = \mathbf{B}^k (\mathbf{x}^{(0)} - \mathbf{x})$   
Hence,  

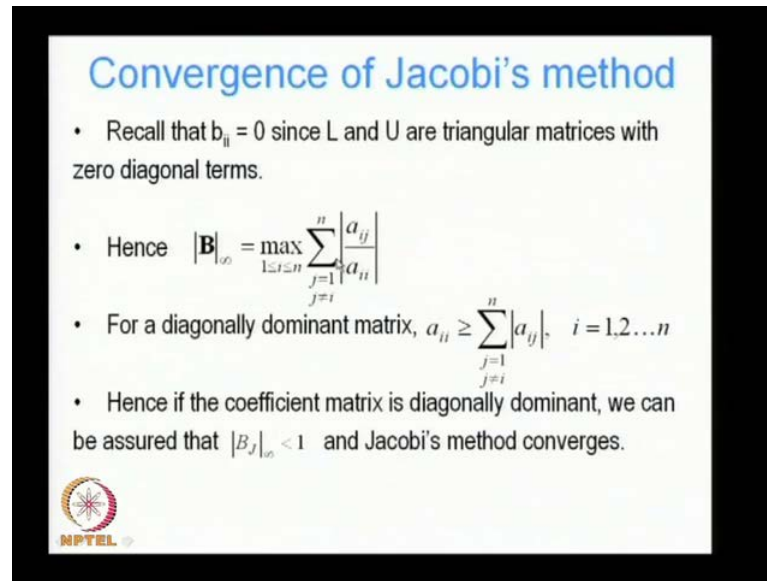
$$|\mathbf{x}^{(k)} - \mathbf{x}| \leq |\mathbf{B}^k| |\mathbf{x}^{(0)} - \mathbf{x}| \leq |\mathbf{B}|^k |\mathbf{x}^{(0)} - \mathbf{x}|$$
- Thus  $|\mathbf{B}| < 1$  in some consistent norm is a sufficient condition for convergence
- For Jacobi's method:  

$$B_J = -(L + U). \text{ Hence, } b_{ij} = -\frac{a_{ij}}{a_{ii}}, i \neq j, b_{ii} = 0$$

Our sufficient condition was that mod of b must be less than one in some norm and we have shown that in the infinite norm b j is less than 1.




(Refer Slide Time: 53:25)



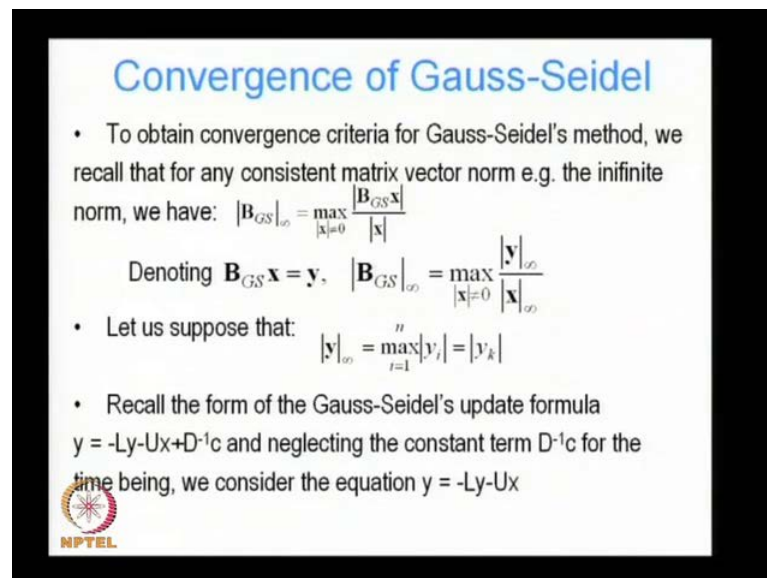
**Convergence of Jacobi's method**

- Recall that  $b_{ii} = 0$  since L and U are triangular matrices with zero diagonal terms.
- Hence  $\|\mathbf{B}\|_{\infty} = \max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^n \left| \frac{a_{ij}}{a_{ii}} \right|$
- For a diagonally dominant matrix,  $a_{ii} \geq \sum_{j=1, j \neq i}^n |a_{ij}|$ ,  $i = 1, 2, \dots, n$
- Hence if the coefficient matrix is diagonally dominant, we can be assured that  $\|B_j\|_{\infty} < 1$  and Jacobi's method converges.




So, that means that my Jacobi iteration Jacobi method is going to converge as I increase the number of iterations.

(Refer Slide Time: 53:52)



**Convergence of Gauss-Seidel**

- To obtain convergence criteria for Gauss-Seidel's method, we recall that for any consistent matrix vector norm e.g. the infinite norm, we have:  $\|\mathbf{B}_{GS}\|_{\infty} = \max_{\|\mathbf{x}\|_{\infty} = 1} \|\mathbf{B}_{GS}\mathbf{x}\|_{\infty}$
- Denoting  $\mathbf{B}_{GS}\mathbf{x} = \mathbf{y}$ ,  $\|\mathbf{B}_{GS}\|_{\infty} = \max_{\|\mathbf{x}\|_{\infty} = 1} \|\mathbf{y}\|_{\infty}$
- Let us suppose that:  $\|\mathbf{y}\|_{\infty} = \max_{i=1}^n |y_i| = |y_k|$
- Recall the form of the Gauss-Seidel's update formula  $\mathbf{y} = -\mathbf{L}\mathbf{y} - \mathbf{U}\mathbf{x} + \mathbf{D}^{-1}\mathbf{c}$  and neglecting the constant term  $\mathbf{D}^{-1}\mathbf{c}$  for the time being, we consider the equation  $\mathbf{y} = -\mathbf{L}\mathbf{y} - \mathbf{U}\mathbf{x}$



So, next class we are going to look at the convergence of Gauss Seidel, which is slightly more complicated and then we are going to move on to a slightly modified version of Gauss Seidel. Hopefully, we will round off our next class with some discussion on iterative methods for computing Eigen values.

Thank you.