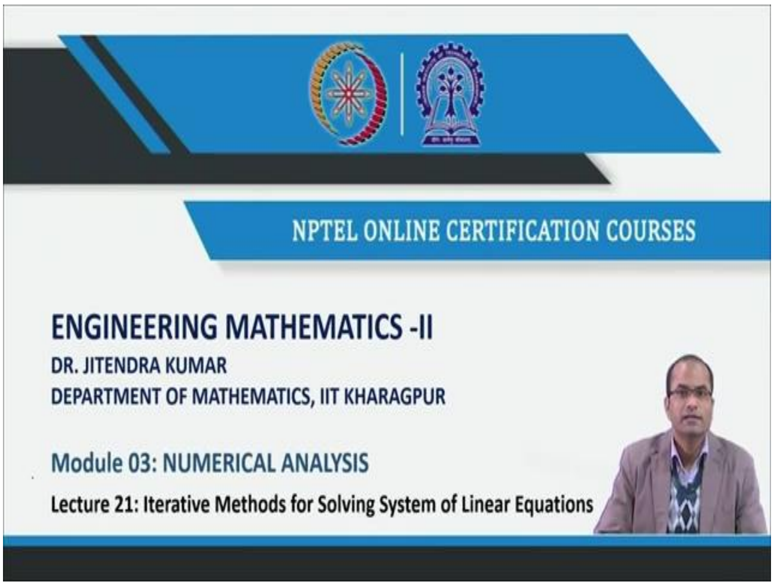
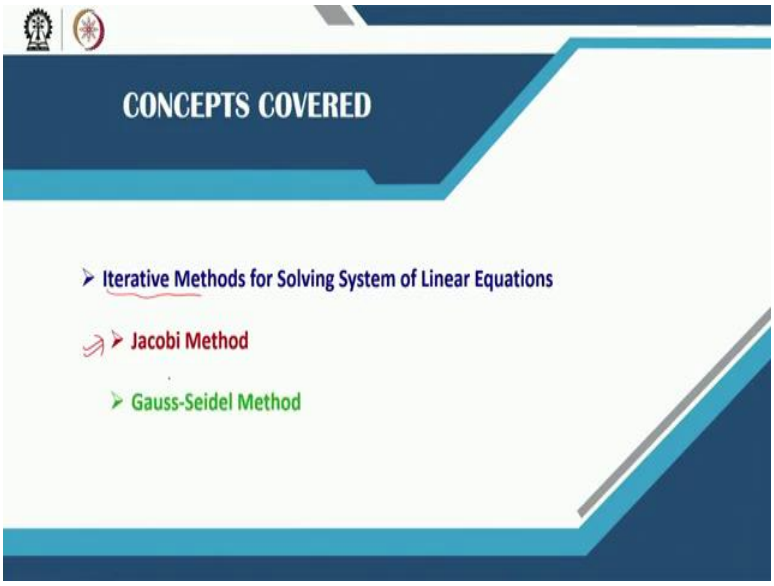
Engineering Mathematics-2
Professor Jitendra Kumar
Department of Mathematics
Indian Institute of Technology Kharagpur
Lecture: 21
Iterative Methods for Solving System of Linear Equations

(Refer Slide Time: 0:14)



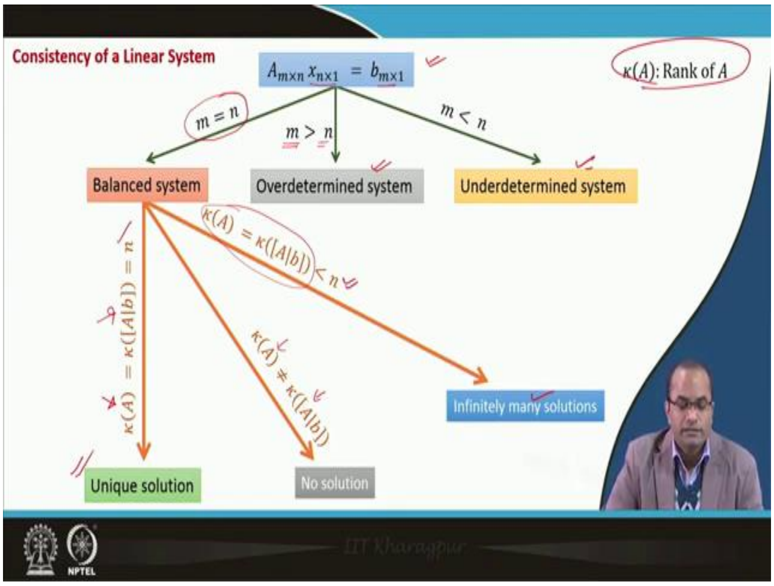
So, welcome back to lectures on engineering mathematics 2 and today we will begin with module 3 on numerical analysis. So this is lecture number 21 on iterative methods for solving system of linear equations.

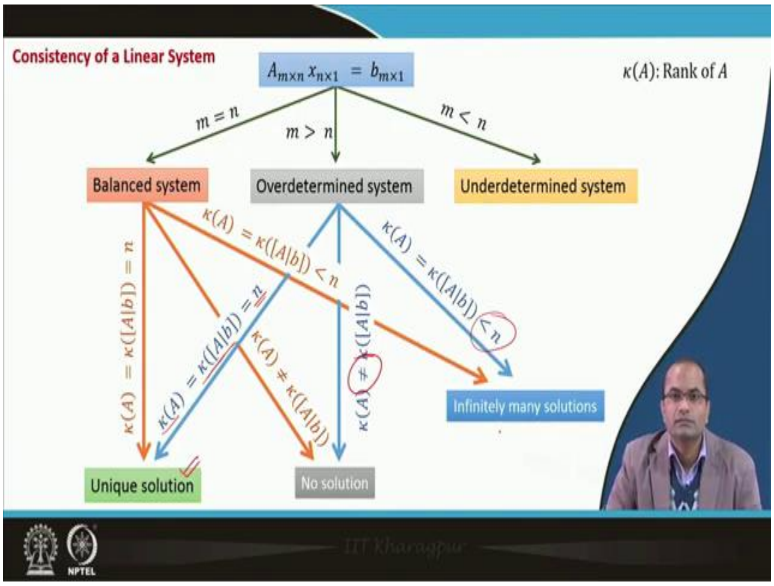
(Refer Slide Time: 00:30)

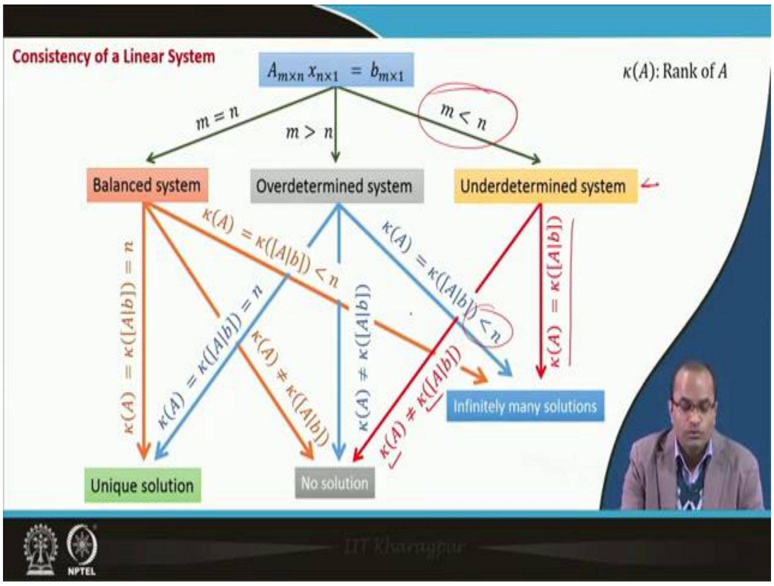


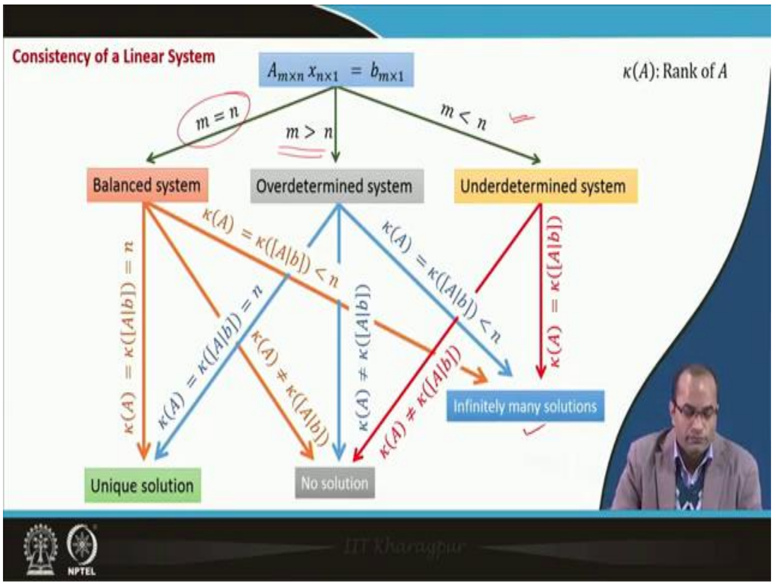
So, starting with the introduction to the iterative methods for solving system of linear equations we will basically discuss in detail two methods one is the Jacobi iteration method the other one is the Gauss-Seidel iterative method. So these two are the classical iterative methods which we will derive today in this lecture.

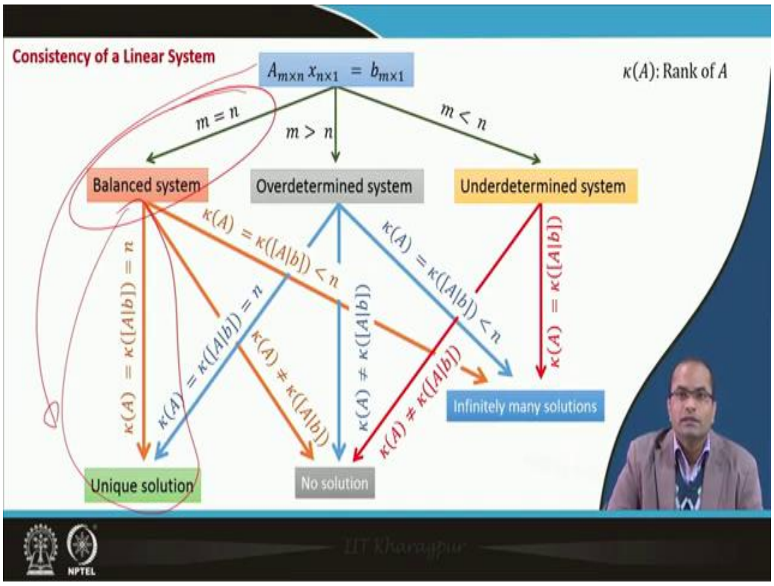
(Refer Slide Time: 00:52)






Consistency of a Linear System
$A_{m\times n} x_{n\times 1} = b_{m\times 1}$
$\kappa(A)$: Rank of A
$m = n$
$m > n$
$m < n$
Balanced system
Overdetermined system
Underdetermined system
$\kappa(A) = \kappa([A|b]) = n$
$\kappa(A) = \kappa([A|b]) < n$
$\kappa(A) = \kappa([A|b]) = n$
$\kappa(A) \neq \kappa([A|b])$
$\kappa(A) \neq \kappa([A|b])$
$\kappa(A) = \kappa([A|b]) < n$
$\kappa(A) \neq \kappa([A|b])$
$\kappa(A) = \kappa([A|b])$
Infinitely many solutions
Unique solution
No solution
IIT Kharagpur
NPTEL


Consistency of a Linear System
$A_{m\times n} x_{n\times 1} = b_{m\times 1}$
$\kappa(A)$: Rank of A
$m = n$
$m > n$
$m < n$
Balanced system
Overdetermined system
Underdetermined system
$\kappa(A) = \kappa([A|b]) = n$
$\kappa(A) = \kappa([A|b]) < n$
$\kappa(A) = \kappa([A|b]) = n$
$\kappa(A) \neq \kappa([A|b])$
$\kappa(A) \neq \kappa([A|b])$
$\kappa(A) = \kappa([A|b]) < n$
$\kappa(A) \neq \kappa([A|b])$
$\kappa(A) = \kappa([A|b])$
Infinitely many solutions
Unique solution
No solution
IIT Kharagpur
NPTEL


Consistency of a Linear System
$A_{m\times n} x_{n\times 1} = b_{m\times 1}$
$\kappa(A)$: Rank of A
$m = n$
$m > n$
$m < n$
Balanced system
Overdetermined system
Underdetermined system
$\kappa(A) = \kappa([A|b]) = n$
$\kappa(A) = \kappa([A|b]) < n$
$\kappa(A) = \kappa([A|b]) = n$
$\kappa(A) \neq \kappa([A|b])$
$\kappa(A) \neq \kappa([A|b])$
$\kappa(A) = \kappa([A|b]) < n$
$\kappa(A) \neq \kappa([A|b])$
$\kappa(A) = \kappa([A|b])$
Infinitely many solutions
Unique solution
No solution
IIT Kharagpur
NPTEL

So just to begin with to remind the consistency of a linear system which was also discussed in engineering mathematics 1 in this action of linear algebra. So, here we are talking about let us say the system of linear equations having m equations n unknown. So this A the matrix A having dimension m cross n, so there are m rows and n columns. Similarly this vector x here has n rows and then similarly this vector b has m rows there. So we have this A m cross n and then n cross m this vector x and then this vector b having the m rows.

So now if we have the case here m equal to n what happens we call this as a balance system when m so the number of equations are greater than the number of unknowns then we call such as system as over determined system and the third case when m is less than n, so the number of equations are less than the number of unknowns then we call such a system as undetermined system. So just to recall that for balance system when we have the same number equations as the unknown here this K denotes the rank of matrix.

So the rank of A equal to rank of this augmented matrix Ab so b is just augmented next to this matrix A. So if the rank of the matrix A is equal to this rank of matrix A augmented b and if this rank is equal to n then we have the case of unique solution that was well discussed in linear algebra. The second possibility when these two ranks, so the rank of A is not equal to the rank of Ab rank of augmented matrix Ab then this was the case of no solution and the third possibility when these two ranks are equal.

So rank of A equal to rank of Ab but this number is less than equal to, or less than n. So in that case we have infinitely many solutions of the given linear system. The second now we have the over determine system again we have the possibility of having a unique solution that means if this rank of A is equal to rank of Ab and this number is equal to n then we have the case of unique solution from this over determined system as well.

And when these two rank are not equal so the rank of A is not equal to rank of Ab then this is the case of no solution and the third possibility we have when this rank these ranks are equal but it is again less than equal to n it is very similar to the first balance system then we have again the case of infinitely many solutions. So for all determine system also we can have the possibility of unique solution, no solution and infinitely many solutions.

The last case where we have the number of equations less than the number of unknown that means this undetermined system in this case we have 2 possibilities either this rank A is not equal to rank Ab in that case we have no solution but when this number is equal rank of A is equal to rank of Ab and naturally that rank is going to be less than n because the rank cannot be greater than m or n.

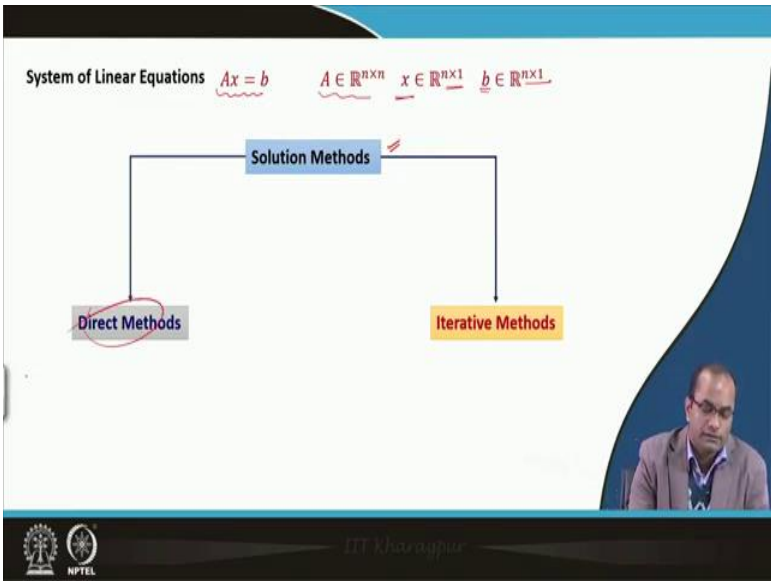
So it should be less than m maximum of m and n. So when these the rank of A is equal to rank of Ab it has to be less than n, so it is eventually the case which we have discussed more infinitely many solutions for balance and over determined system. So here also when this rank is equal we do not have to say less than n because which is obvious. So in this situation we have again infinitely many solutions.

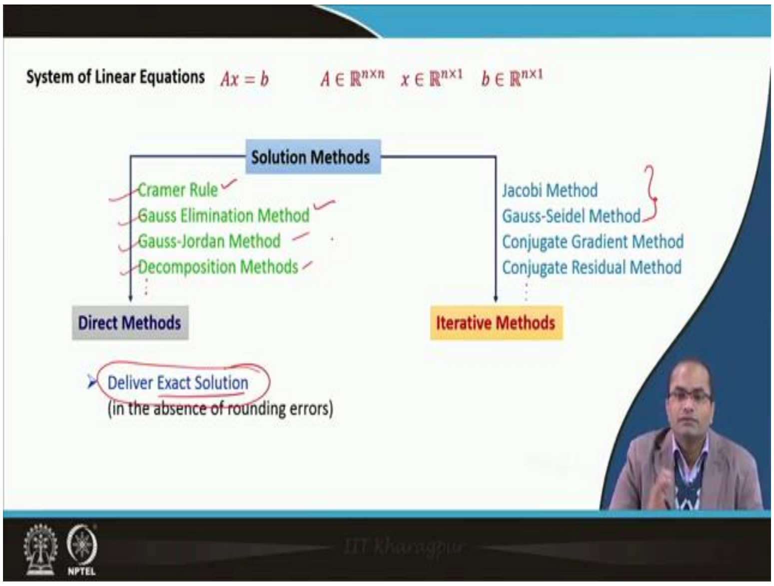
So this is the overall tree for a general linear system having the equal number of equations as the unknown here the number of equations are more than unknown and here the number of equations are less than the number of unknowns, and each case we have seen the possibilities having the unique solution no solution and in the last case undetermined system we have two possibilities that there would be no solution or infinitely many solution here we cannot have the situation of unique solution because number of equations are less than the number of unknowns.

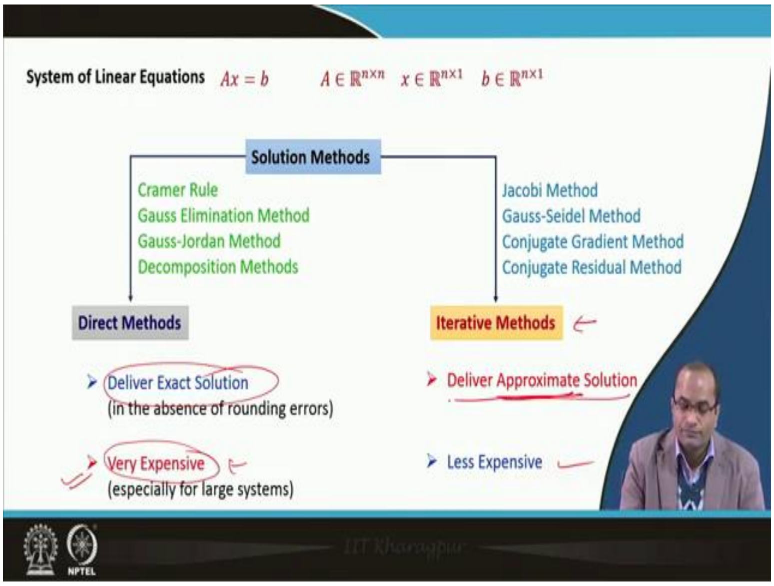
So we can always assign some free variables in case there is a solution, so there will be no possibilities of having unique solution in that case. What we are going to discuss now the iterative methods in this course we are restricting our self to this balance system. So when the number of equations are equal to the number of unknowns and that too the situation when we have the unique solution.

So only this case will be discussed now for the balance system when we have the unique solution in that case we will discuss now how to find the solution using the so called iterative methods.

(Refer Slide Time: 07:00)







So let us discuss the solutions for the system Ax is equal to b where A is n cross n matrix x is column vector of dimension n cross n, b is again a column vector of dimension n cross n. So we have n unknowns n equations and then we assume that the system is having a unique solution so what are the possible solution methods for such a system we have naturally the so called direct methods.

Which and we have here the iterative methods so broadly they are two categories when we call the direct methods another one we call as iterative methods. In the category of this direct methods we have Cramer rule, which this students are familiar with we have the Gauss elimination method which was discussed in linear algebra engineering mathematics 1 and we have the Gauss Jordan method.

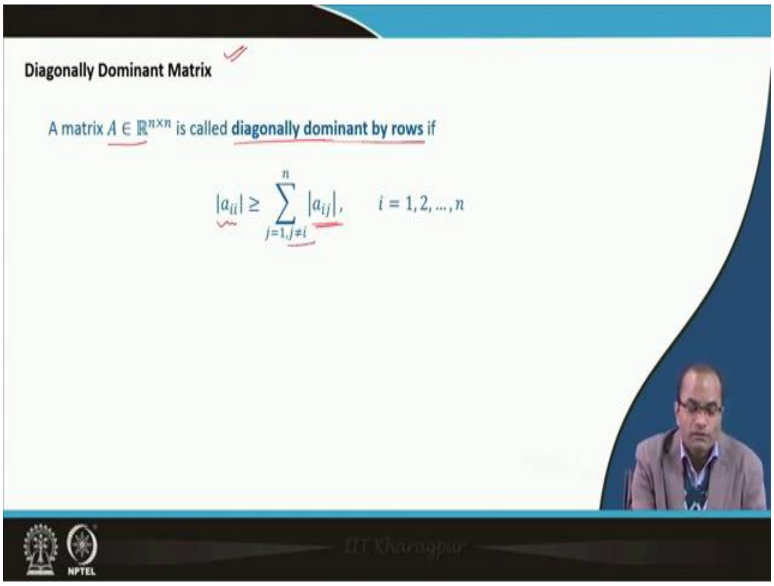
So again similar approach and we have also many other the so called Decomposition methods etc. So it is a long list we will continue and in this categories when we are talking about the iterative methods the two well-known classical methods are the Jacobi method and the Gauss Seidel method, we have many more some of them are like Conjugate Gradient method, Conjugate Residual method etc.

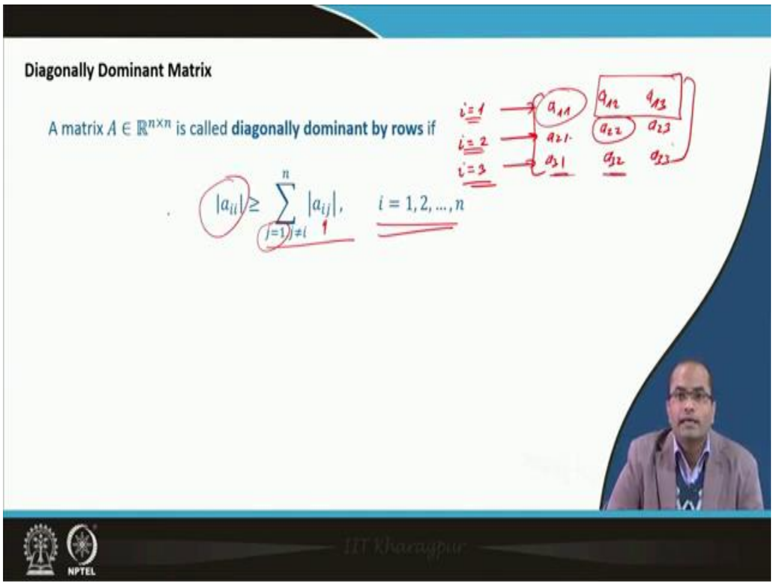
So it is again a long list for iterative method some of them are listed here. What are the characteristics of this direct methods they deliver exact solution so that is the beauty here in the absence of rounding of error they deliver exact solution of the given system like whether it is a Cramer rule or Gauss elimination, Gauss Jordan and many decomposition methods. So they deliver exact solution but there might be approximation due to rounding error etc.

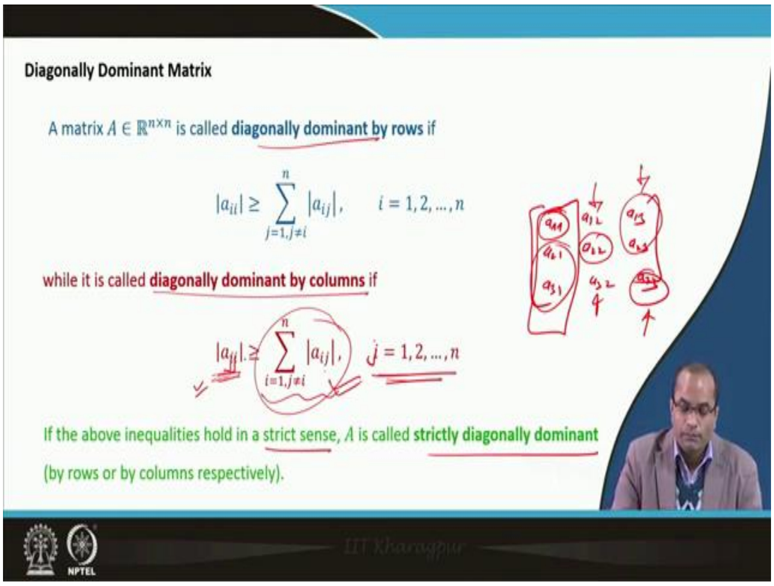
Whereas the iterative methods and that is the difference between the two category here they deliver approximate solution. So here there will be a sequence of solution which we will be obtaining they will be approximating the given system and we have to stop at some point of time depending on the set accuracy at the beginning. So here they end up with some kind of approximate solution and the accuracy will depend how many iterations we are doing to get the solution.

Here there was a good point that they provide exact solution but the draw back here is that they are very expensive especially when they have they are used for large system then it is almost impossible to work with the direct methods because they are very, very time consuming and that is the reason that most of the time for practical problems we go for iterative methods because they are less expensive and through this iteration we can really solve very very large system where this direct methods are just restricted to a way small system of dimension 30, 40, 50 whatever. But when we have the order in thousands or lakhs then we go with these iterative methods.

(Refer Slide Time: 10:35)







Well so before we begin with the derivation of the method there are some terminologies which will be discussed which will be used. So one of them is what is diagonally dominant matrix? Matrix A is called diagonally dominant by rows and when we call this diagonally dominant by rows if ai, i is greater than, so the diagonal entry of the ith row let us say is greater than or equal to the sum of all other entry so except this j equal to i.

So we are not adding again the diagonal entry. So except this diagonal entry when we add all element because here the submission is going over this j the second one. So if we have the matrix a11, a12, a13 that is the standard notation we use for the coefficients a23 then we have a31 a32 and then we have a33. So what we will do for each i so i corresponds to rows here. So i is equal to 1 for instance we have i is equal to 2 we have i is equal to 3 and so on.

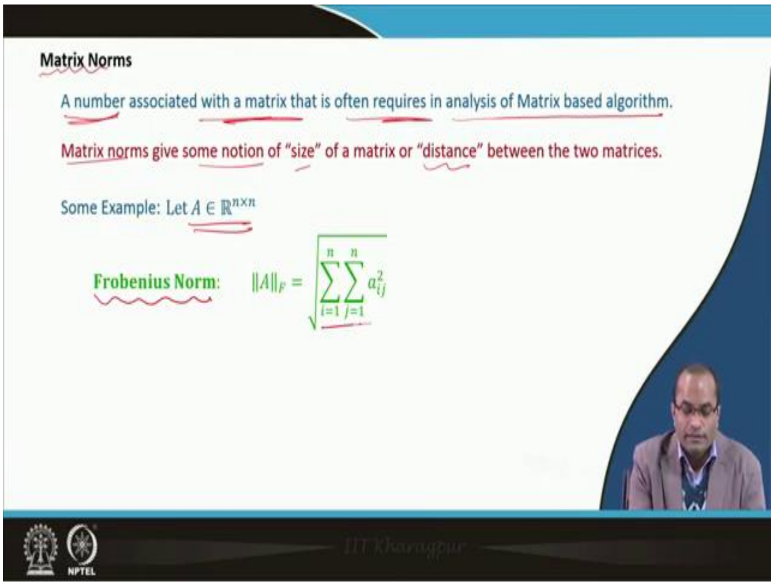
So for each i if we add the elements the absolute value except the first one now here in this case here we will leave this and add a21 and a23 as per this notation here will add a31 and a32, so if the diagonal entry has larger magnitude then the rest then we call diagonally dominant by rows. While it is called diagonally dominant by columns it is other way round, now if we go with the columns the first columns the first entry a11 should be greater than when we add all other elements of that column if it is greater than equal to the sum here and that we have to do for each i.

So i equal to 1, 2, 3 and so on for a22 then a33 and so on for all these this has to be full filled so for each column now we will check this for each infect whether to write here a j j and then put here j is equal to 1, 2, 3 and so on because the matrix her going to be now a11, a12, a13 a21, a22, a23 and a31, a32, a33. So the first column we will pick and then we will check whether the first entry is greater than some of the absolute values of the last two then in the column two.

So j equals to 2 we will check again whether this and this sum is greater than is less than this value here and similarly we will check here whether this element here is greater than the sum of absolute value of the first two. So we will go with the column wise. So let us say here ajj is greater than i equal to 1. So we will sum in this direction for j equal to 1 than for j is equal to 2 we will sum and j is equal to 3 we will sum all these.

Well if the above any quality is hold in a strict sense then A is called a strictly diagonally dominant. So if we have greater than equal to we simply called diagonally dominant respectively by columns and rows but if these inequalities are strict then we call strictly diagonally dominant. So that is the one definition we will be using in following lecture or in this lecture.

(Refer Slide Time: 14:26)



Matrix Norms

A number associated with a matrix that is often requires in analysis of Matrix based algorithm.

Matrix norms give some notion of "size" of a matrix or "distance" between the two matrices.

Some Example: Let $A \in \mathbb{R}^{n \times n}$

Frobenius Norm: $\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2}$

Row Sum Norm: $\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|$

Matrix Norms

A number associated with a matrix that is often requires in analysis of Matrix based algorithm.

Matrix norms give some notion of "size" of a matrix or "distance" between the two matrices.

Some Example: Let $A \in \mathbb{R}^{n \times n}$

Frobenius Norm: $\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2}$

Row Sum Norm: $\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|$

Coloumn Sum Norm: $\|A\|_\infty = \max_j \sum_{i=1}^{n} |a_{ij}|$

Matrix Norms

A number associated with a matrix that is often requires in analysis of Matrix based algorithm.

Matrix norms give some notion of "size" of a matrix or "distance" between the two matrices.

Some Example: Let $A \in \mathbb{R}^{n \times n}$

Frobenius Norm: $\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2}$

Row Sum Norm: $\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|$

Coloumn Sum Norm: $\|A\|_\infty = \max_j \sum_{i=1}^{n} |a_{ij}|$

Another terminology we are going to have it, it is a matrix norms just a brief very brief introduction the details are not provided here. So it is just to use these terminologies in these numerical analysis portion. So this is the number basically associated with a matrix that is often requires analysis of matrix base algorithm. So here also we are talking about now the matrix and these iterative algorithm based on the matrices.

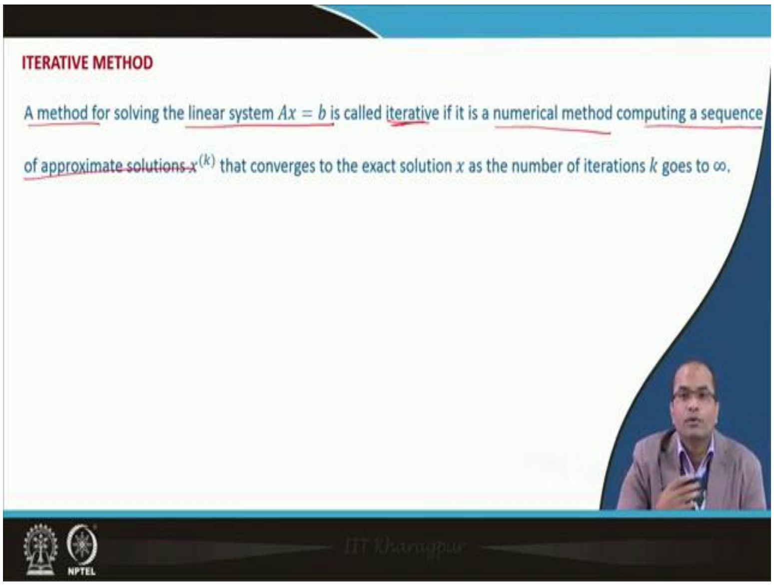
So here this number which is call the matrix norm it is a very useful for defining for instance the convergence etc. That slowly we will discuss, so this matrix norm give some notion of the size of a matrix because it is a number associated with each matrix. So we can for example compare 2 matrices in this sense or we can find the distance between the two matrix how whether they are close to each other or they are very different from each other.

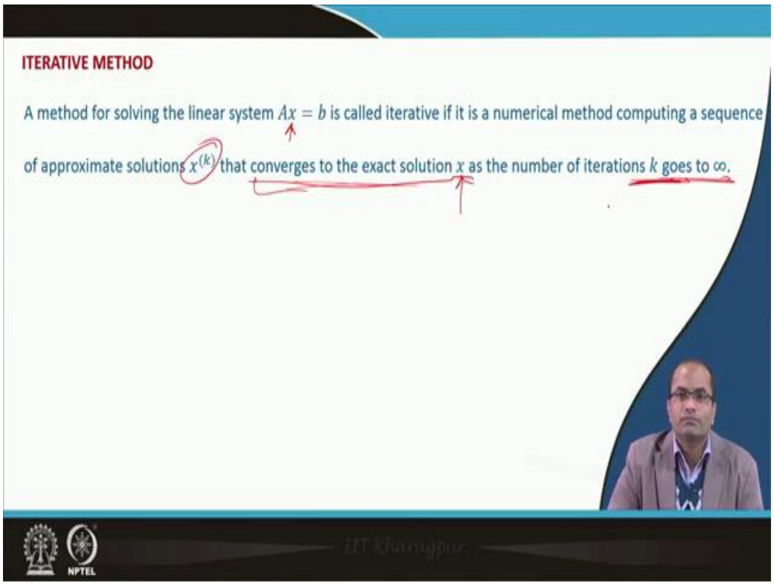
So those notions can be also captured with this matrix norms which we are going to define now. So some example of the matrix norm because there are many ways to define the matrix norms but some of them are very popular one very standard one we will state here suppose A is a n cross n matrix then one is called the Frobenius Norm which is defined as in this way. So we will square each element of the matrix and then take this square root so this is called the Frobenius Norm.

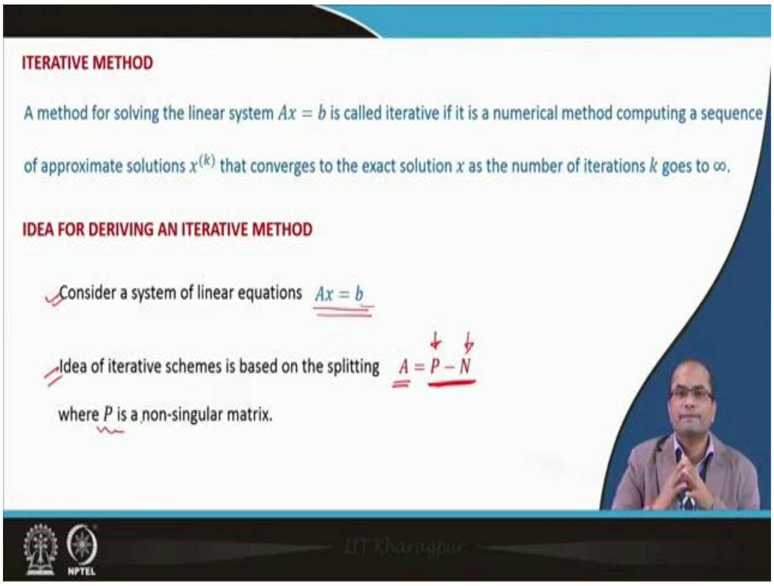
So this is naturally going to be positive number which we are denoting by this norm A this is the notation for the norm and F stands for Frobenius. Row sum norm the other one is the row sum norm as the name suggest that we will sum each row and we will take than the maximum among these numbers. So the sum of each row of the absolute value the sum of the absolute value of each row will give a number and then we will find out what is the maximum among these numbers.

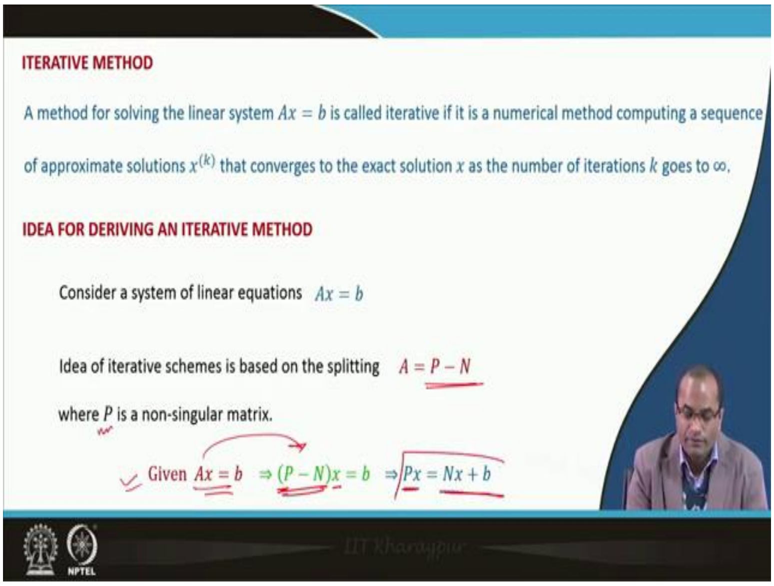
That will be the row sum norm or we have the column sum norm where we will sum now each column and then among these values the sum of each column the absolute values of each element in a column that the maximum will be (())(17:01) and that is going to be now the column sum norm. So only these three norms at this moment we will adjust the state and later on perhaps we will go little more into the details. So these are the numbers associated with a matrix which are called Frobenius norm row sum norm and the column sum norm.

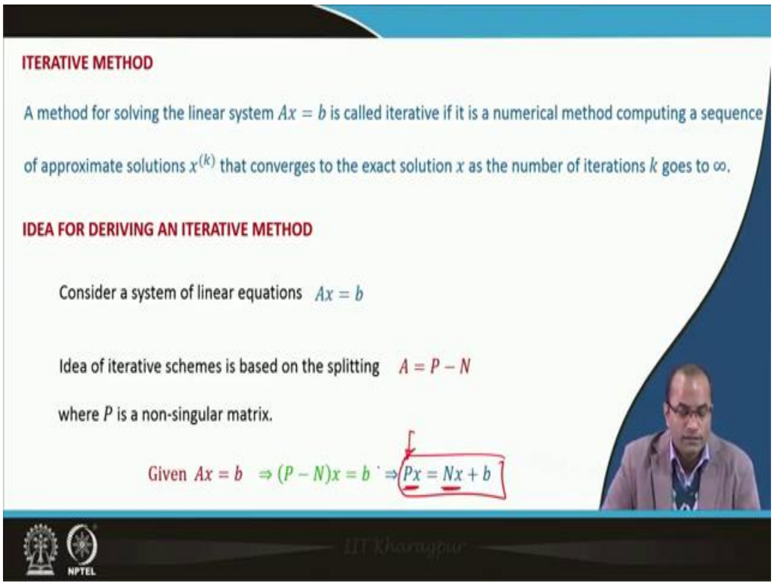
(Refer Slide Time: 17:26)











Now coming to the iterative methods so a method for solving linear system Ax is equal to b is called iterative if it is a numerical method computing a sequence of approximate solutions it will compute a sequence of approximate solution this x K we are denoting this x K, x is a vector which is the solution of this system here and xK is the sequence which is approximating this x. So x1, x2, x3 and so on and in such a way that should converge to the exact solution if we keep on going with sequence finally our solution should go to x to the exact solution.

As basically the number of iteration goes to the infinity so such a method when we have a sequence of the solutions though the sequence are actually the approximation of the solution of the given system and if such a sequence goes to exact solution or this is what we call converges to the exact solution x as the iteration goes to infinity then we call that this is a iterative method.

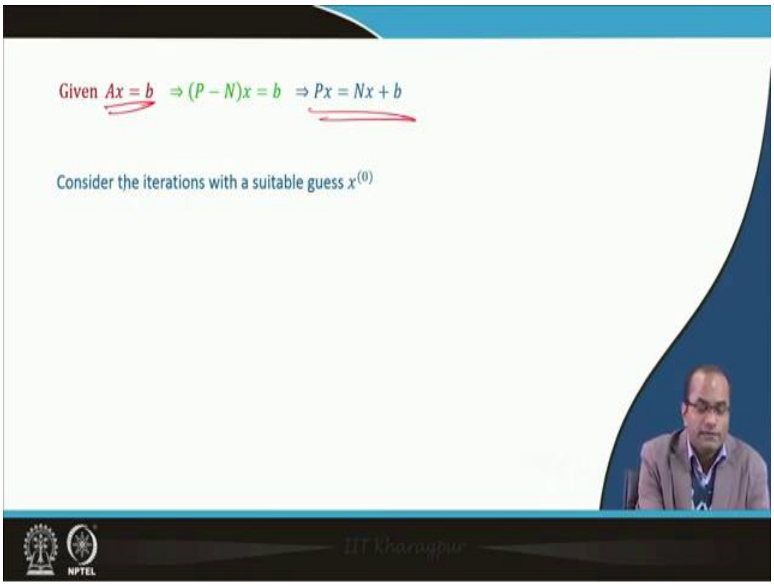
In many cases the sequence may not go to, may not converge to x and that is also part of discussion that under what condition this method actually particular method converges to the actual solution of the system. So what is the idea? This is the general idea of deriving an iterative method this is a very general approach here but we will go for specific cases in next slides.

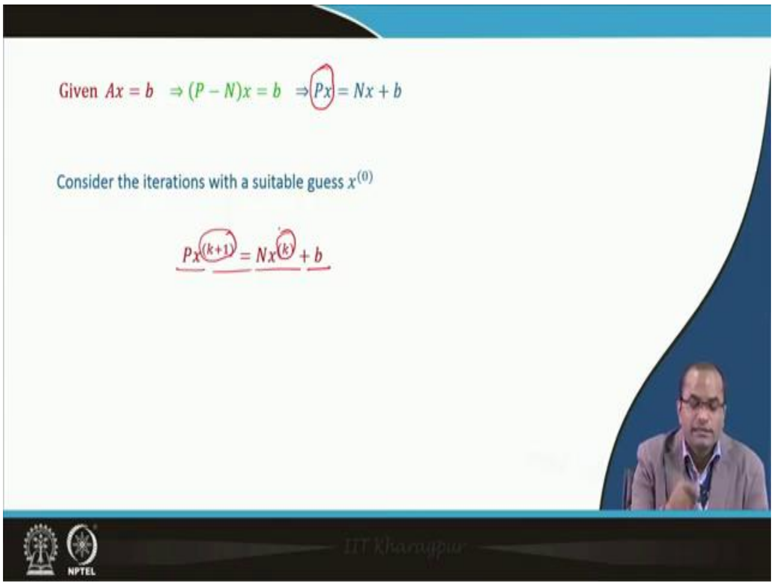
So consider for the system of linear equations Ax is equal to b having this n and cross n size of A and respectively the N cross 1 and N cross 1 size of P. So the idea for this iterative scheme is based on this splitting of the matrix A so we split this matrix in some kind of these matrices so matrix P minus the matrix N is giving us this A and in such a way that P is a non-singular matrix, I am depending on this is splitting we will have a different, different numerical methods.

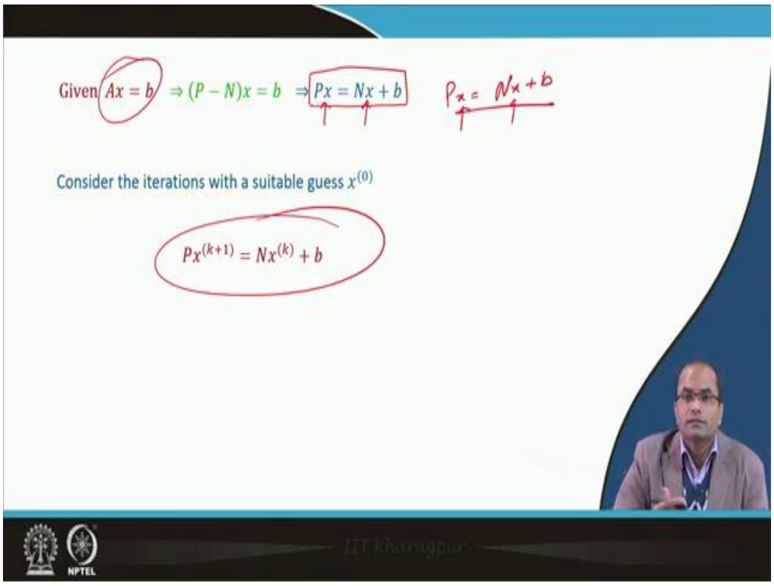
But overall in each of these iterative methods we will do some kind of splitting in such a way that this one matrix P is non-singular matrix. So given the equations Ax is equal to b what we will do A we have written as P minus A and x equal to b or we can have this P times x is equal to Nx plus b. So we have formed here another equation where we have Px so the x is sitting left hand side x is also sitting right hand side and we know that the P is a non-singular matrix.

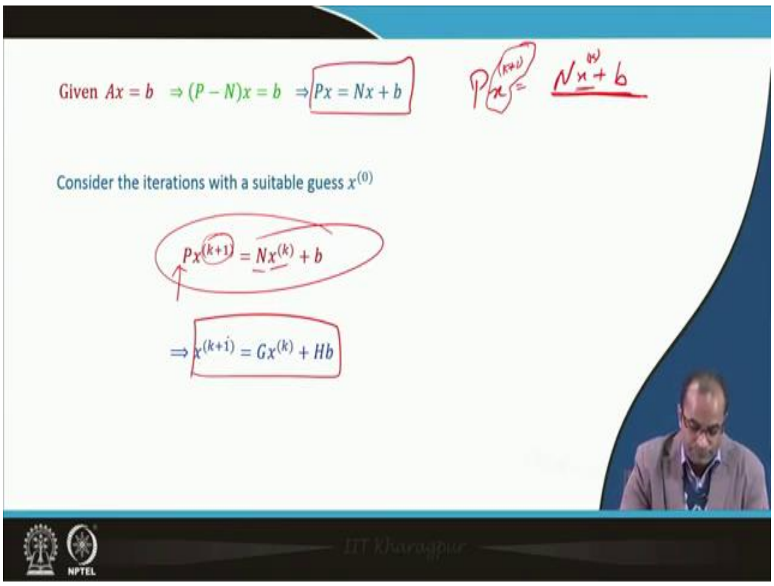
So that we can invert it and having to have this x completely one x to the left hand side and the rest goes to the right hand side.

(Refer Slide Time: 20:41)




Given $Ax = b \Rightarrow (P - N)x = b \Rightarrow Px = Nx + b$
Consider the iterations with a suitable guess $x^{(0)}$
$Px^{(k+1)} = Nx^{(k)} + b$
NPTEL
IIT Kharagpur


Given $Ax = b \Rightarrow (P - N)x = b \Rightarrow Px = Nx + b$
Consider the iterations with a suitable guess $x^{(0)}$
$Px^{(k+1)} = Nx^{(k)} + b$
NPTEL
IIT Kharagpur


Given $Ax = b \Rightarrow (P - N)x = b \Rightarrow Px = Nx + b$
Consider the iterations with a suitable guess $x^{(0)}$
$Px^{(k+1)} = Nx^{(k)} + b$
$\Rightarrow x^{(k+1)} = Gx^{(k)} + Hb$
NPTEL
IIT Kharagpur

Given $Ax = b \Rightarrow (P - N)x = b \Rightarrow Px = Nx + b$

Consider the iterations with a suitable guess $x^{(0)}$

$$Px^{(k+1)} = Nx^{(k)} + b$$

$$\Rightarrow x^{(k+1)} = Gx^{(k)} + Hb$$

Given $Ax = b \Rightarrow (P - N)x = b \Rightarrow Px = Nx + b$

Consider the iterations with a suitable guess $x^{(0)}$

$$Px^{(k+1)} = Nx^{(k)} + b$$

$$\Rightarrow x^{(k+1)} = Gx^{(k)} + Hb$$

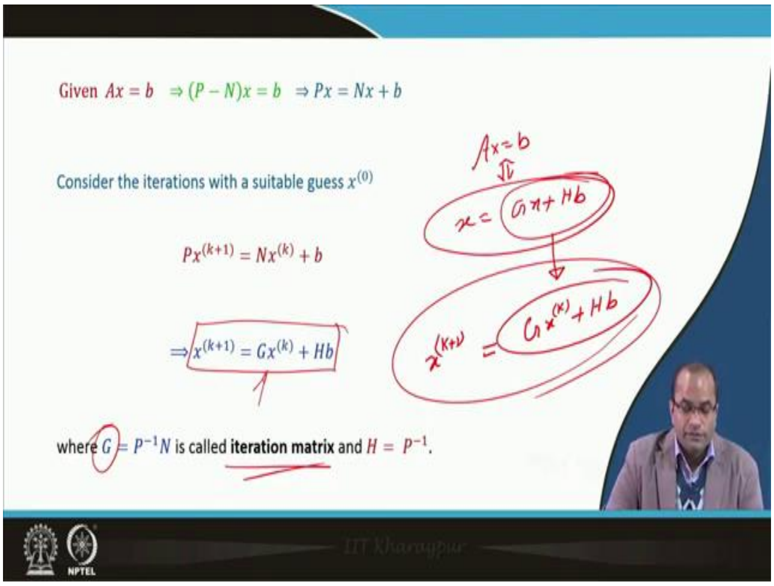
Given $Ax = b \Rightarrow (P - N)x = b \Rightarrow Px = Nx + b$

Consider the iterations with a suitable guess $x^{(0)}$

$$Px^{(k+1)} = Nx^{(k)} + b$$

$$\Rightarrow x^{(k+1)} = Gx^{(k)} + Hb$$

where $G = P^{-1}N$ is called **iteration matrix** and $H = P^{-1}$.



So having this Px is equal to Nx plus b from the given Ax is equal to b what we do now we can consider the iteration with a suitable gas that is another point here for each iterative scheme we have to also choose initial guess or initial solution to start with and that is what we call this x 0. So this Px K is equal to Nx K plus b this is what. So the left hand side of this x we have kept here the K plus 1 iteration given the value at the this kth iteration and that is exactly the point how we define the iteration here.

Here the equation was Px is equal to Nx plus b here the x, this x will satisfy this equation exactly. So when x is a exact solution naturally this Px is going to be N x plus b because this is the given Ax is equal to b. So if we have a vector x which satisfy is this equation that is naturally the solution of the given system of equation. So having this Px is equal to Nx plus b what we do supply here the right hand side some known solutions starting with x 0 and then compute Nx plus b and then if this is equal to this P times x power K plus 1.

And in this way we search for which what is the x K plus 1, so the value expected the improved value or let us first look at this, so this P is invertible matrix. So we have instead of writing this Px power k plus 1 not power the notations so k plus 1 and x K plus b so we have written finally in this form that x k plus 1 is equal to Gx K plus xb.

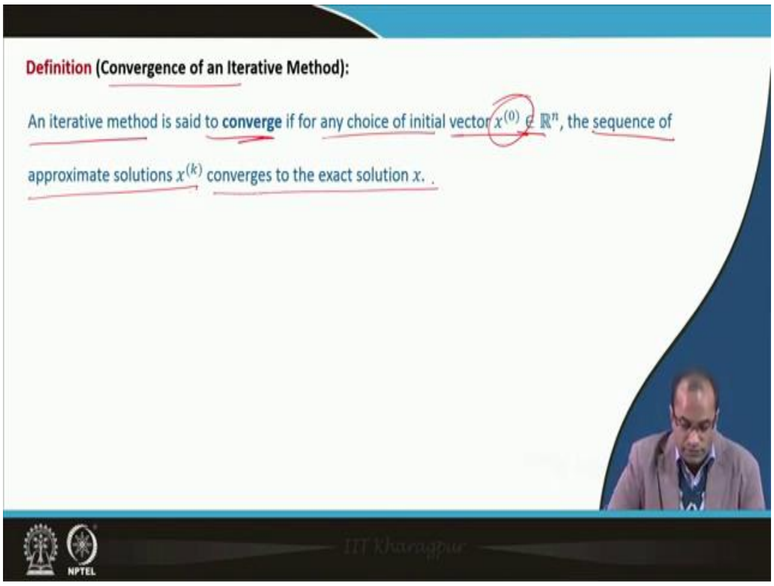
So given a value here of this x K we will compute this Gx K plus Hb are that we will call this number as x K plus 1 because substituting some value of this xK there Gx K plus Hb is not given… will not be the exact solution because the exact solution will be Gx plus Hb when we give here the exact solution then only we will get here x this is the given system of equation but what we are doing here? We are putting some other value x K.

So we have to start with for example x0. So we will first give here x0 we have to start with and then add this Hb there multiply with this G and this we will call this is our x 1. So which is suppose to be a better approximation than what we have a chosen at the beginning and then this x1 we will substitute there and then we will get a new and so on. So in this way we will proceed with the algorithm.

So this G here which is P inverse n this is an very important matrix so called iteration matrix because the convergence of these schemes will depend on this iteration matrix and then here we have this H as just simply P inverse which is coming from there. So what is important here we have rewritten our system into this form the x is equal to this G x plus this Hb form this is our system Ax is equal to b is written in this form and from this we have setup iteration that provide there some values of x and then compute Gx plus this Hb and whatever we get this number let set to the value at K plus 1 iteration.

And in this way when we trade this we are supposed to go towards the actual solution and this G plays a very important role which is called the iteration matrix of the given scheme.

(Refer Slide Time: 24:59)



Definition (**Convergence of an Iterative Method**):

An iterative method is said to **converge** if for any choice of initial vector $x^{(0)} \in \mathbb{R}^n$, the sequence of approximate solutions $x^{(k)}$ converges to the exact solution x.

Definition (**Error**):

We call the vector $r_k = b - Ax^{(k)}$ **residual** (respectively **error** $e_k = x^{(k)} - x$) at the kth iteration.

Definition (**Convergence of an Iterative Method**):

An iterative method is said to **converge** if for any choice of initial vector $x^{(0)} \in \mathbb{R}^n$, the sequence of approximate solutions $x^{(k)}$ converges to the exact solution x.

$b - Ax = 0$

$Ax = b$

Definition (**Error**):

We call the vector $r_k = b - Ax^{(k)}$ **residual** (respectively **error** $e_k = x^{(k)} - x$) at the kth iteration.

IIT Kharagpur

Definition (**Convergence of an Iterative Method**):

An iterative method is said to **converge** if for any choice of initial vector $x^{(0)} \in \mathbb{R}^n$, the sequence of approximate solutions $x^{(k)}$ converges to the exact solution x.

Definition (**Error**):

We call the vector $r_k = b - Ax^{(k)}$ **residual** (respectively **error** $e_k = x^{(k)} - x$) at the kth iteration.

IIT Kharagpur

Definition (**Convergence of an Iterative Method**):

An iterative method is said to **converge** if for any choice of initial vector $x^{(0)} \in \mathbb{R}^n$, the sequence of approximate solutions $x^{(k)}$ converges to the exact solution x.

Definition (**Error**):

We call the vector $r_k = b - Ax^{(k)}$ **residual** (respectively **error** $e_k = x^{(k)} - x$) at the kth iteration.

IIT Kharagpur

Definition (**Convergence of an Iterative Method**):

An iterative method is said to **converge** if for any choice of initial vector $x^{(0)} \in \mathbb{R}^n$, the sequence of approximate solutions $x^{(k)}$ converges to the exact solution x.

Definition (**Error**):

We call the vector $r_k = b - Ax^{(k)}$ **residual** (respectively **error** $e_k = x^{(k)} - x$) at the kth iteration.

IIT Kharagpur

Definition (**Convergence of an Iterative Method**):

An iterative method is said to **converge** if for any choice of initial vector $x^{(0)} \in \mathbb{R}^n$, the sequence of approximate solutions $x^{(k)}$ converges to the exact solution x.

Definition (**Error**):

We call the vector $r_k = b - Ax^{(k)}$ **residual** (respectively **error** $e_k = x^{(k)} - x$) at the kth iteration.

REMARK:

In general, we have no knowledge of e_k because the exact solution x is unknown. However, it is easy to compute the residual r_k, so convergence dedicated on the residual in practice.

IIT Kharagpur

So, now just few more definition for the convergence we are talking about the convergence of an iterative method, an iterative method is set to converge if for any choice of initial vector any choice of initial vector the sequence of approximate solution this x K converges to the exact solution then we call the iterative method converges is set to converge if for any choice any choice of the initial vector x naught the sequence of approximate solution converges to the exact solution x.

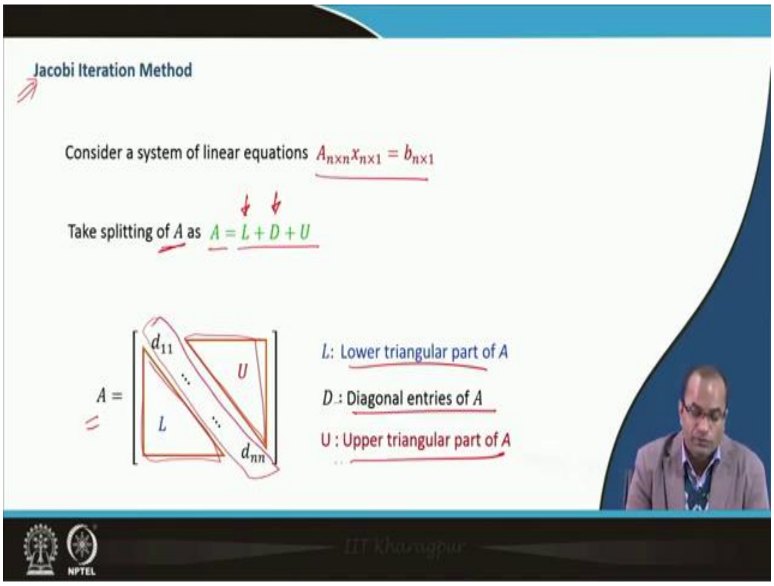
And the definition of the error because we will be also talking about error. We call the vector r K b minus Ax K the residual or respectively the error, error is define as a x k minus x at the Kth iteration. So here there are two types of errors one is the residual which is b minus Ax because b minus Ax is supposed to be 0 if x is the exact solution because A x is equal to b is our system.

But this is not 0 for those approximate solution. So we can have actually we can define I mean this gives the notion of the error itself and this is called the residual at the Kth iteration or the actual error is x K minus x but since the actual solution x may not be known. So this may not be used actually this error may not be computed because here we need this actual solution x.

But this b is known A is known and x K is the approximate solution. So we can readily compute this b minus Ax K and that will give actually the notion of error which is called the residual. Just a remark, in general we have no knowledge about this eK because the exact solution x is unknown. However it is easy to compute the residual this rK so the convergence normally is dedicated to the residual in practice.

So when we talk about the convergence in practice we check whether this b minus Ax K is close to 0 or it is going towards 0. So this is based on this residual only.

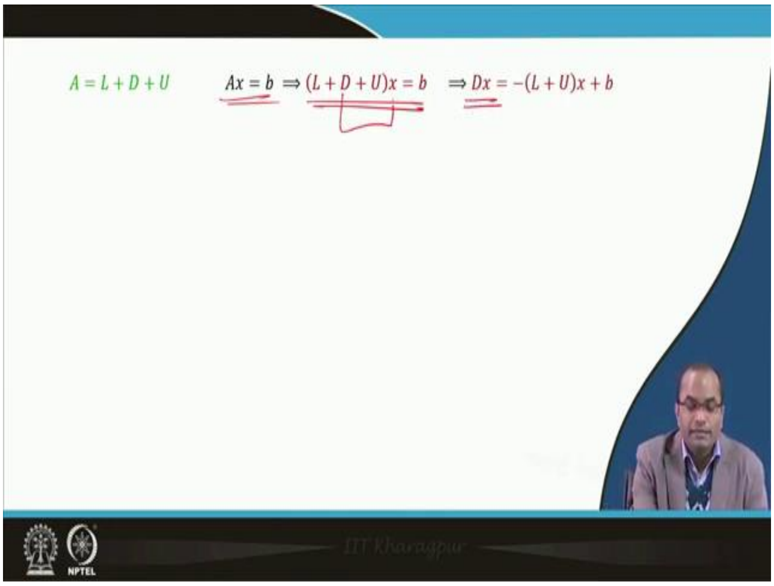
(Refer Slide Time: 27:26)



Now coming to the Jacobi iteration method, if we consider the system Ax is equal to b. So here now we can derive this particular type of iteration method. So take this splitting of A as A is split into L plus D plus U. Here we call the lower triangular matrix, this is the D is the diagonal entries of A so the diagonal matrix and U is upper triangular part of A. So if you have this type of splitting of A the lower portion here below this diagonal we call at L and then here we call this U and the middle one is the elements of the D.

So this is lower triangular having these elements L and the rest will set to be 0. In D only this D1, D2, D3 and Dn will be there in the matrix rest everything will be 0 in the upper one. We will have this upper entries of the matrix and the rest everything will be set to 0.

(Refer Slide Time: 28:30)



$A = L + D + U \qquad Ax = b \Rightarrow (L + D + U)x = b \quad \Rightarrow Dx = -(L + U)x + b$

Assume that D^{-1} exists, then $\Rightarrow x = -D^{-1}(L + U)x + D^{-1}b$

$A = L + D + U \qquad Ax = b \Rightarrow (L + D + U)x = b \quad \Rightarrow Dx = -(L + U)x + b$

Assume that D^{-1} exists, then $\Rightarrow x = -D^{-1}(L + U)x + D^{-1}b$

Introducing iterations, the iterative method known as Jacobi iteration method, becomes

$$x^{(k+1)} = -D^{-1}(L + U)\, x^{(k)} + D^{-1}b$$

$A = L + D + U \qquad Ax = b \Rightarrow (L + D + U)x = b \quad \Rightarrow Dx = -(L + U)x + b$

Assume that D^{-1} exists, then $\Rightarrow x = -D^{-1}(L + U)x + D^{-1}b$

Introducing iterations, the iterative method known as Jacobi iteration method, becomes

$$x^{(k+1)} = -D^{-1}(L + U)\, x^{(k)} + D^{-1}b$$

In component form

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(k)}\right) \quad i = 1, 2, \dots, n$$

$A = L + D + U$ $\quad Ax = b \Rightarrow (L + D + U)x = b \quad \Rightarrow Dx = -(L + U)x + b$

Assume that D^{-1} exists, then $\Rightarrow x = -D^{-1}(L + U)x + D^{-1}b$

Introducing iterations, the iterative method known as Jacobi iteration method, becomes

$$x^{(k+1)} = -D^{-1}(L + U)\, x^{(k)} + D^{-1}b$$

In component form

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(k)}\right) \quad i = 1, 2, \dots, n$$

$A = L + D + U$ $\quad Ax = b \Rightarrow (L + D + U)x = b \quad \Rightarrow Dx = -(L + U)x + b$

Assume that D^{-1} exists, then $\Rightarrow x = -D^{-1}(L + U)x + D^{-1}b$

Introducing iterations, the iterative method known as Jacobi iteration method, becomes

$$x^{(k+1)} = -D^{-1}(L + U)\, x^{(k)} + D^{-1}b$$

In component form

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(k)}\right) \quad i = 1, 2, \dots, n$$

$A = L + D + U$ $\quad Ax = b \Rightarrow (L + D + U)x = b \quad \Rightarrow Dx = -(L + U)x + b$

Assume that D^{-1} exists, then $\Rightarrow x = -D^{-1}(L + U)x + D^{-1}b$

Introducing iterations, the iterative method known as Jacobi iteration method, becomes

$$x^{(k+1)} = -D^{-1}(L + U)\, x^{(k)} + D^{-1}b$$

In component form

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(k)}\right) \quad i = 1, 2, \dots, n$$

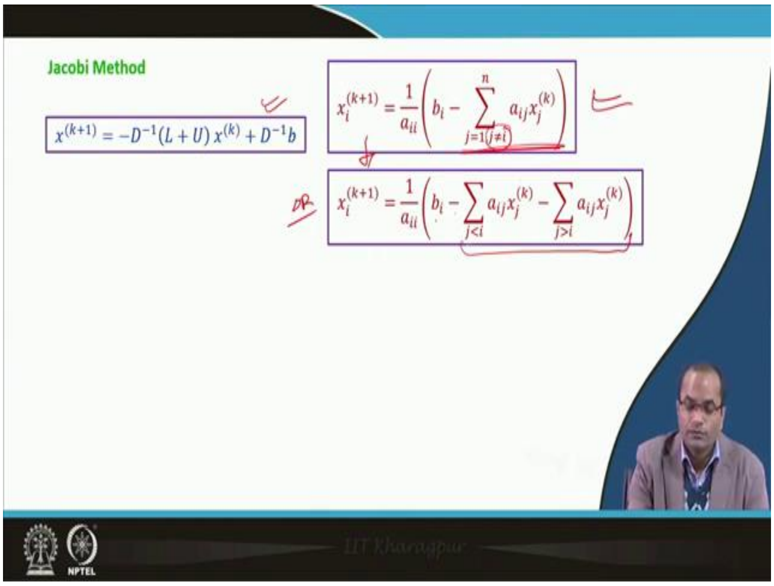
So having this splitting now A is equal to L plus D plus U, what we do? Ax is equal to b can be written now in this form L plus D plus U x is equal to be. That means now we take the D with the x and rest everything to the right hand side from here we will set up the scheme. So we have Dx is equal to minus L plus Ux is equal to b that is our equation and assuming that D inverse exist because D are the diagonal entries.

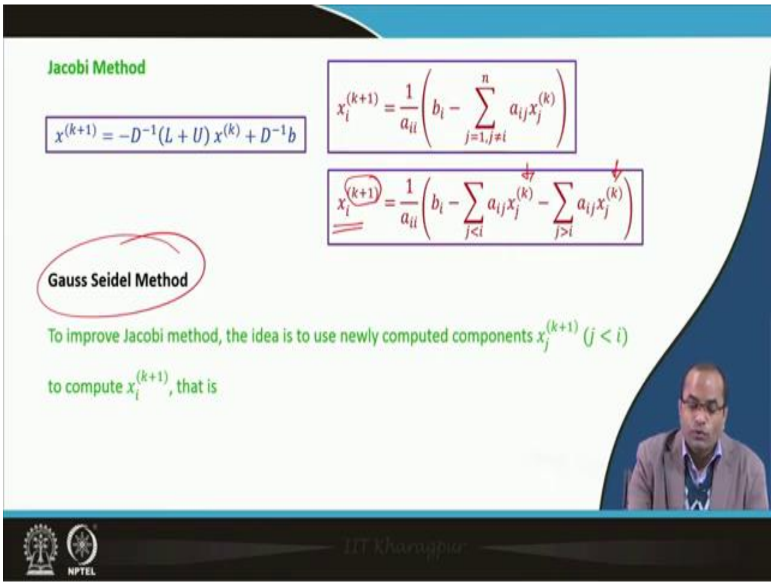
So assuming that the D inverse exist we can write down x is equal to minus D minus 1 the D inverse L plus Ux and D inverse b from here now we can set up our scheme that the left hand side we can put the value after K plus 1 at iteration given the values at Kth iteration. So the iteration method is given by this scheme now x K plus 1 is equal to minus D inverse L plus U x K plus this D inverse B this is the so called the Jacobi iterations method the only important point here is that A was splitted in this form and the D was taken for this inverse to the right hand side and we have this relation for x actual x.

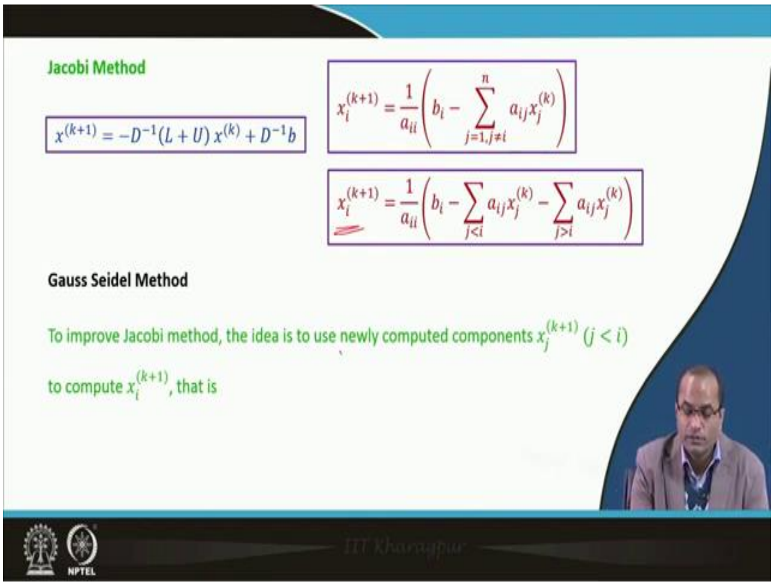
And then for approximate x supplying here we are getting the new value of x K plus 1. So in component form this we can also write in component form because this is written in matrix form so this is the matrix form but we can also write in component form. So component form means that all these unknowns x1, x2, x3 xn we have n unknowns this is exactly coming from this matrix form itself this D inverse because this was the diagonal entries.

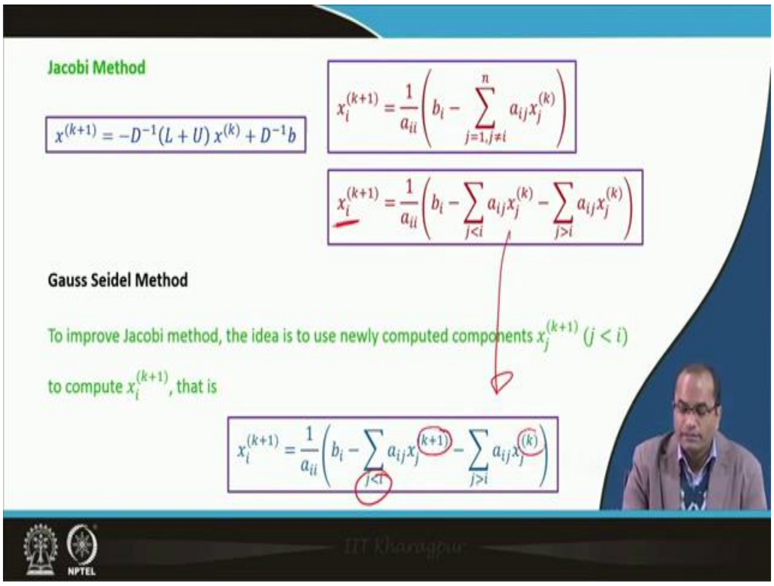
So only this 1 or ii, aii element will come here and D inverse b again so this bi will come the D inverse is already sitting there for the i th row. So bi and then the see here L plus U, so except this diagonal entry the rest are summed up. So a ij with this xj the Kth 1. So we will discuss this bit more in the next lecture. So there are two forms one is the diagonal form another one is the component form.

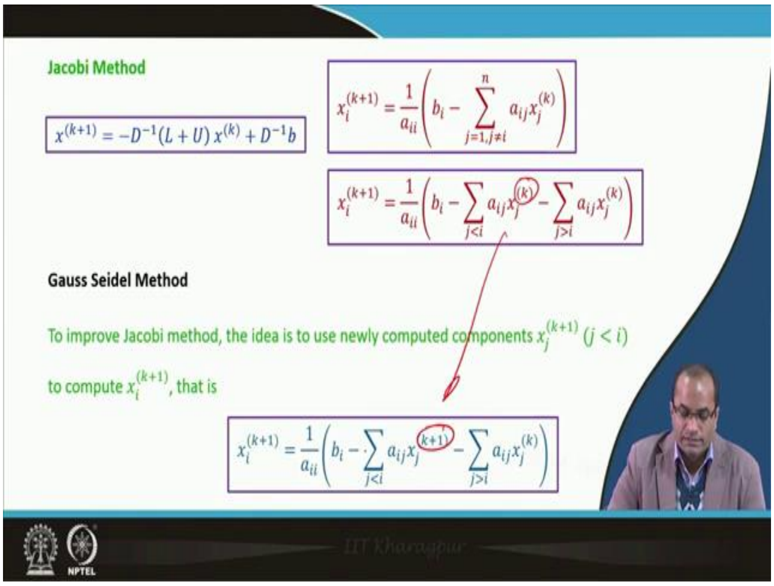
(Refer Slide Time: 31:01)











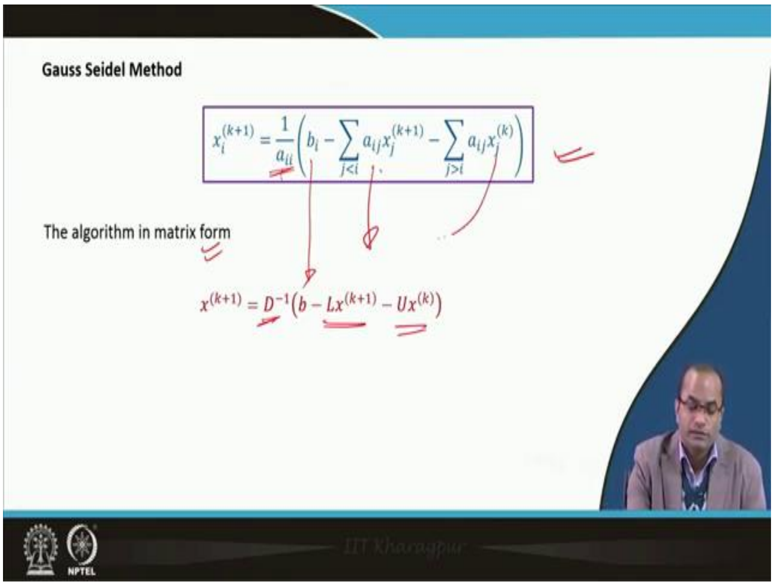
So now coming to this Jacobi only this is what we have seen either this form or whatever this form they are same form this we can rewrite here this one we can rewrite as 1 over ai bi minus this submission. We have splitted into two because the J is not equal to i we are not considering the i th entry there for the sum when j is equal to i so basically what we are doing when the sum is up to j less than i and then the sum is for j greater than i.

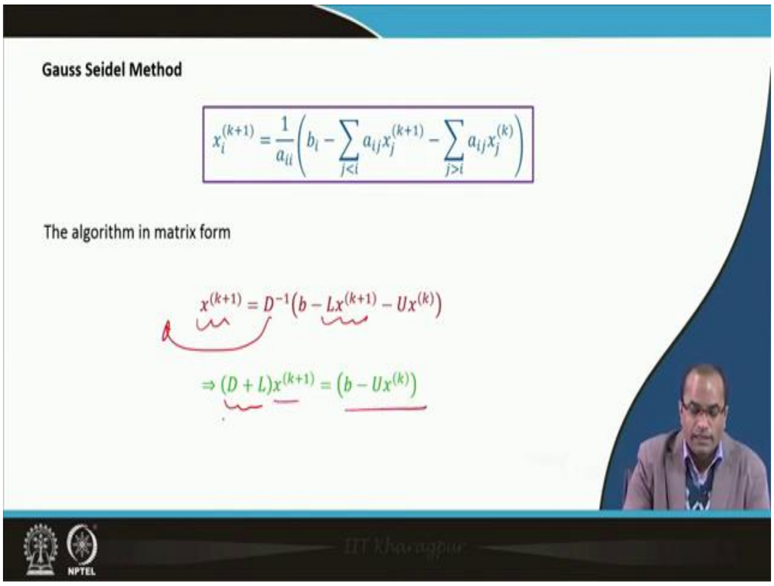
So this we can also rewrite in this form and this is exactly the point now we can derive the another method Gauss Seidel method which is usually or supposed to be a better approximation then the Jacobi method what we do in the Gauss Seidel method here this all values of x are kept as K th iteration and then all values of this X i this x1 x2 x3 xn they are computed at K plus 1 th iteration.

So all the values here of the previous iteration are used in the right hand side to compute the new values what is the idea of the Gauss Seidel method that newly computed components. So whenever we have computed for instance here x1 this x1 should be used there instead of the values from the previous iteration when x1 x2 is available so then we should use this x1 x2 to get x3 for instance.

This is the idea of the Gauss Seidel, so this is scheme basically becomes that whenever j is less than i will be using this computed values and j greater than i we will all obviously use from the previous iterations so that is the only difference in this two. Here the K th values are used here k plus 1 at so newly computed values of x are used in the algorithm.

(Refer Slide Time: 33:02)





Gauss Seidel Method

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k)}\right)$$

The algorithm in matrix form

$$x^{(k+1)} = D^{-1}\left(b - Lx^{(k+1)} - Ux^{(k)}\right)$$

$$\Rightarrow (D+L)x^{(k+1)} = \left(b - Ux^{(k)}\right)$$

$$x^{(k+1)} = -(D+L)^{-1}Ux^{(k)} + (D+L)^{-1}b$$

Gauss Seidel Method

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k)}\right)$$

The algorithm in matrix form

$$x^{(k+1)} = D^{-1}\left(b - Lx^{(k+1)} - Ux^{(k)}\right)$$

$$\Rightarrow (D+L)x^{(k+1)} = \left(b - Ux^{(k)}\right)$$

$$x^{(k+1)} = -(D+L)^{-1}Ux^{(k)} + (D+L)^{-1}b$$

NPTEL IIT Kharagpur

Gauss Seidel Method

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k)}\right)$$

The algorithm in matrix form

$$x^{(k+1)} = D^{-1}\left(b - Lx^{(k+1)} - Ux^{(k)}\right)$$

$$\Rightarrow (D+L)x^{(k+1)} = \left(b - Ux^{(k)}\right)$$

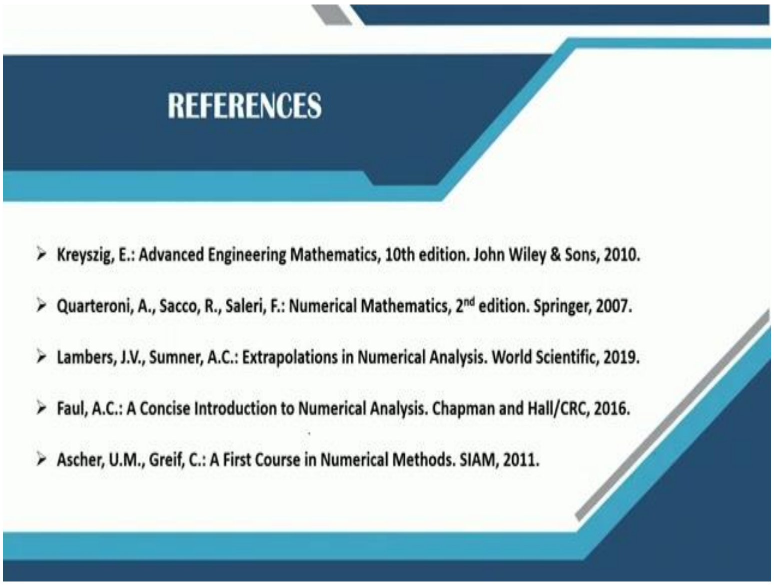
$$x^{(k+1)} = -(D+L)^{-1}Ux^{(k)} + (D+L)^{-1}b$$

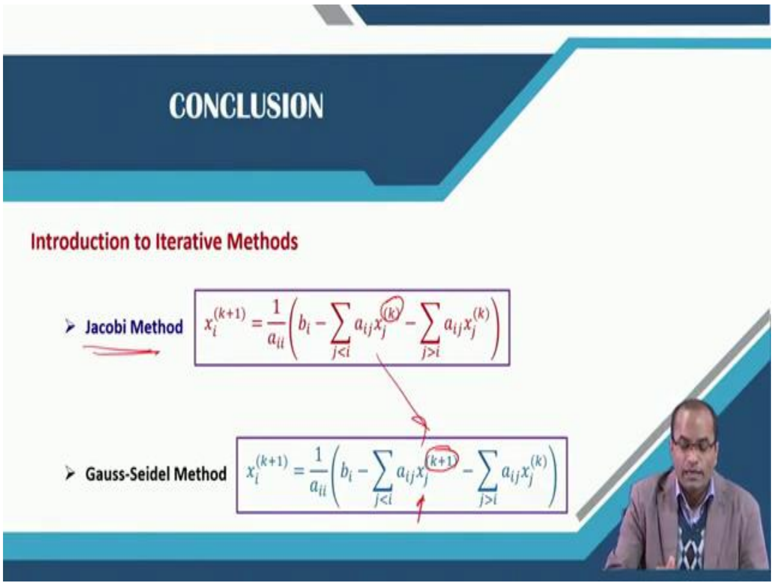
NPTEL IIT Kharagpur

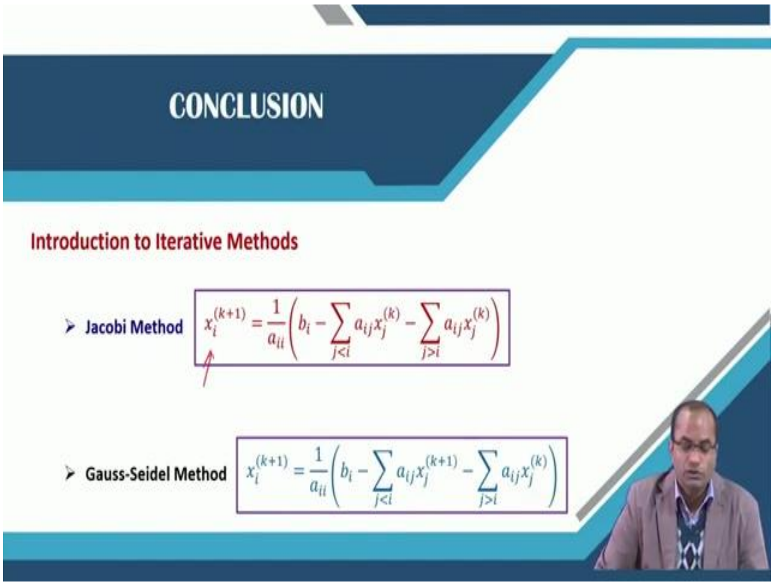
So we have this is the Gauss Seidel method and in matrix form again we can write this so this will free for to compensate this D1 then we have this for b and then here this is the lower part of the matrix is used here L x k plus 1 and there you have Ux the values are used here at K th iteration. And now we can just do a bit more simplification, so here the L can go to the left hand side so we have D plus L so first we will multiply by D there.

So we have Dx and here plus L x is equal to this one and then we can invert it so finally in the matrix form we have this algorithm that x K plus 1 is equal to minus D plus L inverse and the upper triangular matrix x K and D plus L minus 1 b. So there is only slight difference in the Gauss Seidel method than the Jacobi method that in Gauss Seidel we are using the recently computed values and as a result this matrix form takes this form where D plus L inverse U and here D plus L inverse b is coming. So this is the iteration matrix for the Gauss Seidel method.

(Refer Slide Time: 34:31)







Well so these are the references we have used for preparing this lecture and in this lecture we have introduce the idea of the iterative methods and in particular we have derived the Jacobi method and the Gauss Seidel method and they slightly differ from each other at this expression here. So the values of x recently evaluated values are used in this case all the values from the previous iterations have been used for the computation of the new values. Well so that is all for this lecture and I thank you for your attention.