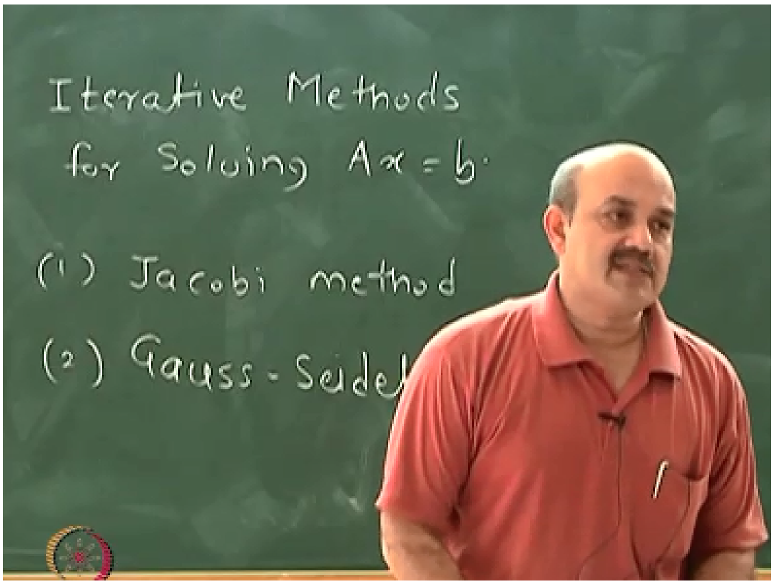
Advanced Numerical Analysis
Prof. Sachin Patwardhan
Department of Chemical Engineering
Indian Institute of Technology – Bombay

Lecture - 27
Iterative Methods for Solving Linear Algebraic Equations

So in the last lecture, we were looking at iterative methods for solving linear algebraic equations and in particular we looked at 2 different methods. One was Gauss-Seidel method another was Jacobi method.

(Refer Slide Time: 00:43)

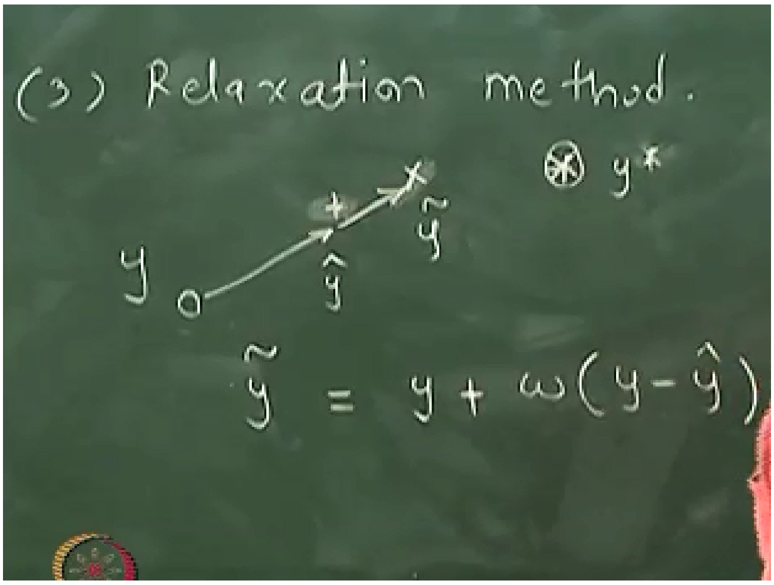


I want to solve linear algebraic equation Ax=b. We looked at 2 methods one was Jacobi method other was Gauss-Seidel method and I want to look at one more variations of these 2 methods. In fact, there are many variations. I am just going to indicate few of them, but this is over relaxation or relaxation method is slightly conceptually different from these 2 in the sense that you are trying to introduce a tuning parameter which will accelerate convergence.

So what we are really going to look at is what I told you that in general Jacobi method is slower than the Gauss-Seidel method. Gauss-Seidel method the difference is that as and when a new iterate is formed we use that in the next calculations that is a difference between Gauss-Seidel and Jacobi method. So Gauss-Seidel method actually tends to converges much faster than the Jacobi method.

Now I want to see whether I can do still better, can I still further enhance the convergence?

(Refer Slide Time: 02:19)



So this relaxation method this is a variant of these iterative schemes. Basic idea is like this. Suppose I am at this point y let us call this point y and I want to reach y *. I want to reach this point y *. I am currently at this point y and to reach from here to here I apply some method. So let us say my method takes me from here to this point which is y cap. So I will denote this point as+.

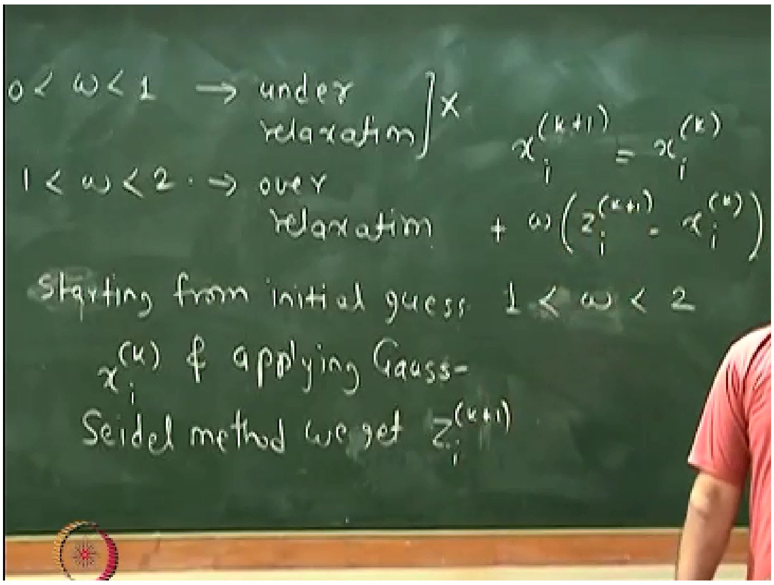
I am at initial point y. I have some method by which I am going to reach y *. Y* is where I want to go. And I have a method of going from here to here. Then I apply this method I go from y to y cap. Now if going from y to y cap is taking me closer to y *. I could move more in this direction and then go closer even closer try to go even closer. So what I can do is I can go to a point y tilde which is y+ omega times y-y cap.

So y cap is where my first method takes me. Y tilde is where if I enhance movement in this direction will take me. So let us say this is my y tilde. So I am going to move further in the same direction. So I am going to enhance movement in that direction. So my next iterate is not going to be y cap, but I am going to have my next iterate as y tilde which is obtained by moving further in a direction.

So the idea here is like this that if I start using Gauss-Seidel method I will get let say I am at y through Gauss-Seidel I reached y cap. So instead of taking y cap as my next iterate what I could do is I can look at this direction y-y cap and then you know multiply that and add that to y. So I will reach a point which is even further even probably closer to y *. This is the basic idea in this relaxation business.

There are 2 kinds of relaxation One relaxation is over relaxation so we are going further in some cases it is under relaxation. Sometimes you know this direction in which you are going might be too far away from y*. You may have to contract you may have to come back.

(Refer Slide Time: 05:43)



So there are 2 possibilities. One is omega is < 1 so this is and well there should be some limit in this case it turns out to be for linear system it turns out to be 2. So this is over relaxation. So basically what I am going to do here is something like this. Well I am not going to use in the present case under relaxation I am going to use over relaxation. I am going to have omega which is>1.

So see I am starting from initial guess xk and applying Gauss-Seidel method. Let us say we get. See xik I am now talking about the ith element of the vector. From xik if I apply Gauss-Seidel method let us say I get Zi k+1. I am not calling this x again. This is my intermediate variable because this is not going to be next iterate. What I am going to do is I am going to come up with the next literate xk+1 by enhancing the movement in this direction.

So my xik+1. This is going to be xik+ omega times Zi k+1-xik where omega is>1. If omega is=1 we are not interested in that case because we will reduce to Gauss-Seidel method, I want to enhance. I want to go further than what Gauss-Seidel method is giving me. Just try to understand this. What I am going to do is I am going to start applying Gauss-Seidel method to find a new point, but the new point I am going to take it as an intermediate point.

This new point which I get through Gauss-Seidel I am going to take an intermediate point. And the next iteration will be calculated using old guess+ this correction. Now this correction is developed by enhancing or moving further in the direction of the Gauss-Seidel. This is Gauss-Seidel step. This is the original point so difference between this will give you direction in which you have to have to move and omega times that.

So this is why I am going to generate the new iterate. What happens if I put omega=1? It will be Gauss-Seidel. If omega=1 it is just Gauss-Seidel method if omega is>1 it is over relaxation. You are moving further than that and the hope is that if I do this I will reach my solution faster. So in general you can even make it move faster to the solution if you choose this omega between 1 and 2.

Now the question that naturally comes is well how to choose this omega and when will it converge and so on. So we will of course in a due course I will answer all those questions, but is the idea clear. We just want to enhance the direction in which we are moving that is Gauss-Seidel direction and move a little further so that maybe we will reach the target faster, yeah but when you are reaching covers the solution this will be also small.

So there might be some oscillations it depends upon whether it will cause oscillations or whether it will cause a smooth decay will depend upon eigenvalues and I am going to talk about that. So that is a good guess. You may have a problem when we are close to solution you might overshoot so that may happen. So we have to understand how the convergence occurs.

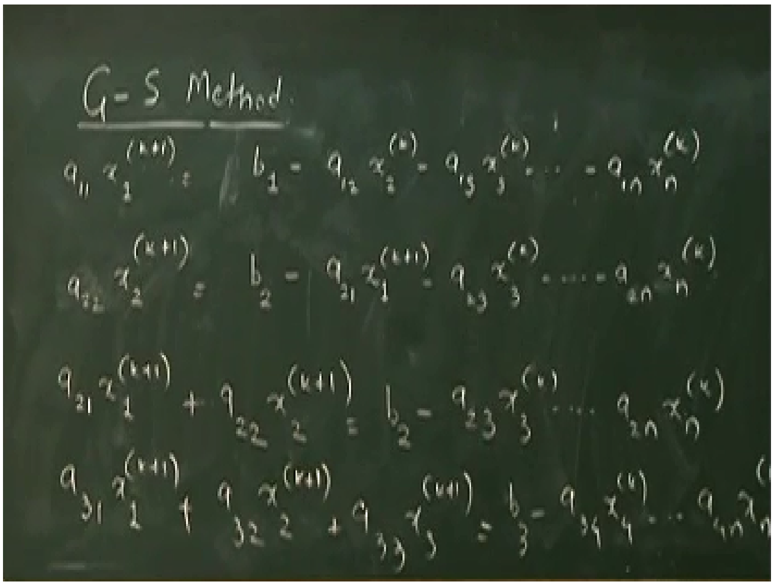
So the actual details you can see here the algorithm I have given here how to implement. The idea is clear algorithm will not be difficult to understand. Now let us start getting into analysis. Implementation part will be covered when you do programming and I am not worried about implementation part or the algorithm. How will you efficiently implement this is given me the table sin the notes and you can have a look at that.

The critical part is convergence analysis. So what I am going to do now is rearrange my equations in the matrix vector form. Now this will give me a way to analyze the system of equations or the convergence behavior that is what I want to analyze. The vector matrix notations which I am going to develop here. I am going to use maybe for next 2 or 3 lectures is not meant for programming.

Programming will be done row by row you put a for-loop and each row you do calculations that is how you do the programming first. The analysis which I am going to do is going to be using matrices and vectors and that is mainly because analyzing convergence becomes very easy using vectors matrix notations. So do not confuse the 2 things. Whatever we did till now was meant for actually programming.

Now what I am going to do is for analysis because getting insights into the analysis of these method using summations becomes very, very difficult instead of that if you do everything using matrix notation it is very easy.

(Refer Slide Time: 12:45)



So let us get on to the Gauss-Seidel method. So what is my Gauss-Seidel method. My first equation in the Gauss-Seidel method. I can write as a11 x1 k+1=b1-summation i going from 2 to n or j going from 2 to n let say. a1j right or let us write it more explicitly. Let us write this instead of summations it will be better to write this explicitly. So this will be a12 x2k-a13 x3k, a1n xnk. This is my first equation in the Gauss-Seidel method.

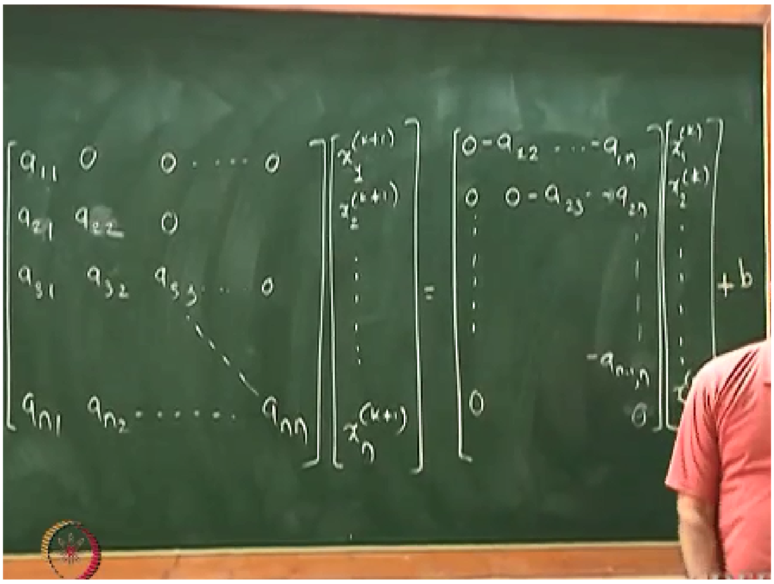
Earlier when I wrote the algorithm I had divided by a11. Now my intention is different I want to get some convenient vector matrix notations so I have multiplied by a11 so this is the equation which I have to solve. My second equation here is actually a22 x2 k+1=b2-a21 x1 k+1-a 23 is this fine this is my second equation. The iteration in Gauss-Seidel method. This is

Gauss-Seidel method.

The first iterate that was formed is used here. The first iterate that was formed is immediately used here. So I am going to rearrange this I am going to take this on this side. So I will get a21 x1 k+1+a22 x2 k+1=b2 a23 x3k everyone with me on this. I have taken k+1 on one side and I have side k on one side. I have left all xk on one side. I am taking k+1 on one side now clear.

I have just taken this which was substituted on right hand to left hand side. Likewise, I can go on doing this so what will be the third one can you guess a31 x1 k+1+a32 x2 k+1+a33 x3k+1=v3-a34 x4k up to a4n xnk. I am taking x k+1 on the left hand side. I am leaving everything that is xk on the right hand side. Now I want to use put this into a vector and matrix notation. It is going to be a long equation.

(Refer Slide Time: 17:27)

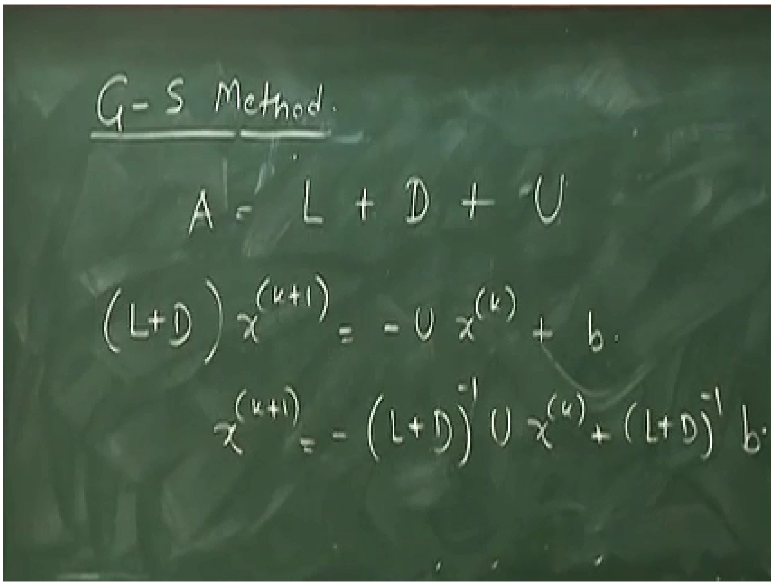


So this here is a11 a21 a22 do you see this. On the left hand side is the lower triangular part of A matrix=lower triangular part and the diagonal part. What will I get here 0 a12 up to a1n 0, 0 a23 a2n. This will be last element here will be 0 and this will be a n-1 n. This * x1k x2k xnk+ what will be on this side? Well I have made a mistake one mistake it should be – here this should be -, -.all this is -signs will appear here because we have all – signs coming there.

And here will be b vector if you see it is b1, b2, b3, b4. The b vector will appear here. This matrix appears here; this matrix appears here. What is the relationship of these matrices to original matrix a. What LU decomposition? LU decomposition is completely different do not mistake LU decomposition or LDU (()) (20:24) transpose that is not this. This is physically the lower triangular and the diagonal part of the matrix.

This is physically the upper triangle part –lower triangular part. Pardon me. You add it. You add it so this is if I write this matrix if I decide to write A matrix as sum of 3 matrices.

(Refer Slide Time: 21:06)



See if I decide to write A matrix as sum of 3 matrices L+D+U do not confuse this LDU transpose. LU decomposition is different from this. I am just using the notation L and U here. This is not LU decomposition. This is I mean if I draw a picture this is splitting a matrix into 3 parts. This is diagonal this is strictly upper triangular part of this matrix and this is strictly lower triangular part of the matrix.

This is diagonal elements this is all elements that are in the okay this is my U this is my L and this is my D. So writing this matrix as if I just do pictorially I am just writing this matrix as addition of 3 matrices strictly lower triangular part. The diagonal part and strictly upper triangular part. I am just using the same notation as LU decomposition this is not same as A=LU that is multiplication that comes through Gauss elimination.

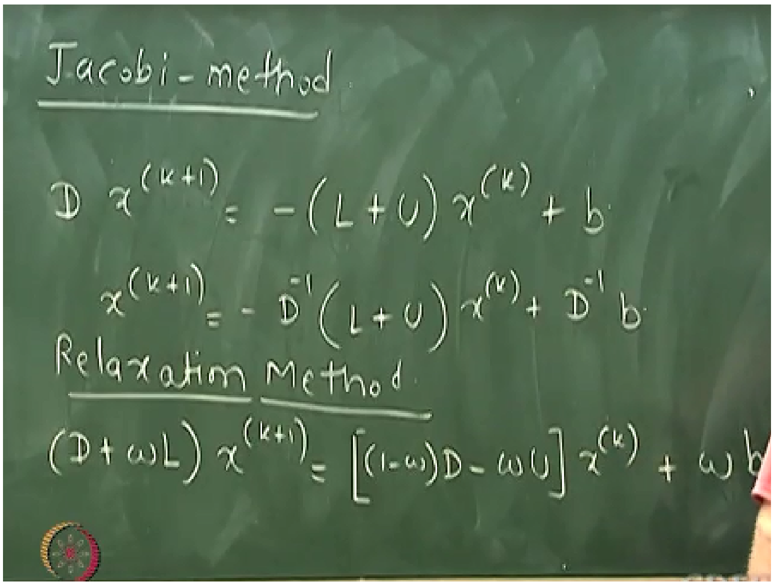
This is just simply separating 3 physically separate parts so this and this do not confuse I am just using same notation that is all or also you might see in some books or (()) (22:41) you also have this sometimes you write A as LDU transpose so do not confuse that and these only notations are same. So I have written it like this so this makes me or this particular way of writing this matrix A allows me to do express this method Gauss-Seidel method in a vector matrix notation.

So this equation which I have written in matrix form I am going to rewrite this as L+D *x k+1= -U xk+ b (()) (23:36) just look at the matrices. This is L+D on the left hand slide multiplying xk+1. Here it is U strictly upper triangular part of A matrix. A is written as L+D+U and then I can write this method next iterate is obtained by solving this matrix equation is equivalent to solving this matrix equation.

Actually we are going to do line by line as I told you. Algorithmically we are not going to write this matrix equation ever. We are going to do line by line calculations, but what you are doing line by line is actually equivalent to this calculation is that okay. So which means my next iterate is obtained as x k+1= -L+D inverse U xk+. Conceptually what you are doing is obtaining a new iterate by this method.

Sometimes I talked about inverting the matrix or approximately inverting the matrix and so on. If you look here we are trying to say well I cannot invert the full matrix easily, but I can invert the lower triangular part. It has a nice structure I can exploit that and invert that to come up with a new iterate. If I do a similar rearrangement of equations in the case of Jacobi method then what will I get if I do this is for Gauss-Seidel method.

(Refer Slide Time: 25:34)



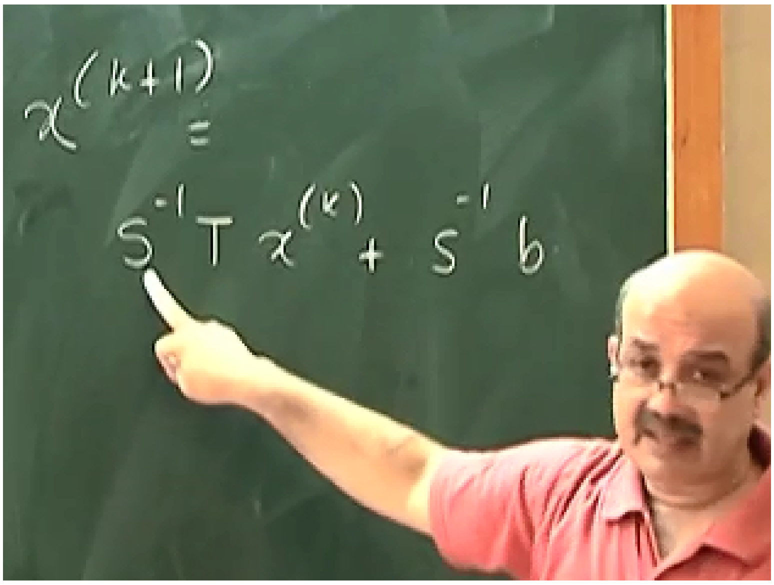
For Jacobi method what you can show is that doing the iterations is equivalent to D times x K+1= -L+U xk+ b x k+1=-D inverse L+U xk+ D inverse times b. For Jacobi method if you do rearrangement of the calculations you can show actually doing one Jacobi step is equivalent to solving this equation. What is nice about Jacobi method inverting the diagonal matrix is very easy just write one upon the diagonal element.

That is why we chose to do this so this is very, very easy as compared to inverting the whole matrix. If all the diagonal elements are non 0 if you arrange the matrix in such a way that all the diagonal elements are non 0 then doing this is very, very easy. So Jacobi method is equivalent to doing this. Now what about relaxation method. Relaxation method you can show that you have to do a little more algebra to come up with these matrices for the relaxation method.

But nevertheless finally you get this equation D+ omega L. For relaxation method it turns out that it is this matrix right hand side is this and solving this equation iteratively at each point. So likewise there are other variations what I have taught you is called as actually Forward Gauss-Seidel method there is also Reverse Gauss Seidel method then there is a Reverse or Backward Gauss Seidel method.

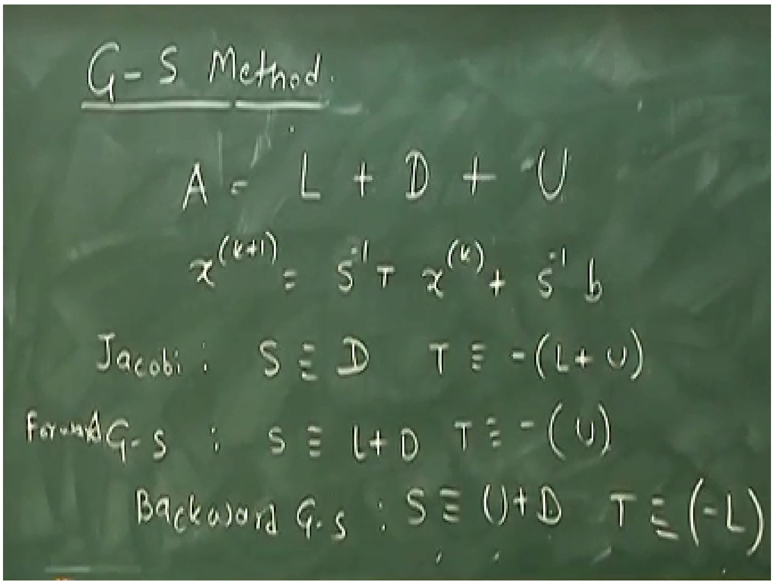
Then there is a symmetric Gauss-Seidel method in which you first do Forward Gauss-Seidel method then you Backward Gauss-Seidel method so there are all kinds of variations which makes convergence faster. So all of them you can actually express iterations in terms of these fundamental 3 components L D U and so I have listed some of these variations here. Now let me generalize this.

So in general if I look at this equation I can have a pattern here what is this pattern?
(Refer Slide Time: 28:45)



This pattern is like this and every time constructing a new iteration by following formula S inverse T x k +S inverse b. What is S matrix, what is T matrix changes from method to method, but essentially my equation looks like this whichever iteration method I take I will take Gauss-Seidel what is S in Gauss Seidel L+D. What is T here –U.

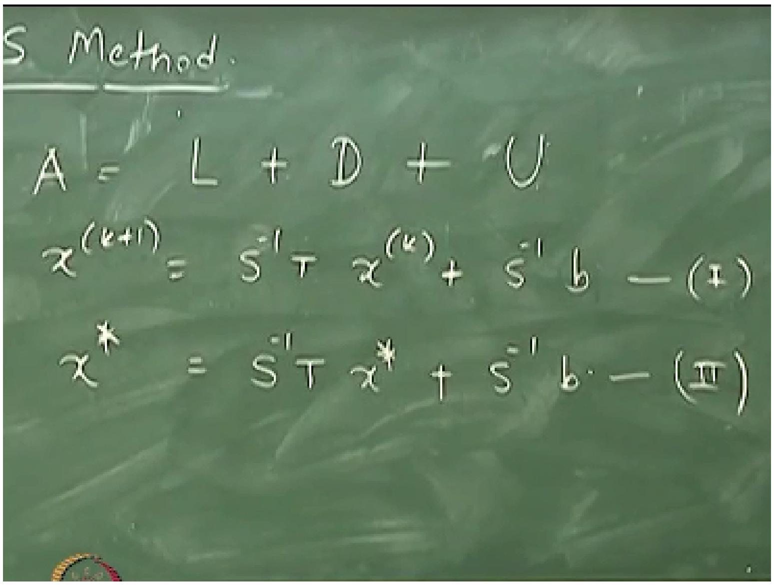
(Refer Slide Time: 29:35)



So I can actually list this. So my basic fundamental iteration equation is x k+1= S inverse T x k+ S inverse b. Now Jacobi method S corresponds to D and T corresponds to –L+U. Gauss Seidel method S corresponds to L+D T corresponds to –U can you guess this is Forward. Can you guess what will be Backward Gauss-Seidel. See here it is L+D what will be Backward Gauss Seidel U+D and here it will be –L.

Backward Gauss-Seidel will be S corresponds to U+D and T corresponds to –L. In this case relaxation method what is S? This will be S matrix, this will be T matrix actually you will have to probably divide by omega and then get that. To divide everything by omega then you will get S and T matrices. If you divide everything by omega you will get S matrix T matrix and then same idea. The fundamental equation is this.

(Refer Slide Time: 31:35)



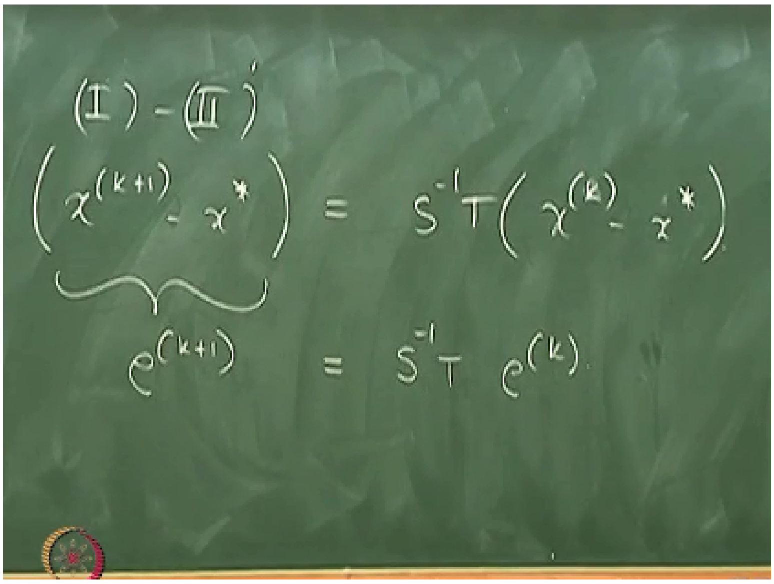
My fundamental equation is this. So basically what I need to do is I need to analyze behavior of this equation. How does this equation behavior? What is this equation have you seen this kind of equation? New value=matrix* old value has you seen this kind of equations these are called as linear difference equations. New iterate= matrix* old iterate. This is linear difference equation except you might have seen this in time.

Here this equation with respect to that iteration index of iteration. So if I put it in some standard form which I am going to do soon then things will fall in place, but you should be able to connect things what you have done some abstract form and when it is going to be applied somewhere. Now let us say these iterations converge. We still have not analyzed whether they converge or not, but let say they converge and say x * is the solution.

Let us say my x* is the solution. So what will be the final equation? What do you mean by converge, what will happen if this converges? It will give itself right. The final value should give itself. So x *= S inverse T x*+S inverse b. Is this fine. So finally this is where I want to reach. Let me call this as equation 1, let me call this as equation 2. I am going to subtract this equation from this equation can I do that.

What will cancel this term will be 0.

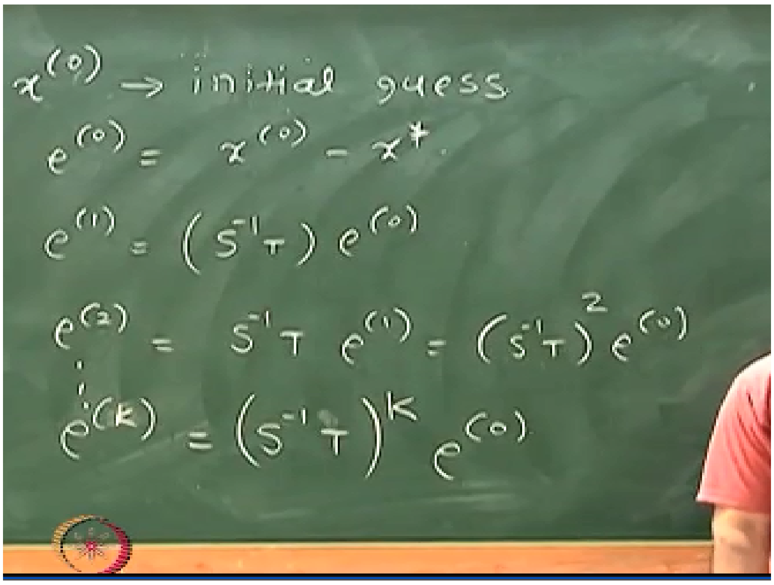
(Refer Slide Time: 34:12)



So if I subtract 2 from 1 I get this equation x k+1-x *= S inverse T x k-x*. Let me call this as error is this fine. This is new error. What is error? Error is the distance from the true solution. X* is the true solution well I will show you that x* actually indeed is a true solution. When you do this method we have to show that we should reach finally the point which is solution of Ax=b that also has to be shown. We will do that of course.

Is everyone with me on this? This is a linear difference equation and then finally I want to reach the solution of Ax=b. So far so good. So now let start analyzing this equation. So we started from those complex row summations. Now we have come to a very compact nice form everything looks like just you know one simple equation x e k+1. How does this equation behave?

(Refer Slide Time: 36:19)



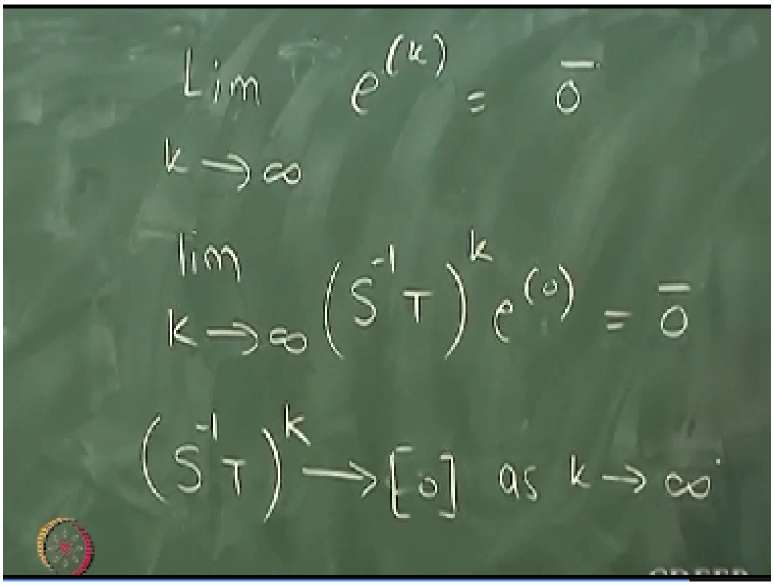
Let us say I start with some initial guess say x0 is my initial guess. What is the error at time x0? It is x*-e0=x0-x*. What is this error I do not know if I knew then I would know x*. I do not know what is this error? Nevertheless, I can do analysis without requiring to know the error. I am going to analyze the behavior of this equation the beauty of this analysis is that you can analyze without requiring to know what is e0 or what is e1?

I am just going to look at properties of S inverse T. If S inverse T has certain property and I am guaranteed that this sequence of errors generated by this difference equation will go to 0. If error goes to 0 I am reaching the solution. What is the meaning of error going to 0 difference between the iteration and the true is reducing that is what I want to happen. So let us start applying this equation.

So what is e1 S inverse T e0. What is e2? S inverse T even which is S inverse T square e0 this is okay. I am just substituting for even. What can you say about the error at instant K at ek can I write this is S inverse T raise to k can I write this? Well while denoting vectors this e is a vector by the way. X is a vector x* is a vector. So x0-x* is a vector e is a vector. This notation of superscript in the brackets is intentional you should not confuse it with raise to something.

This is not e raise to 0 this is 0th vector. So in the iterations at any point k in the iterations I have to now analyze what happens to this matrix when it is multiplied with itself multiple times.

(Refer Slide Time: 39:33)

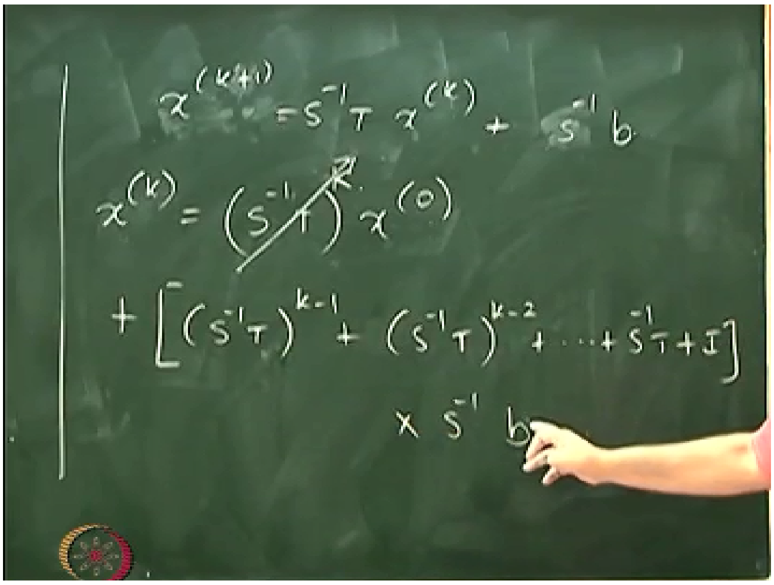


So what should happen now is do you agree with me. What I want to happen is this limit as k tends to infinity. If I do more and more iterations I go closer and closer to the solution that is what should happen. So which means limit as k tends to infinity S inverse T raise to k e0 should be 0 vector. Is e0 a 0 vector? No it is not a 0 vector. So what I want to happen actually is S inverse T raise to k should tend to 0 to null matrix as k tends to infinity. What should happen is S inverse T.

So I should choose S inverse T is such a way that this condition holds looks quite formidable how do I choose a splitting of a matrix in such a way that this condition holds. Well not that bad as it looks like we will look at the theoretical basis then it will be clear that it is not that difficult to do this. Let me first show that we are indeed going to go to the solution by doing this.

Error is going to go to 0 if error goes to 0 where will you reach? You will reach the solution. Actually if I apply the difference equation again and again

(Refer Slide Time: 41:30)



I can start with this difference equation xk=S inverse T or xk+1= S inverse T xk+S inverse b I can start with this difference equation apply it from 0, 1, 2, 3, 4 and I can show that actually xk= S inverse T x0. X0 is my initial guess. I am not deriving this you can derive this very easily what expression that I am writing finally just recursively write the equation and you will get what I am writing+ S inverse T k-1+S inverse T+2k-2 (()) (42:28) up to S inverse T+I* S inverse b.

If you use this difference equation again and again starting from x0 we will able to derive this expression. Now this expression has 2 components if you look here this expression has 2 components? What are the 2 components? One component is this S inverse T raise to K. I forgot to put K here. So this as S inverse T raise to K* x0. Now what we wanted to happen was S inverse T raise to K should go to null matrix.

So which means if this goes to null matrix this part will be nullified. If I chose S and T intelligently such that this condition holds then this part will be nullified. Where will the iteration go? Iteration will go to this. Now whatever I am writing here is this same as solving Ax=b we have to prove that. Are you getting what I am saying. See here I started with the argument that well look here I started with this argument here.

I said that error should be defined like this then there is difference equations that governs the error if I apply the difference equation actually I will get that error at time k will be S inverse T S 2 k error at time 0. Error will go to 0 provided this matrix goes close and close to null matrix. That is why next argument. This matrix should go close and close to null matrix.

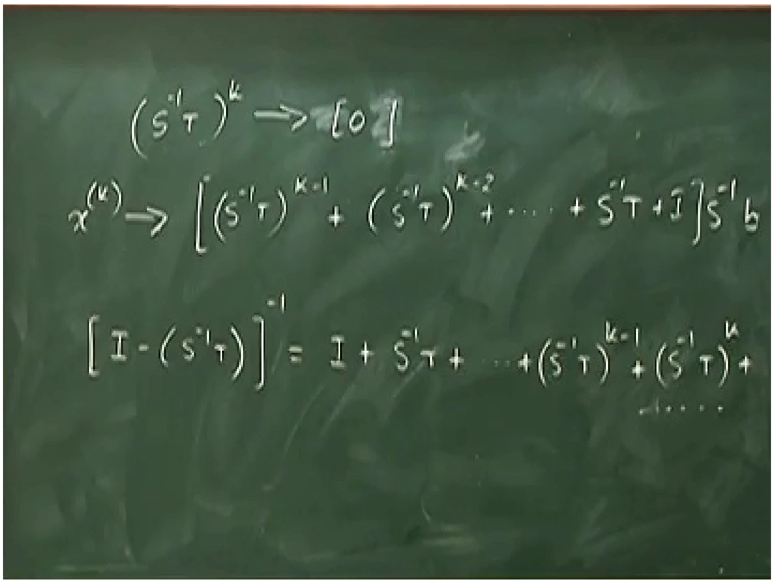
Then I looked at my original equation. This is my original equation iteration equation

I applied it again and again, again and again and then I could derive this it is very easy to derive this expression. You start with x0 x1= something x0 this then x1= or x2=S inverse T x1 and instead of x1 you substitute for you can eliminate and get everything in terms of x0 and then right hand side. It just repeated application of this equation starting from 0, 1, 2, 3, 4 you will get this expression not very difficult to derive.

Now if this part goes to 0 what it means is that as you progress in the iterations the effect of initial guess goes to 0 even if our initial guess is wrong if you have chosen S inverse T correctly this part will become 0. And whatever is your initial guess the solution will start going towards this vector on the right hand side.

So now what I have to show is that vector on the right hand side is indeed the solution that is first what I am going to show. Second what I want to do is to give you insights how do you choose S and T such that convergence is guaranteed that is my next mission.

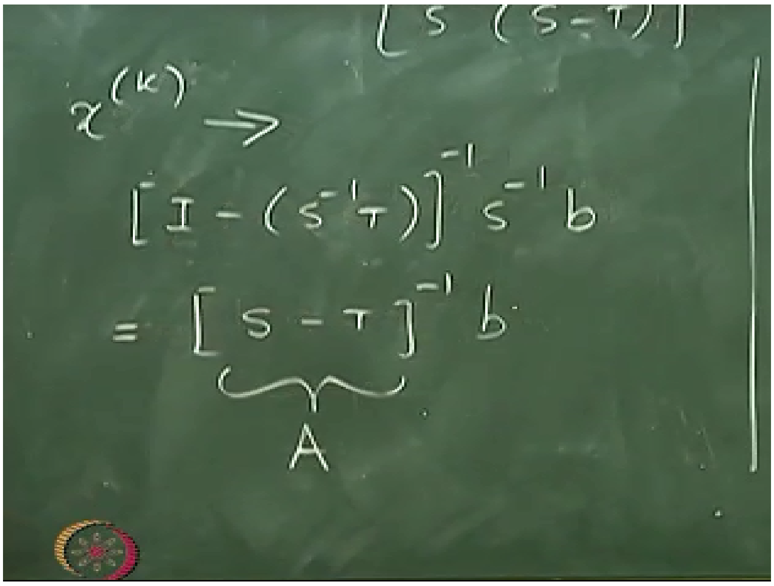
(Refer Slide Time: 46:15)



So my claim is that if this condition happens that is S inverse T raise to K if this goes to 0 null matrix if this is tending to null matrix. The solution that is xk tends to okay if this happens then iterations tends to this vector. I want to use one matrix identity. I suppose you have studied this in your first year or in your 12th standard I-A inverse can be written as a series. So this is actually a series where K is increasing.

So actually I can replace this by this. So which means if I use this particular series expansion.

(Refer Slide Time: 47:20)



I can say that actually xk is tending to I-S inverse T inverse S inverse b. So this S inverse is going to be there. I am just replacing this square bracket. I am replacing the square bracket by its limit as K tends to infinity this is the limit. This equation is as k tends to infinity as K tends to infinity I can replace this thing in the bracket by I-S inverse T whole to the inverse is everyone with me.

So what is this= can you expand this further. This can be very easily shown to be S-T inverse just expand this. You will get S-T you will take S inverse common you will get S. See this thing in the bracket can be written as S inverse * S-T whole thing inverse. What is S inverse S multiplied by S inverse will give you I. What remains is S inverse T inverse, but what is S-T? S-T is nothing A. So this is nothing, but just go back and see this is nothing but A.

So this sequence is going to converge to the true solution that is A inverse B this method of constructing sequence of iterates is actually equivalent to inverting A matrix and solving getting the solution provided you choose S and T correctly. What do you mean S and T correctly? S and T should be such that S inverse T raise to k should go to null matrix. So now the next part of the puzzle is that well under what conditions this will happen.

How do you choose S and T such that you will go to under what kind of matrices will look at class of matrices that will guarantee convergence then we will look at we will come to the point where we will be able to tweak these matrices using relaxation method? Basically what you may not expect right now or what is going to come is eigenvalue (()) (50:23). Now what I am going to do is relate convergence to eigenvalues of this matrix S inverse T.

And what I am going to show is that if eigenvalues of S inverse T are strictly < 1 then convergence will occur then the problem is transferred to how to choose S and T such that eigenvalues is of S inverse T are strictly < 1. The problem is transferred to how do I choose S and T such that. So we are going to relate this to eigenvalues a very, very fundamental and actually analyzing difference equations of this type is where the eigenvalues problem arises that is what I want to also highlight.

When we are taught in the linear algebra course eigenvalues problem where the teacher comes and write comes lambda Av= lambda V I mean why Av= lambda v. So you must have seen that eigenvalues problems pop out when you are trying to solve linear differential equations. I am going to show that it pops out when you try to solve linear difference equations.

Same thing happens when you are trying to solve linear partial differential equations you get Eigen function problems. Eigen functions and eigenvalues. Here you will get Eigen vectors and eigenvalues same idea used in different, different domains to understand how matrices behave without actually requiring to solve it. So the beauty of that analysis is that we will be able to talk about convergence without actually solving for it.

We can just look at the eigenvalues and say whether this will converge or not converge that is what we are going to go towards. So we will continue that with the next class. I will start with analyzing linear difference equations and their behavior and then go that to analysis of this equations.