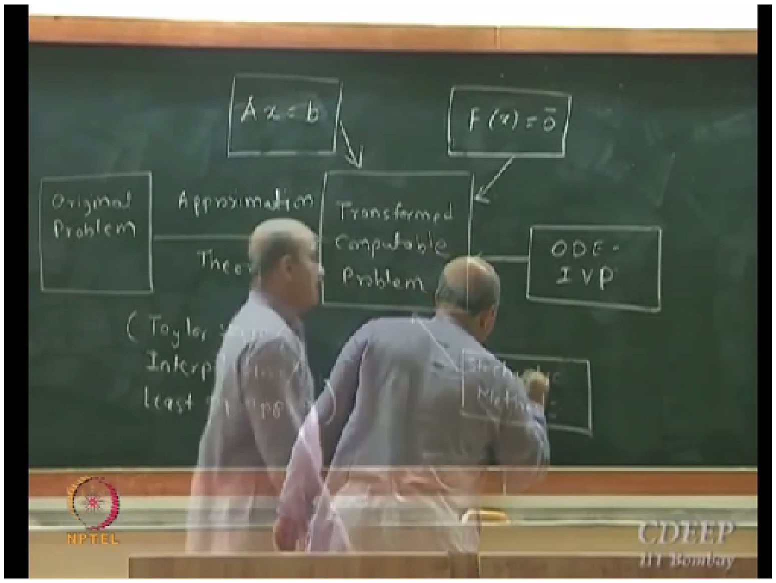
Advanced Numerical Analysis
Prof. Sachin Patwardhan
Department of Chemical Engineering
Indian Institute of Technology - Bombay

Lecture – 25
Solving Linear Algebraic Equations and Methods of Sparse Linear Systems

So till now, we have been looking at problem with discretization. We transformed the problem from original problem which was, let us say, a boundary value problem or a partial differential equation into a computable form, that computable form could be set of linear algebraic equations or non-linear algebraic equations. The transformed form could be ODE initial value problem and now we need ways of actually solving constructing the solution.

So I just want to draw the picture that we started with when, so we had this original problem.

(Refer Slide Time: 01:02)



This original problem was non-linear algebraic equations, boundary value problems, partial differential equations, all kinds of things that arise in modeling of engineering systems. There could be differential algebraic equations. So we have, we have different kinds of equations that we need to solve. So this is coming from your physics. This is coming from your modeling transport phenomenon, heat and mass transfer and so on.

So this will give you set of equations, most of the times, these equations are non-linear not at all amenable to constructing an unradical solution. These are in multiple dimensions and you have to solve them using some numerical techniques, okay. So the next step was problem approximation. So we constructed, we used concept of approximation theory and we got this transformed problem.

This is transformed computable problem. So in approximation theory, we used 3 different tools, one was Elsevier's approximation. Basically we used polynomial approximations. That was one of the fundamental tools but of course we used also functional approximations later. Then we talked about least squares, right. So we had, we used concepts of approximation theory to come up with a transformed problem.

So here either interpolation Elsevier's approximation or least squares, these were 3 basic ideas that were used to transform the problem. So I would just list them here, is Taylor series or interpolation. So Taylor series, interpolation or least square approximation, so these were 3 predominantly used tools to transform the problem to a computable form and now we want to attack this problem.

So we are, as you could see that this problem which you started with was a partial differential equation, you ended up here sometimes with non-linear algebraic equation, sometimes with initial value problem, ordinary differential equations, so you, the transformed problem, from the viewpoint of the structure of equations, could be completely different, okay. So it did not resemble the original problem.

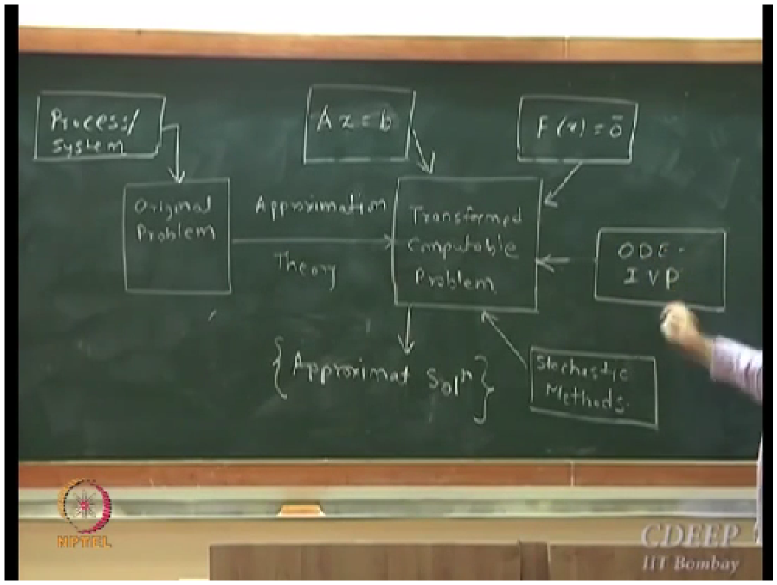
Original problem is a partial differential equation in space and time, here you get ordinary differential equation only in time. Original problem is partial differential equation in space, you just get algebraic equations, either linear or non-linear. So transformed problem is completely different. We hope that if we solve the transformed problem, we get something closure to true solution, not the exact solution. Now how do you attack the transformed problem? The next thing is look at the tools.

So I want to talk about 3 different tools, Ax=b. I would put this as solving linear algebraic equations. This is one of the fundamental tools that we are going to use. So this tool will be used attack this problem. Well the other one, other tool that I am interested is actually solving F of x=0. So this is another tool which is used to solve the transformed problem, okay. The third tool which I am going to be looking at is the ODE initial value problem.

Initial value problems are again a fundamental block and Euler method, Runge-Kutta method, all those methods will come under this. Here we will revisit Newton's method. We will look at its other new answers and see how you can enhance the convergence and when the fourth tool, which is quite commonly used, is stochastic methods but I am not going to deal with this, okay. Stochastic methods, this is a fourth tool.

What we expect after that, once we attack this problem with these tools or their combinations, it is not always that you will use only one tool, you might use this and this because (()) (06:54) will require this, okay, to solve this and so on. So there are combinations and finally what comes out is the approximate solution.

(Refer Slide Time: 07:04)



So this is my end result, this is my end result. So I start with the original problem, I used approximation theory, come up with a transformed problem which is computable. I have only 3-4 tools with me which I use ingeniously to concoct a solution. You should get this feel at the end of the course that, actually it is like a bhelpuri shop. You only have few things with you, okay and with the same things you can make savepuri, you can make dahibatata puri, you can make bhelpuri.

So it is the same thing, it is the same ingredients, okay and by just making mixtures, you can create different dishes. The same idea is here that we do not have too many tools, we do not understand how to solve very very complex equations exactly. We just know after all that we have done progressive mathematics, we just know how to solve Ax=b, we just know how to solve non-linear algebraic equations and all these you will realize, we know how to solve this approximately, we know how to solve this approximately.

So ultimately what you land up with is a third order approximation, you approximate from here to here, okay. Then you try to use these tools but they themselves are approximate, okay and then there are errors in computation because of inherit limitations of a computer. So all 3 combined together, okay, you get a third order approximation which you hope is close to reality what the real solution is, okay.

Well there is one more approximation which I forgot. If you start with a real system, okay. So this is the real process or a system. When you write a mathematical model, okay, that itself is approximation. When I say that a reactor is like a CSTR, it is not CSTR or when I say it is a PFR, it is not a perfect PFR. PFR and CSTR, these are models, these are idealised models which try to approximate the reality. So from here to here, itself there is an approximation. What we formulate here, we cannot solve exactly.

So there is one more level of approximation. Then when you try to attack them using different tools, those tools are also approximate, okay, because all these ideas of Taylor series, then polynomial approximation, interpolation, all these again, we will revisit when you are doing this, okay. So they are not going to leave us. Because again many of these will be not solvable, so you go back to again Taylor series, you go back to again and then you solve an approximation and so on.

So you could say it is a fourth order approximation and in between you know you have computer which is having its own limitations. So all things put together what you get is approximate solution. Now this approximate solution should represent something that is happening in this problem and the reality and this is where your inputs as engineer or a physicist or a chemist will come into picture. You should make, you should know whether this makes sense.

Often times you need to actually give a good guess. You must have realised this when you are solving some of the problems numerically and if you do not give a good guess, the solution can be absurd and so giving good guess is where your background in engineering or science is very very critical, okay. So now what we do is, we have come up to this point. Now we look at A, B and C. We will not be able to get into stochastic methods.

Hopefully, you will be able to cover and will be able to do some justice to these 3 tools, okay. So let us begin with Ax=b and then you might wonder well I have been doing this since my 12th standard, so what more to it. What more to Ax=b, I have been doing this for ages. Well lot more to it. I will at least need 8 or 10 lectures, that too we will just touch the tip of the iceberg. When you are using or when you are solving problems in school textbooks with some 2 equations and 2 unknowns or 3 equations and 3 unknowns.

You know simple methods work. Well the first thing that you learn is Cramer's rule, right. Now what I will show is that Cramer's rule beyond 4 equations becomes (()) (12:27), you cannot use it for computer implementations. The most practical method is of course Gaussian elimination, again (()) (12:37) prince of mathematics Gauss. This method is probably one of the most efficient methods of directly solving many of algebraic equations.

But then again these methods will be good if you have 1000 equations and 1000 unknowns. Once you start going into 10,000 and 20,000, 1,00,000 equations and 1,00,000 unknowns, you have problems, okay. You still get into roadblocks of time constraints. You can use iterative methods and we will talk about iterative methods. So iterative methods that you start with a guess solution and then try to converge to the close to the true solution as quickly as possible.

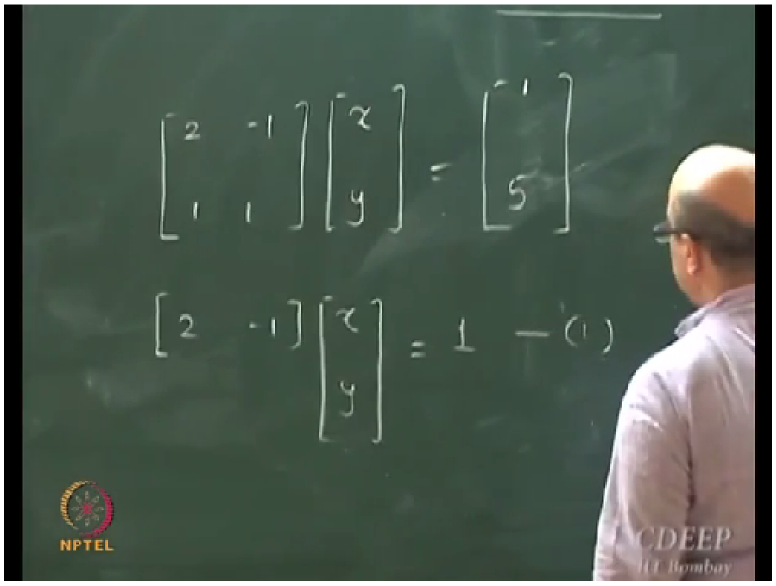
Again iterative methods, we have to look at when you are guessing and trying to go to a solution, you should know whether you will converge or not, okay. So under what conditions you will converge is going to be one of the main themes. Then also optimization based methods which are again interactive methods. I will brief you, touch upon optimization based methods and then finally, I will also talk about a fundamental thing, what is ill condition systems and well condition systems.

So how do you classify Ax=b. Well it depends upon this A matrix. So we will talk about properties of A matrix like a condition number in detail. I suppose you may have been introduced to this idea of condition number but I will derive the basic ideas that lead to a condition number and we will talk about well condition matrices, ill condition matrices. When the matrices are ill conditioned, you cannot get reliable solutions through numerical computations and you should be aware of that because now a days you have tools.

You know you just give a matrix; it will pop out a solution. You should know whether this solution makes sense of not, okay. One is from physics viewpoint, other is from computation viewpoint. If the matrix is ill conditioned, okay, you cannot get a good solution and you should know, should be aware of that, okay. So let us begin by talking about existence of solutions. When does the solutions exist? B lies in the column space of A, so there are 2 actually pictures, there are 2 geometric pictures that are typically used.

I will just use this; you should actually look at a book by Gilbert Strang. He used very nice introduction to solution of linear algebraic equation.

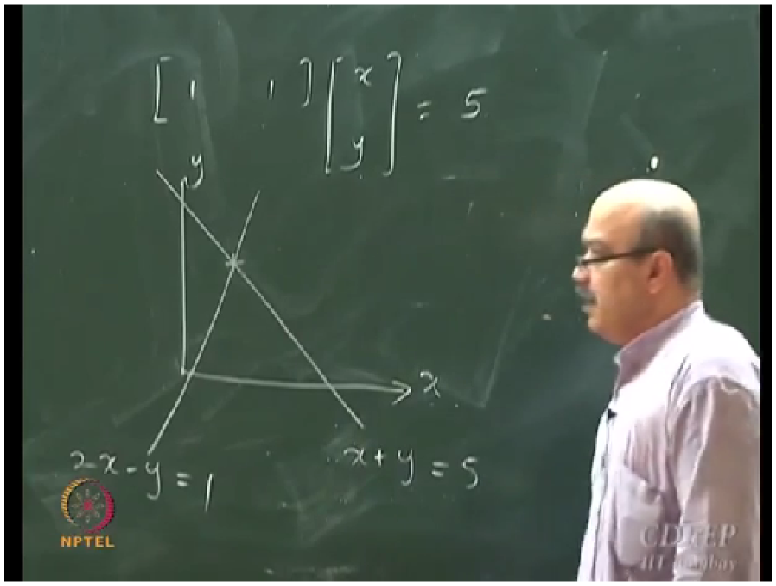
(Refer Slide Time: 15:16)



$\begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$
$\begin{bmatrix} 2 & -1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = 1 \quad -(1)$
NPTEL
CDEEP
IIT Bombay

This is example from Gilbert Strang, I am just taking 1 simple example which is 2 -1 1 1 x y=1 and 5, okay. Now there are 2 viewpoints by which you can geometrically visualize the solution. Well one viewpoint which is taught in our schools is, 2 lines intersecting or when it goes to 3 variables and 3 unknowns, we talk about 3 planes intersecting at 1 point and so on. In general, we can say that hyperplanes intersecting. So one viewpoint, I would say is this that is 2 -1 x y=1, this is equation number 1.

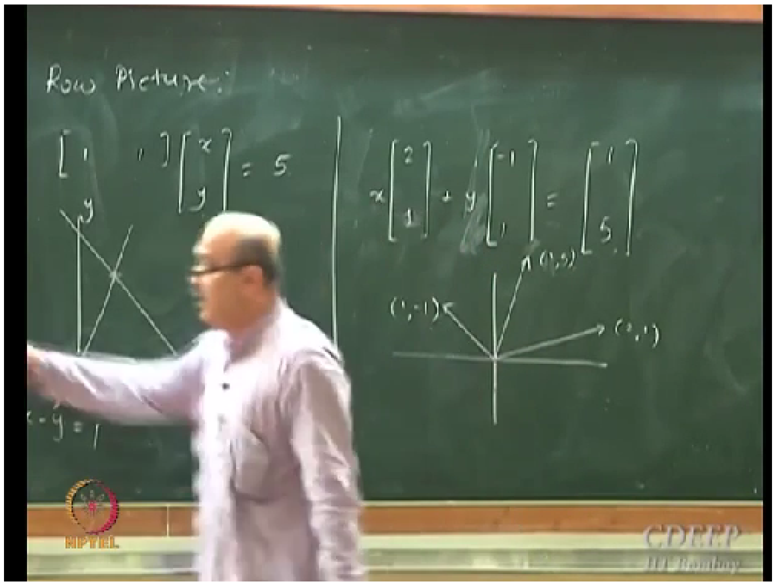
(Refer Slide Time: 16:13)



$\begin{bmatrix} 1 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = 5$
y
x
2x - y = 1
x + y = 5
NPTEL
CDEEP
IIT Bombay

and second equation is 1 1 x y=5, okay. So this is, we would say that well, so we know these 2 lines and this is my xy plane, this is very commonly given example. So this line is 2x-y=1 and the other line is x+y=5 and then this line and this line intersects at this point. This is the solution, okay. Commonly your example means not this particular example but intersecting lines is or intersecting planes. So this is the solution that is what we know. The other interpretation is of course the column space interpretation which is coming from vector addition.

(Refer Slide Time: 17:27)

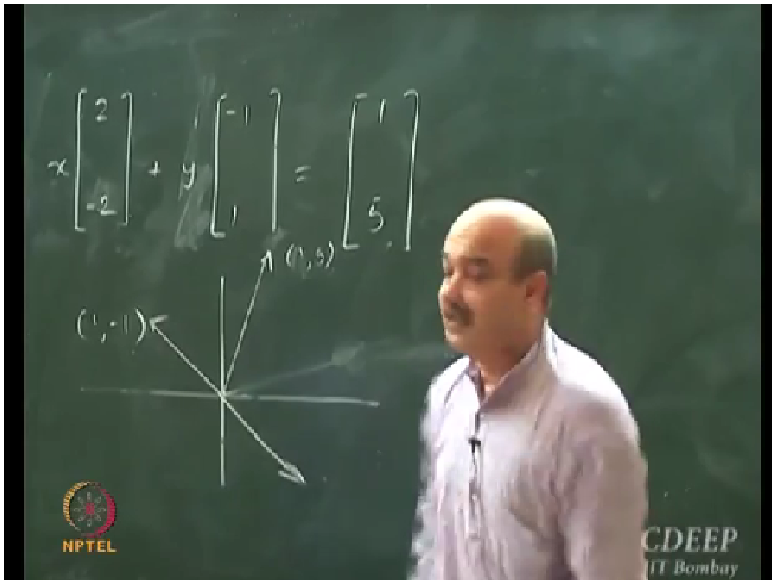

Row Picture:
$x\begin{bmatrix}2\\1\end{bmatrix}+y\begin{bmatrix}-1\\1\end{bmatrix}=\begin{bmatrix}1\\5\end{bmatrix}$
(1,5)
(1,-1)
(2,1)
CDEEP
IIT Bombay

So I could view this equation as 2 -1, sorry 2 1*x+y*-1 1=1 5, okay. This is the second equation and here, we are not drawing pictures in xy plane any longer. X and y are coefficients by which these 2 vectors add to give the third vector. So this is your vector addition. This is your vector addition, okay and this picture is, I will call this as, so this row picture. Row picture becomes very very difficult to visualize in some sense beyond 3 dimensions, okay but my claim is that the column picture is little easier to visualise after 3 dimensions, 4 dimensions, 5 dimensions.

We are just talking about linear combination of vectors. So here what we are saying is, we have these 2 vectors. We have this vector 1 -1 and I have this another vector. This is my vector 2 1, this is my vector 1 -1, okay and then my third vector is this 1 5. We are just trying to use now the law of vector additions, okay and then you can complete the geometric picture by drawing the parallelogram here.

So actually it is just linear combination of these 2 vectors, gives me the third vector, okay, gives me the third vector and when will you, when will you be able to get solution for any right hand side, if you want to specify any right hand side, okay. When the linear combination of the columns support should span the entire space. That means these 2, these 2 should be linearly independent. If these are linearly independent, all possible linear combinations of these 2 vectors will span all 2 and then any vector… Well, there will be a problem.

(Refer Slide Time: 20:15)

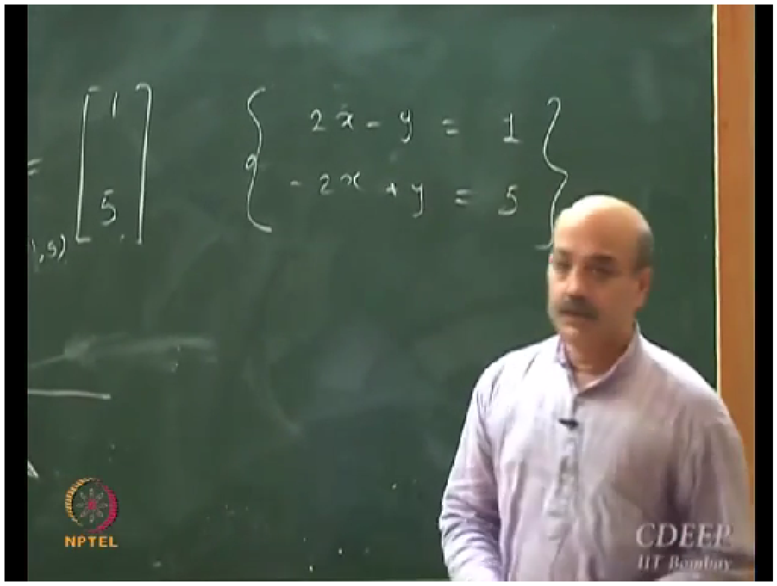


$x\begin{bmatrix}2\\-2\end{bmatrix}+y\begin{bmatrix}-1\\1\end{bmatrix}=\begin{bmatrix}1\\5\end{bmatrix}$
(1,5)
(1,-1)
NPTEL
CDEEP
IIT Bombay

If suppose if this was 2 -2. Suppose I change this equation to 2 -2, what is the trouble? These 2 are vectors in the same direction, okay. No linear combination of these 2 vectors can never give me 1 5, okay. So the column picture would say that equation does not have a solution, that is because you have one equation, you have 2 vectors which are aligned along the same direction.

There is no linear combination which is ever going to give me this vector, okay. No linear combination is ever going to give me this vector. So now when column picture tells you that there is no solution, same thing will happen for row picture. If you take row picture here, okay, for these change case, you will see that there are no 2 different lines. See we have parallel lines which will never meet.

(Refer Slide Time: 21:18)



So 2x-y=1 and -2x+y=5. So this will be the row picture and you will get 2 lines which never intersect, okay. So when this picture shows that there is a problem, you cannot get a solution, the row picture also will give you a problem, there is no solution. No hyperplanes meet or no planes or no lines meet, okay. We can generalize this to any n dimensional case. We can talk about linear combinations of column giving you a solution which lies in the column space, okay.

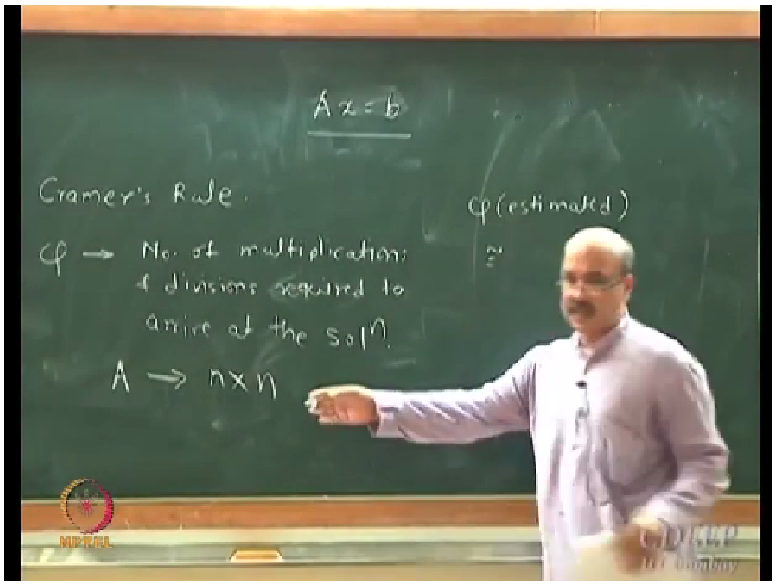
And row picture will be hyperplanes and then the solution when it exists is all the hyperplanes meet in one point, okay. So actually line is a 1-dimensional hyperplane in 2-dimensional space. Plane is a 2-dimensional hyperplane in 3-dimensional space and in 4-dimensional space, there will be a 3-dimensional hyperplane and son on and those 3-dimensional hyperplane should meet, well difficult to visualise how 3-dimensional planes meet at one point but if you look at the column picture and talk about linear combinations of column in n-dimensional spaces, okay.

So we will not get beyond a certain point here. What is important is, because all of you know this, I am just revising this very quickly. When does the solution exist, solution exist when the columns are linearly independent, okay and same thing is true about rows if the columns are linearly independent, the rows are linearly independent and number of fundamental theorem of linear algebra is, number of linearly independent rows equal to number of linearly independent columns is equal to rank of the matrix, okay.

So with this brief brush up, we will just get into the solving problems using… okay. Now Gaussian elimination is something which I am not going to touch upon because I assume that you already have a sufficient background of Gaussian elimination, I have uploaded some notes on Gaussian elimination. You should have a look at those notes. I just want to compare 2 things here to begin with, to give a motivation for methods which are more efficient.

One is number of computations that are number of multiplications and divisions that are involved.

(Refer Slide Time: 24:14)



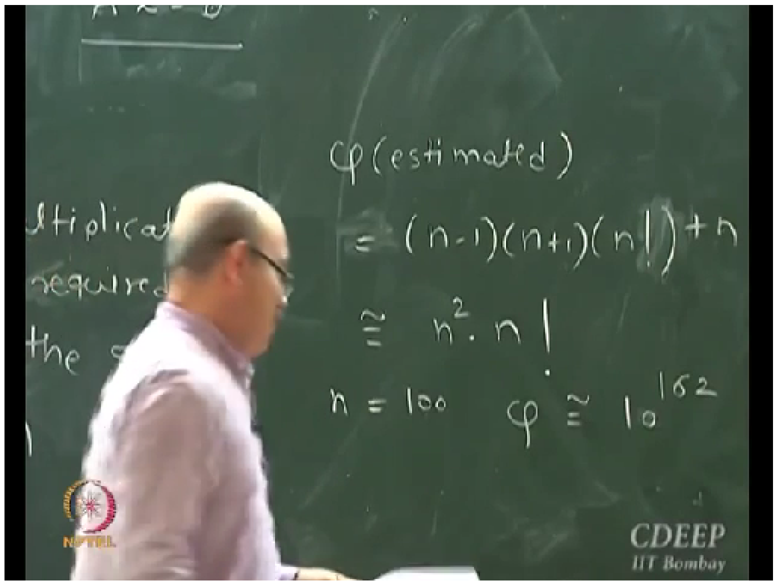
$Ax=b$
Cramer's Rule.
$\varphi \rightarrow$ No. of multiplications & divisions required to arrive at the soln.
$A \rightarrow n \times n$
φ (estimated)
$\simeq$

So 2 methods that you know for solving Ax=b. The first method you learn is Cremer's rule, okay. The first method that you learn is Cramer's rule. Now I am going to use this psi to denote number of … required to arrive at the solution. If psi represents the number of multiplications and divisions that are required to arrive at a solution, then for Cramer's rule, psi estimated is approximately equal to, I am talking about n*n matrix, so my concern here is A is in general and n*n matrix where n could be large, okay.

N could be 1000, 10,000 or whatever, a large number and we have seen that these kind of matrices arise when you do, let us say, finite difference method problem, discretization of boundary value problem. A 2-dimensional boundary value problem will give rise to, even if you take 100, 100 grid points, okay, along x and y. You will get a huge matrix which has to be solved.

So large matrices are not uncommon when you do approximate solutions.

(Refer Slide Time: 25:53)



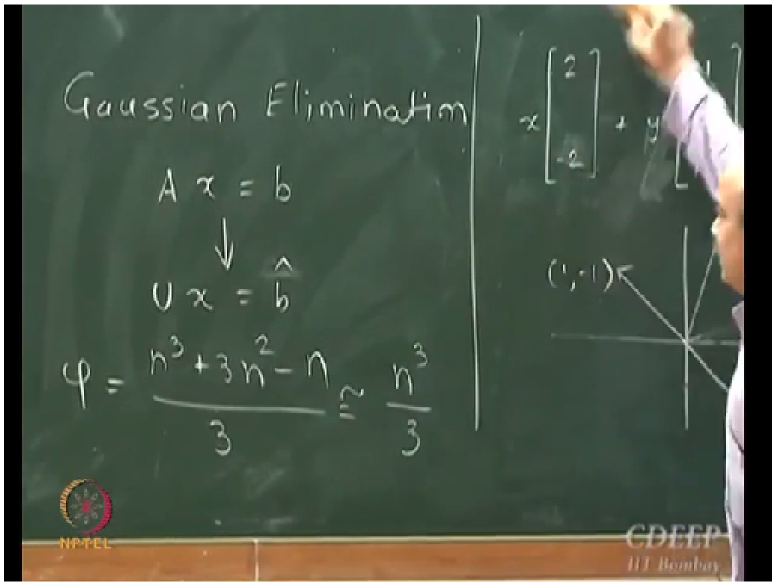
So here it comes out to be n-1*n+1*n factorial+n which is approximately, this is equal to, this is exactly equal to, which is approximately equal to n square*, this is n factorial+n, this is n square*n factorial, okay. Estimated number of operations, divisions and multiplications for Cramer's rule, I am not going to derive this, you just accept this number right now. I am just to give you estimate of what can happen if you try to use Cramer's rule for a matrix of moderate size, 100:100, okay.

If you take n=100, okay, this data I have taken from professor S. K. Gupta's book. So this is, you can show that this psi is close to 10 to the power 162, okay and he says that a tech computer, take 10 90, would take 10 to the power 149 years to solve this problem, okay. So absurd number, so to solve a problem of 100*100 matrix (()) (27:25) *100 matrices, using Cramer's rule is impossible, okay.

A computer would take, the DEC computer when he probably wrote the book was the fastest one and then he has given this number that it will take 10 to the power 149 years, okay. So Cramer's rule is ruled out. It is just good for 10th standard or 12th standard whatever, wherever you learn it and good for solving 3 equations and 3 unknowns, may be 4 equations. If you have patience to do determinants, then. So this is not the way to go, certainly not the way to go. What about

Gaussian elimination? Gaussian elimination seems to be the hope here.

(Refer Slide Time: 28:10)



So in Gaussian elimination, we start with Ax=b, then we transform it to an upper triangular matrix Ux=b prime or b cap, let us call it. We transform by elimination. We transform it to an upper triangular matrix and then we do the backward sweep, okay. So what is the number of operations that you need here? The number of operations that you need here is n cube+3n square-n/3 which is approximately for large n which is approximately equal to n cube/3. So n cube/3 is far far far less than n factorial.

So this is the reason why Gaussian elimination is so popular and is a compute savy method, is because this is one of the most efficient ways by which you can solve Ax=b, okay. So that is why it is introduced right at 10th standard or 12th standard and then you start building on it as you go into your higher grades, okay. So obviously we want to work with Gaussian elimination, okay and now what I want to do is, before I move to iterative methods, I want to see whether Gaussian elimination can be made more efficient, okay.

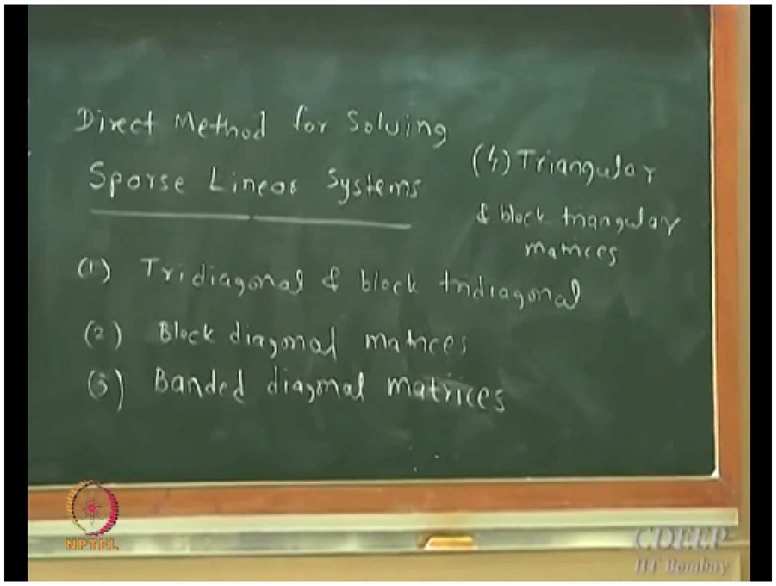
By exploiting certain structures that appear in discretization, problem discretization. So some time back when we read this discretization, I told you that finite difference or orthogonal collocations or orthogonal collocations on finite element and so on, we get some special matrices. These are called as sparse matrices. Sparse matrices are ones which are filled with large number of 0's and they may have some nice structures, okay. If you exploit these structures, okay, you can come up with direct methods.

So these methods which I am talking about, these are classified as direct methods, okay. When I say direct methods, as against iterative methods, iterative methods in which you start with the guess and from the old guess you construct a new guess and so on, okay. Like (()) (30:50) the iterative method. Direct methods in Gaussian elimination, we do not start with a guess, right. You directly solve the problem, okay.

Other variance of Gaussian elimination are Gauss-Jordon method, okay, LU decomposition. So all these are variance of Gaussian elimination but basically the foundation is Gaussian elimination, okay. Also I have uploaded something about LU decomposition. LU decomposition is Gaussian elimination represented as sequence of 2 triangular matrix calculations, one is upper triangular, other is a lower triangular.

So you can have LU decomposition also. We can have a look at LU decomposition but that is derived from Gaussian elimination. So the order of computations are again of the order of n cube/3. It is not 2 different, okay.

(Refer Slide Time: 31:50)



Direct Method for Solving Sparse Linear Systems

(1) Tridiagonal & block tridiagonal

(2) Block diagonal matrices

(3) Banded diagonal matrices

(4) Triangular & block triangular matrices

So in direct methods, now what we are going to look at is direct methods for solving, when I say direct method, not the Cramer's rule, of course the Gaussian elimination, okay. Sparse Linear Systems are those set of equations, Ax=b where matrix A has large number of 0's and I want to exploit special structure of that matrix A to come up with a efficient solution which will significantly relieves the number of multiplications and divisions that you need for solving the problem, okay, that is the motivation for these sparse methods.

Again in this course, I cannot do a justice to Sparse methods. I am just going to introduce you to sparse methods, okay. Everything is an iceberg and we just touch the tip of the iceberg. To look at sparse method, in metlab, there is a sparse matrix toolbox and it will list so many algorithms, I cannot cover all of them but what is important is to sensitize you that there exists something called sparse methods.

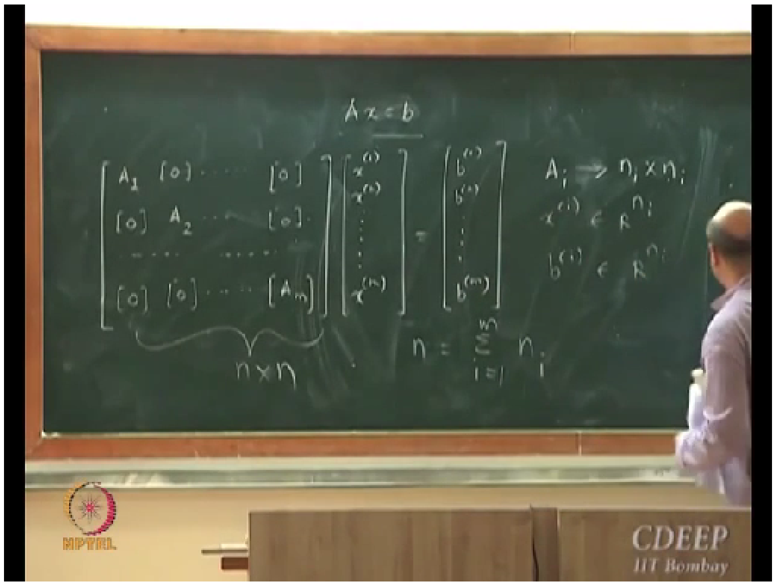
And you should look at, you should try to see whether there is a nice structure in your problem which can be exploited to relieves the computations, that is what is more important. So I am going to touch upon 3 or 4 methods and we will move on to something else but at least this will give you flavour of what it is, what is this sparse matrix business, okay. So we are going to look at somethings which are some matrices, some sparse matrices.

One is tridiagonal and block tridiagonal matrices. We will look at block diagonal matrices. We will look at, where in general one can talk about banded matrices. So banded matrices with only m diagonals, okay. I am not going to touch upon this. I will talk about these 2. Then I will also mention about triangular matrices, block triangular matrices. So these are some nice forms.

Actually if you start looking at this forms just in your text book, you might wonder where do I get all these forms but all these forms appear in problem discretization, okay and once we rise above these forms, we will realize where they appear and how you, so the motivation for these looking at this sparse matrices becomes clear if you look at the problem discretization methods, okay.

Those will give you these nicely patterned matrices which you can actually exploit to come up with efficient solutions. Let us look at first the simplest one, I would say is block diagonal matrix. A block diagonal matrix would appear in orthogonal collocations on finite elements or CFE. A block diagonal matrix would look something like this.

(Refer Slide Time: 35:30)



$Ax=b$
$\underbrace{\begin{bmatrix} A_1 & [0] & \cdots & [0] \\ [0] & A_2 & \cdots & [0] \\ \cdots & & \cdots & \\ [0] & [0] & \cdots & [A_m] \end{bmatrix}}_{n\times n} \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(m)} \end{bmatrix} = \begin{bmatrix} b^{(1)} \\ b^{(2)} \\ \vdots \\ b^{(m)} \end{bmatrix}$
$A_i \rightarrow n_i \times n_i$
$x^{(i)} \in R^{n_i}$
$b^{(i)} \in R^{n_i}$
$n = \sum_{i=1}^{m} n_i$
CDEEP
IIT Bombay

Well this original equation which is Ax=b, I am writing it in the block diagonal matrix form, okay. Here A1 A1 up to Am are matrices which are full. They are not, they do not have too many 0's. These matrices are typically dense matrices, not sparse matrices but all these square 0's are actually matrices of appropriate dimension which contain only 0's. So this is sparse matrix because non-0 elements appear along these diagonals and then here, this is the dense matrix, this is the dense matrix, this is the dense matrix, but all these are 0 matrices, null matrices.

So this is a huge matrix filled with large number of 0's only here on the diagonal, you have some small submatrices which are dense, okay. The small submatrices which are dense. What I have done here, x here, I have partitioned here, okay. I will talk about this partition very soon and even b vector, I am partitioning into b1 b2… okay. Now what are these partitions? Well my matrix Ai is actually a matrix which is ni*ni, okay. So this could, see this could be the 3*3 matrix, this could be a 5*5 matrix. Next one could be 3*3 matrix.

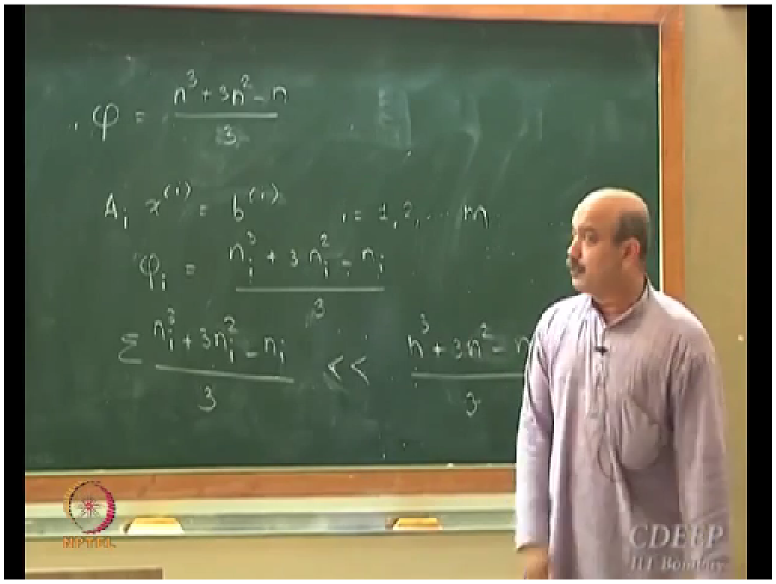
If you are doing orthogonal collocations on finite elements, on the first element, you take 3 internal collocation points. Next one you take 5 internal collocation points and so on. You will get this matrices of different dimensions. Only these matrices on the diagonal are dense. Because if you remember when you do orthogonal collocations on finite elements, only variables in that small segment appear in that equation, okay.

So you have this funny structure that you will get. Well, will you get this or you will get some overlap. Probably in OCFE, you may not get exactly the structure. You might get some overlapping and you will have to do some, some more tricks to bring it to this form, okay. But there are some obligations where you get this nice banded structure and then each one of them is a ni*ni matrix. This vector xi belongs to Rni, so it is a ni dimensional vector, okay and same is true about vector bi which belongs to Rni.

So these are subsystems, these are subsystems which are arranged in a diagonal form, okay. Now what do I do when I want to solve this. Well one way is the (()) (39:17) that this entire matrix and use Gaussian elimination, okay. I can use Gaussian elimination on this entire matrix, okay. So what is n here, total n? My total n is summation i goes from 1-m ni, right. My total n, all number of, how many number of, what is the dimension of this entire matrix, n*n. What is n? See these are all subsystems. I am just adding them.

So the entire matrix, this entire matrix is n*n where n is equal to summation of all ni, okay. These are ni subsystems which I want to solve together. So if I use Gaussian elimination here directly, then the number of multiplications and divisions, if I use one without trying to exploit the structure of this matrix, if I try to use blanket Gaussian elimination, then number of divisions and multiplications, we just listed this, will be n cube+3n square-n/3.

(Refer Slide Time: 40:30)


$\varphi = \frac{n^3+3n^2-n}{3}$
$A_i x^{(i)} = b^{(i)} \quad i = 1, 2, \ldots m$
$\varphi_i = \frac{n_i^3+3n_i^2-n_i}{3}$
$\sum \frac{n_i^3+3n_i^2-n_i}{3} << \frac{n^3+3n^2-n}{3}$
CDEEP
IIT Bombay

The other option is, I could solve subsystems. So I could solve for Aixi=bi, i going from 1, 2, … m. I have this m subsystems, okay. I could each small subproblem, okay. See this is 3*3 or 4*4. Each one of them is a smaller dimensional system. I can solve this each one of them using Gaussian elimination, okay. Now what is psi i? What is psi i here? That will be ni cube+3ni square-ni/3, right.

So this ni are 3 4 5, you know these are smaller dimension matrices whereas this one this matrix is a large matrix. This matrix is a large matrix which has large number of rows and columns. So instead of using a Gaussian elimination on entire one, if I do this trick of solving each subsystem separately, then the total number of listing is sigma ni cube+3ni square-ni/3 is actually far far less than… this will be far far less than this because see what is n, n is summation ni, summation ni cube will have lot many terms then, okay.

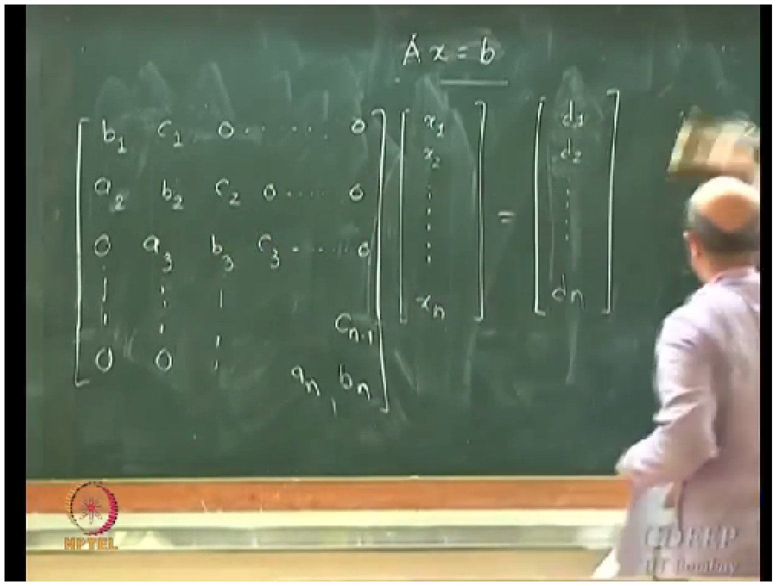
So this simple trick of identifying subblocks which are dense and solving those subblocks, okay, can actually reduce the computations multiple folds, okay. So within Gaussian elimination also, if you are instead of solving one giant problem in which there are lots of 0's and you are going to waste time in eliminating 0's, because they are already 0's. What you do in Gaussian elimination? You first bring it upper triangular matrix.

So you have to bring 0's and then if you just apply Gaussian elimination without thinking, you will waste time in bringing 0's from 0's, does not make any sense but the computer will just do computations without understanding that there is a 0 there. So you have to tell the computer well there are 0's, do not worry about those 0's. Just do the subsystem calculations and then you solve the problem much more efficiently than, okay.

So this is very very important particularly when you have iterative calculations. Suppose you have to solve Ax=b kind of equations inside a (()) (43:39) where A has this, the banded structure. Then just imagine. See when you are doing this repetitively say 1000 times or 10,000 times, well you better solve it efficiently, okay. So suppose this Ax=b, this banded structure appears inside a loop, okay, iterative loop, okay, each time if you do these many calculations that is much more expensive then these many calculations, okay.

That is where this banded matrix structure helps. Well the next one we are going to look at is the Thomas algorithm, okay. So the Thomas algorithm is for a special case of, Thomas algorithm may be some of you have done this in your undergraduate because it is often taught. So Thomas algorithm is for block, is first for tridiagonal matrices, okay. I am going to write this notation. I am going to change a little bit here. Right hand side, these elements, I am going to call them as d1 d2 …dn.

(Refer Slide Time: 44:58)

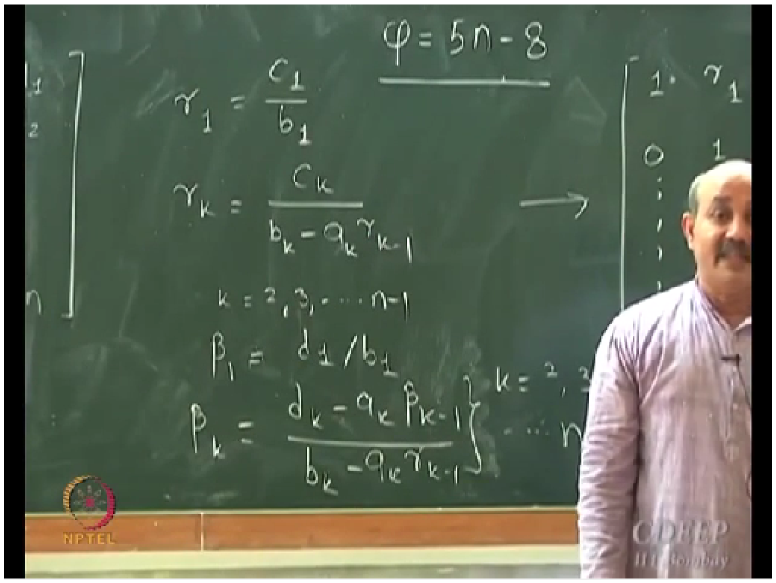


My matrix is now going to be written with d1 c1 0, then a2 b2 c2 0, then 0 a3 b3 c3 0. So this is a tridiagonal matrix. You have 0's here and then finally you will get an, bn. So you have this b1 b2 b3 up to bn that is the main diagonal, okay. There is one more diagonal above this which is c1 c2 c3 and one more diagonal below this a2 a3, so this starts from a2, this starts from c1 but this ends here at cn-1 and this starts from a2, ends in n. you have seen this kind of a matrix, finite difference method.

You have seen this kind of a matrix, okay. So in finite difference method, you get this tridiagonal matrix, okay and then questions is, can I exploit the spatial structure this has, so many 0's, filled with so many 0's. How many elements are non-0? 3n-2, 3n-2. N elements of diagonal, n-1 of first diagonal above and n-1 of first diagonal below. So 3n-2 elements are non-0, rest all the elements are 0, okay.

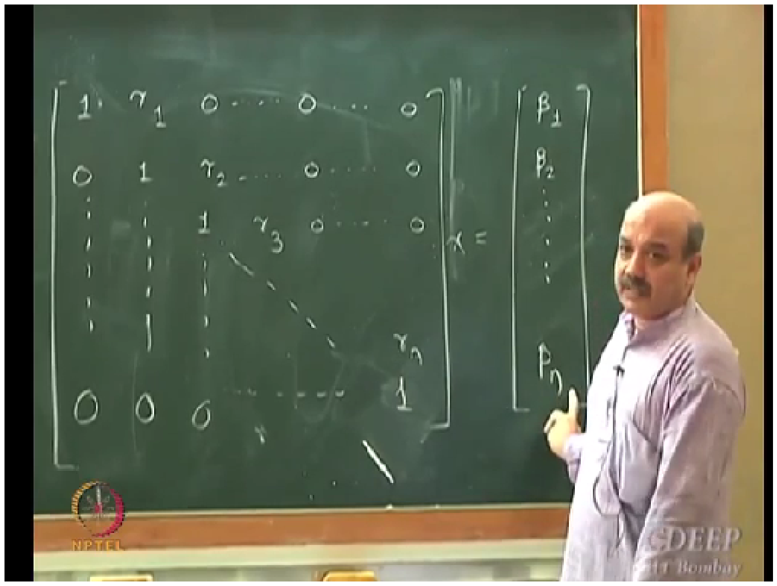
So I am going to do simple Gaussian elimination here. Can you do it? can you try this? How will you do Gaussian elimination? You should just eliminate; I do not have to do Gaussian elimination for these elements here. I just have to eliminate a2, okay. When I come here, I just have to eliminate a3, okay and so on. So I just have to eliminate one element below the diagonal, main diagonal, okay, that will give me the upper triangular matrix and then I just go on doing in the backwards sweep. So if you just write the steps here, it will be something like this.

(Refer Slide Time: 47:31)


$\varphi = 5n - 8$
$\gamma_1 = \frac{c_1}{b_1}$
$\gamma_k = \frac{c_k}{b_k - a_k\gamma_{k-1}}$
$k = 2, 3, \dots n-1$
$\beta_1 = d_1/b_1$
$\beta_k = \frac{d_k - a_k\beta_{k-1}}{b_k - a_k\gamma_{k-1}}$
$k = 2, 3, \dots n$
NPTEL
CDEEP IIT Bombay

So I am going to define this gamma 1 which is c1/b1. I am just writing it algorithmically. What I am doing this basically Gaussian elimination, okay and gamma k=ck bk-ak gamma k-1. This is for k=2, 3 up to n-1. Then I have this beta 1=d1/b1 and beta k=dk/bk-ak beta k-1 bk… so what I have written here, it looks probably at the first site, little complex but all that I am doing is eliminating 1 element below the diagonal, okay and these gamma and beta are the elements which appear in the… So what I am doing is after doing this operations, I am going to get this new matrix which looks like this.

(Refer Slide Time: 49:19)



This new matrix after doing all this, will look something like this 1 gamma 1 0 0, 0 1 gamma 2 0… Of course you have this matrix x here or vector x here and right hand side is beta 1 to beta 2 beta n. These equations written here are for computer implementation, you can put it in a for loop and just do calculations to get this matrix. This is just, these are the steps of Gaussian elimination written in the algorithmic way so that you can put it in the computer program too. So now I get this matrix, this is upper triangular matrix.

When you do upper triangular matrix here, you just have to worry about 1 row above the diagonal, right and now what you do backward sweep, okay. You do backward sweep, you get x1 directly from this, x2 you get right. All of you know back substitution. So now I do backward sweep and then solve the problem. What is the number of multiplications and divisions that are required in Thomas algorithm.

This is called Thomas algorithm. Now I do backward sweep on this. Should I write down the equations or you are clear about it? I think all of you know about this, right. How do you solve from this, it is very, very simple, okay? What is the point to take home is that number of multiplications and divisions for this case produce to 5n-8. This is just linear function of n as against cubic function of n. If I were to use Gaussian elimination without thinking about the structure, I would be doing n cube/3 calculations roughly.

Here I just do 5n-8, significantly lower than what I need to do as my normal Gaussian elimination. So Thomas algorithm is one example where you significantly reduce the computations required by exploiting the structure. The next what we are going to see is of course a simple extension of this is if you have, this I will do in my next class, is what if, see here we have looked at this algorithm where a b c are scalars, what if these are matrices?

What if b1 is a matrix, c1 is a matrix, a2 is a matrix, b2 is a matrix, those things are, those matrices are called as block tridiagonal matrices, okay, block tridiagonal matrices. So we just simply extend this idea to block tridiagonal matrices and still we can do much less computations, than doing entire, solving the entire thing, okay. So that is what is going to be the next extension. So these are some of the well known sparse matrix algorithms and you can significantly reduce the computations even in Gaussian elimination.

We are still doing Gaussian elimination. We are not guided into something new, okay. So we will continue on more of this in the next lecture.