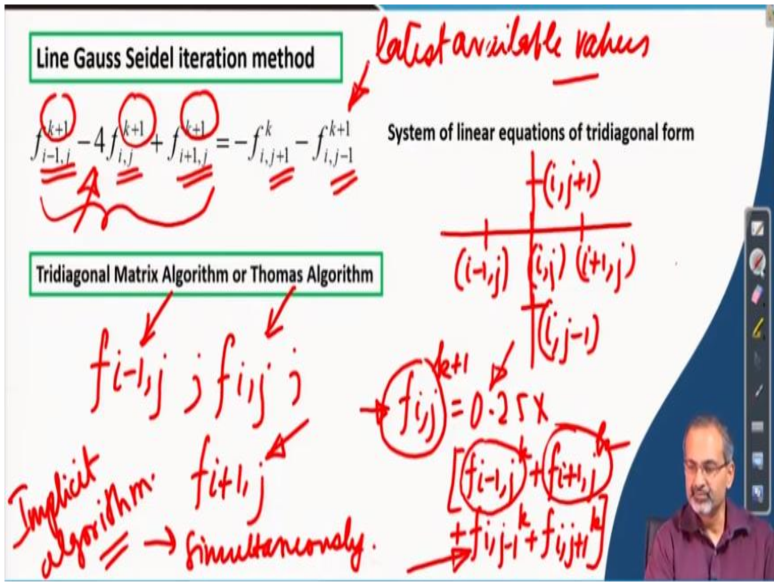
Introduction to CFD

Prof. Arnab Roy
Department of Aerospace Engineering
Indian Institute of Technology – Kharagpur

Module No # 04
Lecture No # 20
Numerical Solution of Steady State Heat Conduction (Elliptic PDE) (Cont'd)

In this lecture we continue to discuss on elliptic partial differential equations we have already discussed about methods like points Jacobi and Gauss Seidel. Now we will go ahead and try to discuss about other methods and also methods which are implicit in nature. So the methods that we have already discussed about are essentially explicit methods while the once which will discuss now we will also include implicit methods.

(Refer Slide Time: 00:59)



Line Gauss Seidel iteration method

$$f_{i-1,j}^{k+1} - 4f_{i,j}^{k+1} + f_{i+1,j}^{k+1} = -f_{i,j+1}^{k} - f_{i,j-1}^{k+1}$$

System of linear equations of tridiagonal form

Tridiagonal Matrix Algorithm or Thomas Algorithm

latest available values
$(i,j+1)$
$(i-1,j)$ (i,j) $(i+1,j)$
$(i,j-1)$
$f_{i-1,j}$; $f_{i,j}$; $f_{i+1,j}$
Implicit algorithm → simultaneously
$f_{i,j}^{k+1} = 0.25 \times [f_{i-1,j}^{k} + f_{i+1,j}^{k} + f_{i,j-1}^{k} + f_{i,j+1}^{k}]$

To begin with we have a scheme which is called as the line Gauss Seidel iteration method. So if you look at this scheme what we have essentially done is that we have associated the k+ 1th iteration levels to values of the function corresponding to grid points along a specific line. So if we look back at our stencil looks like this so in general when we have explicit formulation let us say point Jacobi then what do we do.

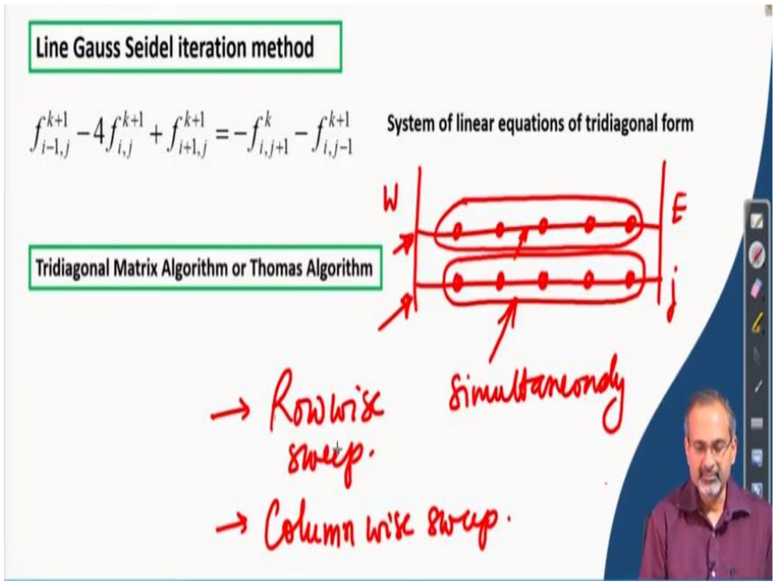
We obtain f ij + 1 by doing f i – 1 j at k + fi+ 1j at k f ij -1 at k and f ij +1 at k now if you choose those functional values or those grid points functional values which are aligned along the x direction then you would have this. So if you bring them all to the left hand side and assigned an iteration of k+1 then this is how it will look like. Because this factor 0.25 of course we; will go and multiply this f ij on the left hand side already.

And then if you transpose the terms this is how it will look like and then the remaining terms will be coming from here. The points which are available along the y direction for j +1 and j – 1 only additional thing is that we are following the; Gauss Seidel approach and therefore we are making use of the latest available values. And therefore we have a k +1 value associated with the ij -1 point because you would have swept that point before you came to the point ij. Now when you come to ij you are solving the values of fi -1j f ij and fi+1j simultaneously.

So that would mean that this is an implicit algorithm. And as you can understand that this will have a tri-diagonal structure because we have 3 unknown's in a line. All these 3 are unknowns and they are lying on a particular row. If you look at the matrix form of the equation so as though you are trying to sweep the domain, and cover all the interior grid points by writing down a system of linear algebraic equations which can be expressed in a tri-diagonal form that would be the structure of this problem.

Now if you do so you would have an efficient algorithm to work out the problem which we talk about as the tri-diagonal matrix algorithm.

(Refer Slide Time: 04:47)



Line Gauss Seidel iteration method

$$f_{i-1,j}^{k+1} - 4f_{i,j}^{k+1} + f_{i+1,j}^{k+1} = -f_{i,j+1}^{k} - f_{i,j-1}^{k+1}$$

System of linear equations of tridiagonal form

Tridiagonal Matrix Algorithm or Thomas Algorithm

→ Rowwise sweep.

→ Column wise sweep.

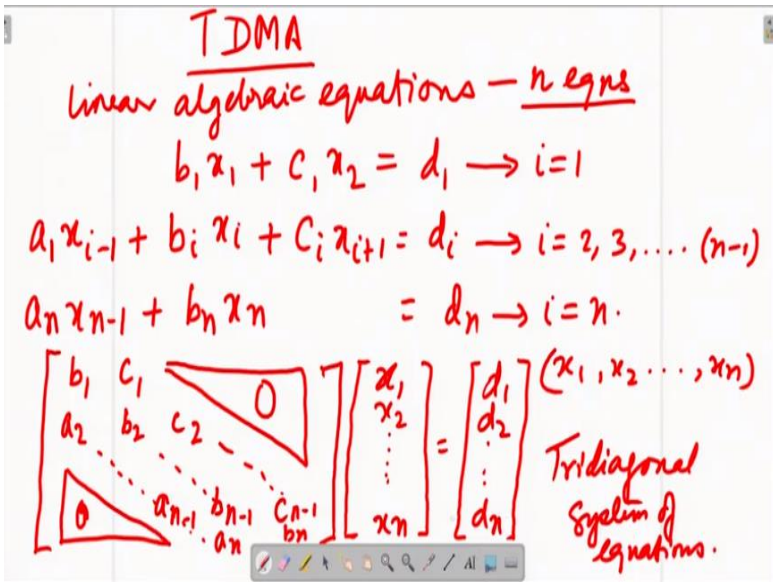
But before we go into that algorithm let us once again look at the structure of the problem as we have lay down in this case. So what we are trying to say is that we are taking the grid points row by row and we are trying to solve them simultaneously and sweep the domain. So if this is the west boundary this is the east boundary of the domain you are trying to solve for all the interior grid points for a given j simultaneous.

So that is what the line Gauss Seidel iteration method is all about. And why is it called as an iteration method? Because if you solve for all the values in 1 short and then move to the next row and then solve them again that way. And you run through the entire domain once you may not be able to reach a converge value of the function that way in one such sweep. You may actually like to do the sweep of the domain in a different manner in the next round.

So here it is a row wise sweep next time you may like to do it as a column wise sweep and that could be good for certain problems which do not have any preferential direction involved as a part of the physics of the problem itself. However, continuing with the row wise sweep over different iterations could help in problems, where inherently the problem as the directional bias.

That means you know if priory from the physics of the problem and the variations are going to be predominantly happening in the row wise directions. So then there is a sense in doing it row wise, so the main question to pose in this cases that if we know understand that tri-diagonal practice algorithm will be a very useful tool to use, then how does this algorithm function at all. So let us try to discuss on the nature of the tri-diagonal matrix algorithm.

(Refer Slide Time: 07:10)



TDMA

linear algebraic equations — n eqns

$$b_1 x_1 + c_1 x_2 = d_1 \rightarrow i=1$$

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i \rightarrow i = 2, 3, \ldots (n-1)$$

$$a_n x_{n-1} + b_n x_n = d_n \rightarrow i = n.$$

$$\begin{bmatrix} b_1 & c_1 & & 0 \\ a_2 & b_2 & c_2 & \\ & \ddots & \ddots & \ddots \\ & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} \quad (x_1, x_2, \ldots, x_n)$$

Tridiagonal System of equations.

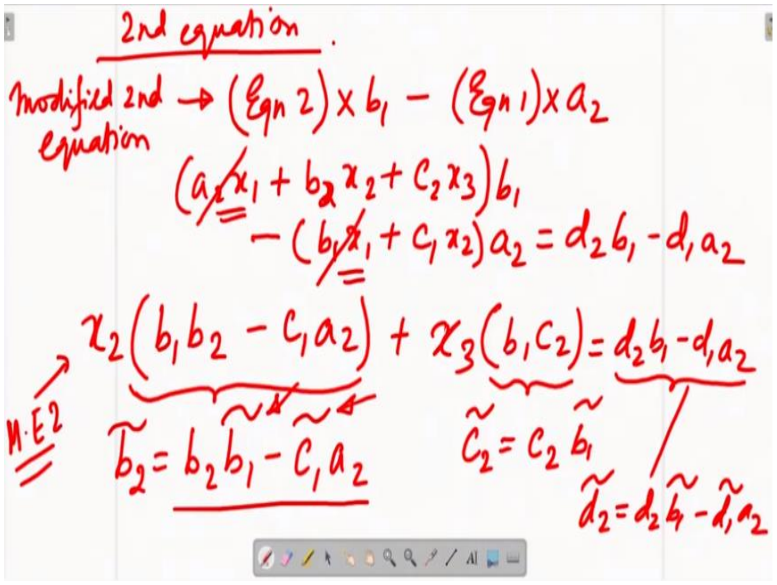
So, that we have so let us try to write down the system of linear algebraic equations. And then try to express them in a matrix form so let us say that the equations look like this. The variables are given by x1, x2 up to say n and then we have coefficients associated with those variables b, c and you have known values which are on the right hand side. So let us say this is the first equation we just write in a slightly different manner.

So this is essentially a system of n linear algebraic equations and we need to solve them simultaneously in order to obtain the values of the unknowns which are x1, x2 up to xn. So this is essentially the solution we are seeking so let us first write it down in a matrix form so how will the matrix look like let us try to fill up a matrix. So we already know about the coefficient matrix we have discussed about that how these entries are going to be zero's also these would be zero's.

And then you have a column vector here the column vector comprises of the unknowns. So this is essentially the matrix representation of the equations that we have written above. So we have to find an efficient way to solve this system. Of course one thing that we realizes that it is a tri-diagonal of equations. Now we will do a few steps in order to eliminate variables from these equations that these n equations that we have in a systematic manner and then the progressive manner.

And then try to find some recursive relations so that when it comes to doing the computations these recursive relations can be intelligently used in our computer course. So let us look at the second equation.

(Refer Slide Time: 11:11)



2nd equation

modified 2nd equation $\rightarrow (\text{Eqn } 2) \times b_1 - (\text{Eqn } 1) \times a_2$

$$(\cancel{a_2 x_1} + b_2 x_2 + c_2 x_3) b_1 - (\cancel{b_1 x_1} + c_1 x_2) a_2 = d_2 b_1 - d_1 a_2$$

M.E.2

$$x_2 \underbrace{(b_1 b_2 - c_1 a_2)} + x_3 \underbrace{(b_1 c_2)} = \underbrace{d_2 b_1 - d_1 a_2}$$

$$\tilde{b}_2 = b_2 \tilde{b}_1 - \tilde{c}_1 a_2 \qquad \tilde{c}_2 = c_2 \tilde{b}_1 \qquad \tilde{d}_2 = d_2 \tilde{b}_1 - \tilde{d}_1 a_2$$

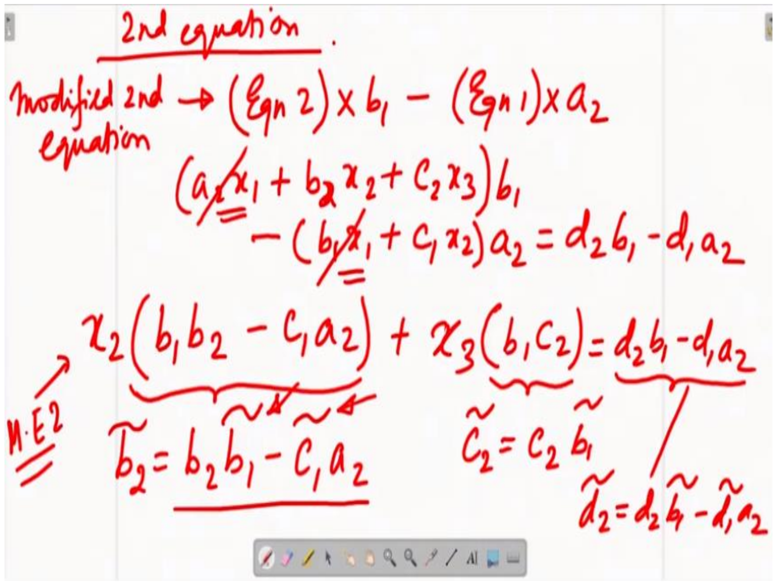
When you look at the second equation we will try to do some modifications in such a manner that we end up having a modified second equation where we manage to eliminate one of the variables. So these steps are essentially having some similarity with even steps that we do in Gauss elimination. So what we will do is we will multiple this existing equation 2 with b1 and subtract equation 1 multiplied by a2. So what does that produce for us?

Let us see whether we are able to eliminate 1 variable from the second equation in the process. So this is what we will get as an outcome you will see interestingly that this term and this term would cancel out in the process. Because you have a2 b1 x1 here you have the same thing produced here and subtracted. So you finally have one of the variables reduced so we will call this as the modified equation 2.

Also we will give some nomenclature for these terms that we have generated. We will call this as b2 tilde and what we write over here on the right hand side of the b2 tilde expression. Also contains some tilde expressions which we are doing for convenience because these would help us generate the recursive relations which we are aiming to generate later on. So you will understand more clearly later what is the motivation behind using these; tilde expression here?

We similarly call this as c2 tilde and that we write as c2 time’s b1 and tilde and this as d2 tilde so we have completed our work with modified second equation.

(Refer Slide Time: 14:32)



2nd equation.

modified 2nd equation $\rightarrow$ (Eqn 2) $\times b_1$ $-$ (Eqn 1) $\times a_2$

$$(\cancel{a_2 x_1} + b_2 x_2 + c_2 x_3) b_1 - (\cancel{b_1 x_1} + c_1 x_2) a_2 = d_2 b_1 - d_1 a_2$$

M.E2

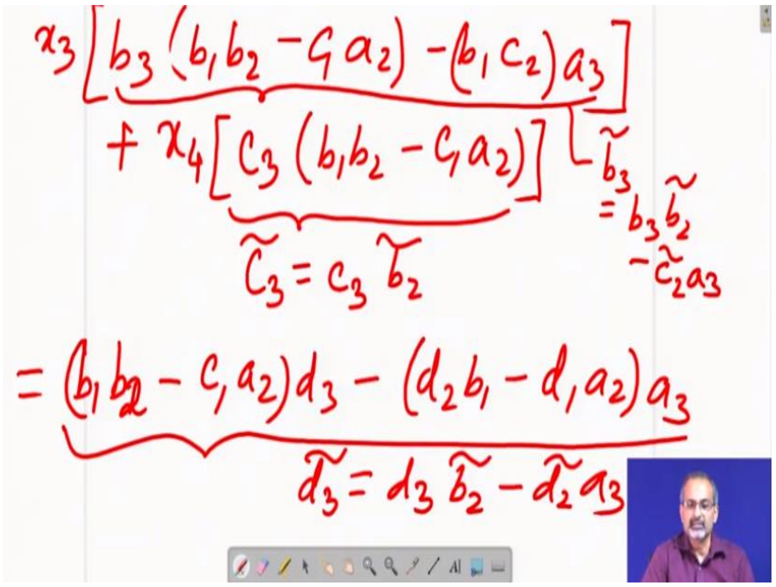
$$x_2 \underbrace{(b_1 b_2 - c_1 a_2)} + x_3 \underbrace{(b_1 c_2)} = \underbrace{d_2 b_1 - d_1 a_2}$$

$$\tilde{b}_2 = b_2 b_1 - c_1 a_2 \qquad \tilde{c}_2 = c_2 b_1 \qquad \tilde{d}_2 = d_2 b_1 - d_1 a_2$$

Let us do a similar exercise for the third equation that means we are going to going to end for the modified third equation where one of the variables gets eliminated. So let us first write down the equation 3 so equation 3 as it stands looks like and we are now going to do a modified equation 3 so abbreviated as ME3. So what will do in generating that then multiply the existing equation 3 by an expression b1 b2 – c1 a2 will understand the significance of this operation later and subtracted and what we subtract from there is equation 2 into a3.

So what does that produce let us try to write down the expressions what you have on the right hand side is this. So what you have achieved essentially over here is that this term a3 x2 times this will get cancelled with this. Because you have a x2 here a3 out here and that exactly matches with this and the bracketed term is identical for both the cases.

(Refer Slide Time: 16:58)



$$x_3\Big[\underbrace{b_3(b_1 b_2 - c_1 a_2) - (b_1 c_2)a_3}\Big] \quad \tilde{b}_3 = b_3 \tilde{b}_2 - \tilde{c}_2 a_3$$

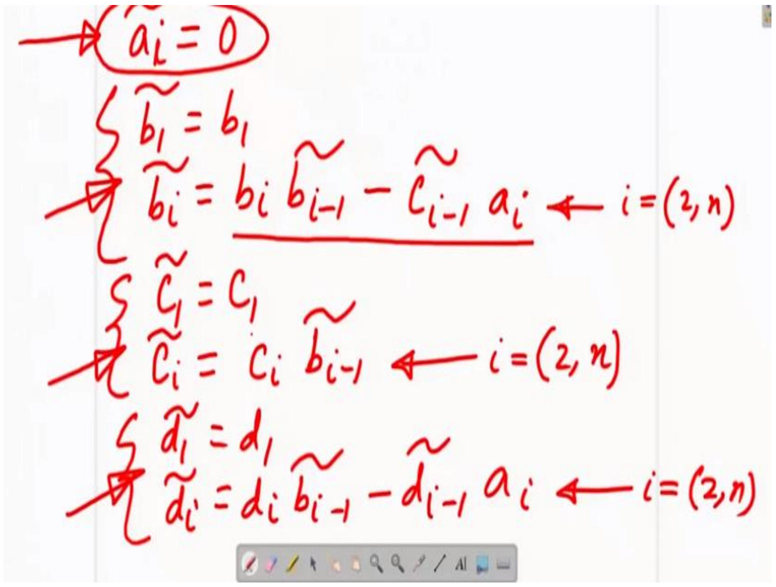
$$+\, x_4\Big[\underbrace{c_3(b_1 b_2 - c_1 a_2)}_{\tilde{c}_3 = c_3 \tilde{b}_2}\Big]$$

$$= \underbrace{(b_1 b_2 - c_1 a_2)d_3 - (d_2 b_1 - d_1 a_2)a_3}_{\tilde{d}_3 = d_3 \tilde{b}_2 - \tilde{d}_2 a_3}$$

So you will have a simplification after you see simplify this, what you can write is and what we will do is we will call this whole expression as b3 tilde and that will be expressed as b3 times b2 tilde – c2 tilde times a3 and we will call this as c3 tilde equal to c3 times b2 tilde. So this is what you have on the left hand side of the equation and what you have on the right hand side of the equation can be written as.

So this whole right hand side we called as d3 tilde so we have whole lot of modified coefficients which are expressed as tildes. And the main question to ask this point is. Are we heading towards a generalization? So if we observe carefully what we have generated by modifying this equations.

(Refer Slide Time: 18:47)



$$\boxed{\tilde{a}_i = 0}$$

$$\left\{\begin{array}{l} \tilde{b}_1 = b_1 \\ \tilde{b}_i = \underline{b_i \tilde{b}_{i-1} - \tilde{c}_{i-1} a_i} \leftarrow i = (2, n) \end{array}\right.$$

$$\left\{\begin{array}{l} \tilde{c}_1 = c_1 \\ \tilde{c}_i = c_i \tilde{b}_{i-1} \leftarrow i = (2, n) \end{array}\right.$$

$$\left\{\begin{array}{l} \tilde{d}_1 = d_1 \\ \tilde{d}_i = d_i \tilde{b}_{i-1} - \tilde{d}_{i-1} a_i \leftarrow i = (2, n) \end{array}\right.$$

We can watch that certain precautions have surfaced so we have different coefficients we have a's, b's, c's and then we have the right hand side d's. So you will see carefully if you look at the modified equations in particular these recursive relations would work for each one of them. And now you will understand better that how those tilde values have been indicated in each of the modified equations.

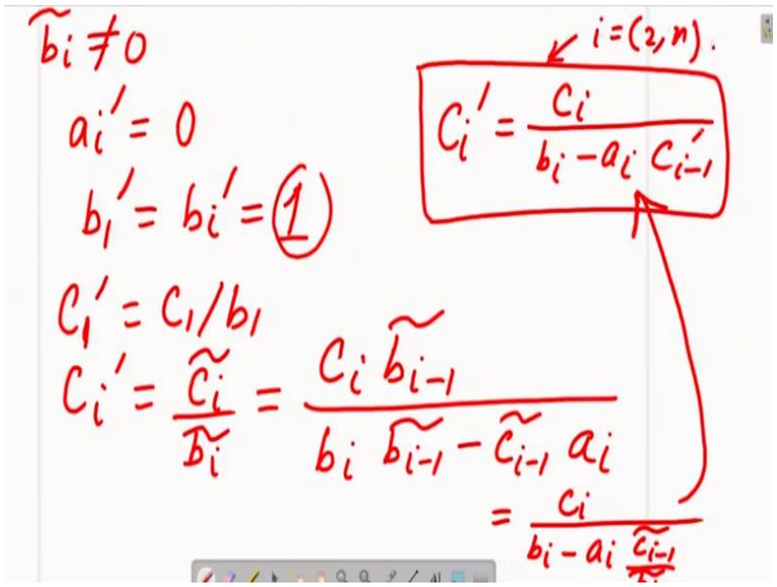
So we have indicated that how these tilde's value are to be interpreted for all i's ai tilde is going to be 0 it is because 1 of the variables is being eliminated in each of the modified equation. And that happens to be the first variable in that row so therefore the coefficients for all of them are going to 0. That is why these are showing up that way whereas for the other coefficient b's and c's b1 tilde is b1. But for all the remaining i's starting from 2 to n the recursive relation works like this, where you do not only have b but you also have products of bi times bi -1 tilde –ci – 1 tilde into ai and so on.

Similar recursion relations are indicated here for c and d and if you look at each one of those modified equations you will find that these recursion relations work. And that is a big convenience for us because when we do computer programming of this algorithm then we can recursively use these expressions for generating the modified coefficients for each and every linear equation that we have in our family.

What do we finally gain out of it we gain a significant simplification in the sense that this way if you keep eliminating 1 unknown from each one of the equation that, we have when we reach the nth equation which should have contained 2 variables would now have only one variable to be resolved. And therefore once you solve that variable you can feedback that solution to the n – 1th equation which will contain 2 variables.

Out of which 1 has been solved already so the other one can be solved from that equation and so on. So you can just solve the entire system of equations by back substitutions so that way if we just try to formally lay down the structure by writing down the remaining equations they would look somewhat like this.

(Refer Slide Time: 22:42)



$\tilde{b}_i \neq 0$

$a_i' = 0$

$b_1' = b_i' = \textcircled{1}$

$c_1' = c_1/b_1$

$$c_i' = \frac{\tilde{c}_i}{\tilde{b}_i} = \frac{c_i \tilde{b}_{i-1}}{b_i \tilde{b}_{i-1} - \tilde{c}_{i-1} a_i} = \frac{c_i}{b_i - a_i \frac{\tilde{c}_{i-1}}{\tilde{b}_{i-1}}}$$

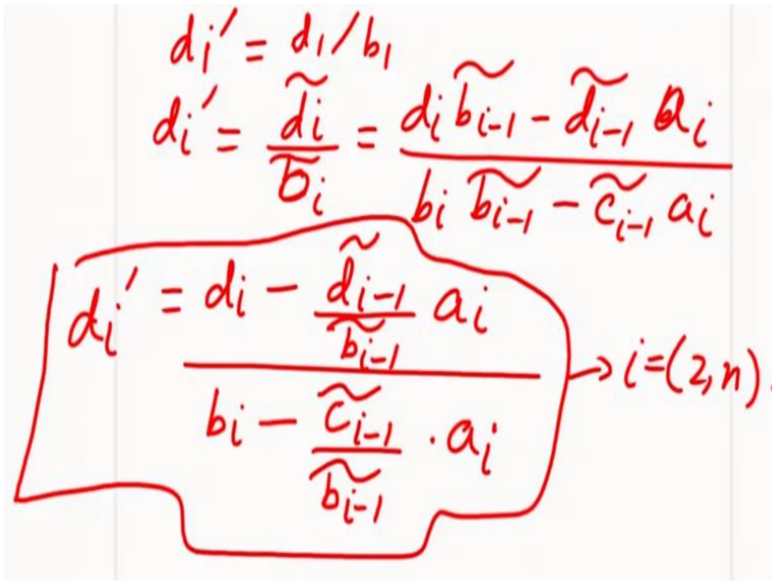
$i = (2, n).$

$$\boxed{c_i' = \frac{c_i}{b_i - a_i c_{i-1}'}}$$

So we can have one further simplification if we have the bi tilde’s not equal to 0 then we can divide all the coefficients by bi tilde and therefore we could name them now as primes. So ai prime would be ai tilde / bi tilde which will remain as 0 b1 tilde will be equal to bi sorry b1 dash will be equal to bi dash = 1 that is because all of them will be divided by bi tilde which will give you a 1 c1 dash will be c1 / b1.

And for the remaining i’s that is from 2 to n it will be ci tilde / bi tilde now if you substitute the expressions for those the numerator and the denominator this is what you will get. And then this can be further simplified as. So this expression that we have written can finally give us an expression for ci dash which will look like this is nothing but ci – 1 dash. And so now you have an expression for ci dash of course this holds good for i = 2 to, n.

(Refer Slide Time: 24:48)



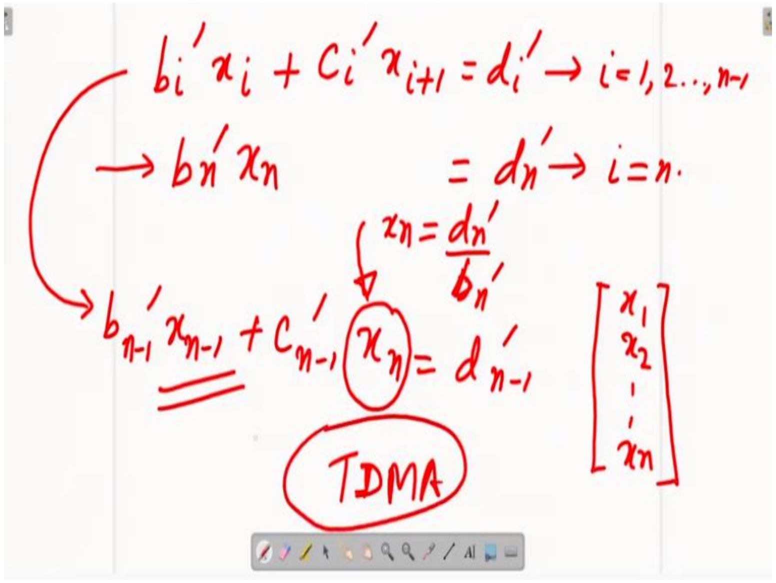
$$d_1' = d_1/b_1$$

$$d_i' = \frac{\tilde{d}_i}{\tilde{b}_i} = \frac{d_i \tilde{b}_{i-1} - \tilde{d}_{i-1} a_i}{b_i \tilde{b}_{i-1} - \tilde{c}_{i-1} a_i}$$

$$\boxed{d_i' = \frac{d_i - \frac{\tilde{d}_{i-1}}{\tilde{b}_{i-1}} a_i}{b_i - \frac{\tilde{c}_{i-1}}{\tilde{b}_{i-1}} \cdot a_i}} \rightarrow i=(2,n)$$

Then we can look at di dash which will be d1 / b1 and then for the other values of i from 2 to n it will be the, i tilde / bi tilde. So what we are doing essentially over here is just using those recursive formulas in the respective places and trying to simplify. So we now have an expression for di dash and this holds good for i = 2 to m.

(Refer Slide Time: 26:11)



$$b_i' x_i + c_i' x_{i+1} = d_i' \rightarrow i=1,2\ldots,n-1$$

$$\rightarrow b_n' x_n = d_n' \rightarrow i=n.$$

$$x_n = \frac{d_n'}{b_n'}$$

$$b_{n-1}' \underline{\underline{x_{n-1}}} + c_{n-1}' \boxed{x_n} = d_{n-1}'$$

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$\boxed{\text{TDMA}}$$

Now if you put all the things together what have you got finally we have modified the same system of linear equations an expressed it like this. This is the form for i = 1 to, n -1 and what you have as the nth equation will looks like this. So this is for the nth equation so what have we got? We have got only one variable to solve in the nth equation which is nothing but xn = dn dash / dn dash. Once you have that you go back to the n – 1th equation where if you substitute i = n – 1 in this equation what we will you get?

You will get bn – 1 dash xn – 1 + c n-1 dash xn = dn -1 dash so you have already solved for xn from the last equation which will be substituted here and therefore you can solve for the only unknown in the n-1th equation which attains to be xn-1. And this way you move backwards towards the first equation ultimately ending up solving for the entire system of unknowns. So this is essentially how the TDMA functions.

So we now realize that if we are able to have a tri-diagonal matrix structure in the coefficient matrix how conveniently it could be done through the TDMA approach. We will look at application of TDMA further in later lectures thank you.