

**Introduction to CFD**  
**Prof. Arnab Roy**  
**Department of Aerospace Engineering**  
**Indian Institute of Technology – Kharagpur**

**Module No # 05**

**Lecture No # 21**

**Numerical Solution of Steady State Heat Conduction (Elliptic PDE) (Cont'd)**

In this lecture we continue our discussion on application of tri-diagonal matrix algorithm.

(Refer Slide Time 00:29)

**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**

In the previous lecture we had discussed at length about how the tri-diagonal matrix algorithm or the Thomas algorithm functions for a system of linear algebra equation which can laid in a tri-diagonal form. And we just would like to reiterate the fact that the last time when we were discussing about the line Gauss Seidel iteration method that was the requirement we needed the tri-diagonal matrix algorithm to solve the problem.

So now that we understood the TDMA we should now be able to apply it to solve this system of linear algebraic equations. As we discussed that we would solve the problem line by line sweeping the domain from bottom to top or it could also be done from say west to east when we sweep column by column. The main message is that we could pick up a set of grid point lying along a particular line which could be disposed along the, i direction or j direction and solve the values of the function f at all such grid points lying on that particular line simultaneously.

And that would be possible through the TDMA. Having understood this let us look at some other possible ways of solving elliptic partial differential equations.

(Refer Slide Time 01:55)

**Point Successive Over-Relaxation (PSOR)**

$$f_{i,j}^{k+1} = \frac{1}{4} [f_{i+1,j}^k + f_{i-1,j}^{k+1} + f_{i,j+1}^k + f_{i,j-1}^{k+1}]$$

Adding and subtracting  $f_{i,j}^k$  on RHS of the above equation and collecting terms

$$f_{i,j}^{k+1} = f_{i,j}^k + \frac{1}{4} [f_{i+1,j}^k + f_{i-1,j}^{k+1} + f_{i,j+1}^k + f_{i,j-1}^{k+1} - 4f_{i,j}^k]$$

As the iterations proceed,  $f_{i,j}^k$  should approach  $f_{i,j}^{k+1}$ . In order to accelerate the solution, the bracketed term is multiplied by a relaxation parameter  $\omega$ .

$$f_{i,j}^{k+1} = f_{i,j}^k + \frac{\omega}{4} [f_{i+1,j}^k + f_{i-1,j}^{k+1} + f_{i,j+1}^k + f_{i,j-1}^{k+1} - 4f_{i,j}^k]$$

For stable convergence of the solution the range of values to be used is  $0 < \omega < 2$

$0 < \omega < 1$	Under relaxation
$\omega = 1$	Gauss Seidel
$1 < \omega < 2$	Over relaxation

Numerical experiments are required to obtain the optimum value of  $\omega$  for a given problem

For accelerated convergence

Very powerful method to accelerate convergence of solution of elliptic partial differential equations by iterative means is what is called as the point successive over relaxation method. So we are coming back once again to iterative method in between we looked at how we could solve a system of linear algebraic equation simultaneously to sweep domain line by line solving the functional value simultaneously.

But we again come back and we look at iterative methods here with a specific purpose of discussing in a particular acceleration convergence tool which we call as over relaxation. So earlier we had discussed about the fact that relaxation essentially means averaging of neighboring values. And what we mean by over relaxation is that whether we could accelerate this process how do we go about doing that?

So; how we do it mathematically just by adding and subtracting  $f_{i,j}$  at the  $k$ th iteration level on the right-hand side of the above equation. And then rearranging the terms a little bit so what you can see is that? What you have put as  $f_{i,j}^k$  shows of here and  $-f_{i,j}^k$  has now been put inside the bracket and therefore now has a coefficient  $-4$  because you have a one fourth outside the bracket. So therefore, they cancelled each other.

However trivial this step may sound like it does a very significant change in the way you approach the algorithm. So what you say is that as the iterations proceed ideally  $f_{ij}$  at the  $k$ th iteration level should be approaching the  $f_{ij}$  at the  $k + 1$ , iteration level. And the sooner it achieves that transformation from the  $k$ th to  $k + 1$ th level the sooner the iteration would converge.

So can you really expedite this process so the process expediting would then happen through what we call as relaxation parameter  $\omega$  which we put over here? So we are essentially multiplying the bracketed term with a factor  $\omega$ . So that we can send the bracketed term as quickly possible to 0. What does that achieve for us? It achieves the fact that if it really reaches 0 then  $f_{ij}$  at  $k + 1$  will approach  $f_{ij}$  at  $k$  and that is essentially reaching convergence.

So the idea is to send this whole term to 0 as soon as possible by using  $\omega$  to expedite the process. Now the obvious question that arises in that case is how big the  $\omega$  that you can expedite as much as you need to. So from stability analysis we can actually show that the range of  $\omega$ s that you can use to have a stable convergence of the solution lies anywhere between 0 and 2.

So if you use 0 then you are not essentially aligning the solution progress at all. If you are using 2 you are accelerating it significantly but you need to be just below 2 in general to keep the solution process stable. So you can marginally keep it below 2 then what you are then doing is what is called as over relaxation that means you are using a large value of  $\omega$  in order to accelerate convergence.

So this happens to be a very important tool in accelerating the convergence of when we solve elliptic partial functional equations. If we are using  $\omega$  value of 1 then we are essentially using Gauss Seidel. If we are using anything below 1 then we are under relaxing that means we are going slower than what Gauss Seidel does. And knowing the optimum value of  $\omega$  is a difficult issue.

So most often for specific equations and specific problem scenarios would actually have to do some numerical experiments before you get to know what the optimum value of  $\omega$  is? So, there is usually no a priory prescription for an optimum  $\omega$ .

(Refer Slide Time 06:43)

**Line Successive Over-Relaxation (LSOR)**

$$f_{i-1,j}^{k+1} - 4f_{i,j}^{k+1} + f_{i+1,j}^{k+1} = -f_{i,j+1}^k - f_{i,j-1}^k \quad \text{Line Gauss Seidel iteration method}$$
$$\omega f_{i-1,j}^{k+1} - 4f_{i,j}^{k+1} + \omega f_{i+1,j}^{k+1} = 4(\omega - 1)f_{i,j}^k - \omega(f_{i,j+1}^k + f_{i,j-1}^k)$$

$0 < \omega < 2$        $1 < \omega < 2$   
over relaxation

Now that we understood the idea of point successive over relaxation you can think of doing it line by line. What we have done for a line Gauss Seidel iteration method earlier where we applied TDMA to solve the problem. You could now incorporate omega into the equation by doing a very simple rearrangement, like the one we did for point successive over relaxation that we just added and subtracted  $f_{i,j,k}$  on the right hand side of the equation.

You can do the exercise yourself and then show that you end up getting an equation of this kind where you now have a control over how quickly this iteration would converge. So you can again end up using the same range of values for omega based on the prescribed constraint. And of course if you are looking for over relaxation it should be lying with the range like this. So this would accelerate your solution.

So this of course involves simultaneous solution of the functional values line by line that is either rows or columns. So row by row sweeping the entire domain or column by column sweeping the entire domains. And since it involves several iterations you can do it in alternate manner in order to remove bias in most cases.

(Refer Slide Time 08:15)

**Alternating Direction Implicit scheme**

$f_{i-1,j}^{k+1} - 4f_{i,j}^{k+1} + f_{i+1,j}^{k+1} = - \left( f_{i,j+1}^{k+1/2} + f_{i,j-1}^{k+1/2} \right)$  TDMA  $(k+\frac{1}{2})^{th}$  level  
 $f_{i,j-1}^{k+1} - 4f_{i,j}^{k+1} + f_{i,j+1}^{k+1} = - \left( f_{i+1,j}^{k+1} + f_{i-1,j}^{k+1} \right)$   $k^{th}$  level

Convergence acceleration is possible through over relaxation

Y sweep ADI scheme  
 → pentadiagonal  
 + → tridiagonal

X Sweep

We would like to discuss another very powerful algorithm for solving elliptic partial differential equations, which is called as the alternating direction implicit schemes in the literature it is often referred to as the ADI scheme. What is the solutions strategy in this case? Let us draw a simple diagram to explain what ADI scheme essentially does? So we are essentially drawing a grid to represent your grid points at let us say the kth iteration level.

And then our, in is to take the solution to k + 1th iteration level. That is the functional values at the inner grid points would then change and reach the k + 1th level. As we do that we actually split up the whole process into 2 sub steps. So like we have marked the kth level and the k + 1th level we would mark an intermediate level which we would call as k + half level. And now we need to figure out what we would like to do with these 3 levels.

Of course the solution is already available to you at the kth level that is the assumption. So once that is available to you what you first do is in the k + halfth level you do what is called as x sweep. What do we mean by x sweep? Like we were talking about a line by line solution of the system of equations here we just take the grid point's row wise. So in this row you would take these 2 grid points can solve for their values once you solve them you move up and you solve for the next row and so on.

So if that is the case how would the governing equation of this form look like let us see this first equation. So, in this first equation we have taken a grid points along the, i direction. So you have

$i - 1$   $i + 1$  and the level of iteration that you have mention here is  $k + \text{half}$  of all of them. What do you have on the right hand side you have the functional values along the  $j$  direction. So you have  $j + 1$  and  $j - 1$  and  $j - 1$  has already been updated because you are sweeping the domain row wise.

So you have swept through the  $j - 1$  row already and reached  $j$ . Therefore you have an updated value here. But  $j + 1$  is yet to reach so therefore it remains at the  $k$ th level value. So these are known to you they are send to the right hand side. Now what will this do? This will generate a system of linear algebraic equations in every row which is specified by a certain  $j$ . And therefore you will have a tri-diagonal system of equations to be solved which will be managed to application of TDMA. So what are we doing in this step essentially?

We are reaching the solution as though by half a step by sweeping the entire domain by doing  $x$  sweeps. And then in the next half portion of the iteration we will come back and we will do the  $y$  sweeps. So in that case the  $y$  sweep would be represented by the second equation where if you look at the left hand side terms then we are now reaching the  $k + 1$ th iteration level, and they are points disposed along the  $y$  direction or the  $j$  direction. So you have  $j - 1$   $j$  and  $j + 1$ .

What you have on the right hand side are values which you have reached at least  $k + \text{half}$  level or even  $k + 1$  level. So  $k + 1$  level as been reached by which grid point it is the  $i - 1$   $j$  grid point because you have sweeping by column wise values. So by the time you came up to the  $i$  value, you have already swept to the column of  $i - 1$ . And therefore that has already reached the  $k + 1$ th level.

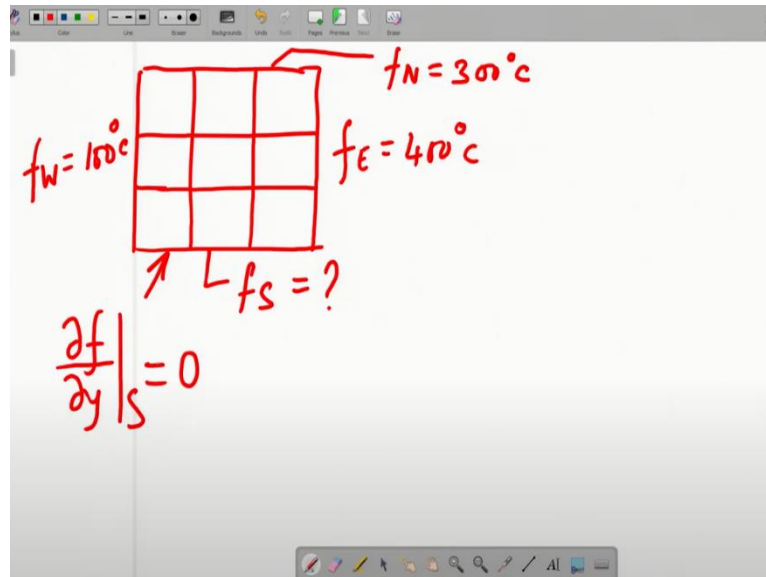
While you have yet the reach the column which has an index  $i + 1$  because that lies to your left sorry lies to your right if you are seeping from left to right. And therefore it remains at the  $k + \text{half}$  iteration level which you had completed through the  $x$  sweeps. So that is the manner in which we interpret what is available on the right-hand side of the discretized equation in the  $y$  sweep. So we understand that the first equation stands for the  $x$  sweep the second equations stands for the  $y$  sweep.

And once you have completed the full round of  $x$  sweep and  $y$  sweeps then you have essentially moves the solutions from the  $k$ th to the  $k + 1$ th iteration level. So this is the strategy by which we

split the direction of the sweep and therefore take the sweeps along 2 respective Cartesian directions separately. Once is succession of the other and therefore reach a next iteration level.

One big advantage that we gain out of it is that if we were to do it in one step then we would have to handle a pentadiagonal coefficient matrix which we have discussed in previous lecture. While in this case because we are splitting directions and sweeping them we have managed to solve 2 sequential tri-diagonal matrix based problems in the 2 sweeps. So that adds big convenience in the sense that we have a very effective TDML algorithm doing this job for us in 2 sequential sweeps. We would like to go back and solve a simple numerical problem.

**(Refer Slide Time 15:55)**



We recall about the problem that we were solving on a square domain with temperatures define along the different boundaries. In the previous case we had considered Dirichlet boundary condition along all the boundaries. In this case we will briefly look at what is the change in the calculations if you were to change the boundary condition at only one of the boundaries. So we have defined this problem as mixed Dirichlet Neumann problems.

We have discrete conditions of east, west and north. And a Neumann condition of 0 normal radiant of, f in the south boundary. So what is the impact on the solution?

**(Refer Slide Time 16:57)**



$$\frac{\partial f}{\partial y}|_s = \frac{-3f_{i,1} + 4f_{i,2} - f_{i,3}}{2\Delta y} = 0$$

$$f_{i,1} = \frac{4f_{i,2} - f_{i,3}}{3}$$

We recall having obtained an expression for the first order derivative in the form of 1 sided finite difference expression earlier involving 3 grid points. So that expression will come in handy over here in order to define the functional value at any grid point lying on the south boundary. Now that you have a Neumann boundary condition you have to apply a finite difference approximation of this derivative and set it to 0 at the south boundary in order to get an expression for f of i 1.

Where i vary from the minimum to the maximum range if you have done that then this can help you to start the computations in this manner.

**(Refer Slide Time 18:10)**

Guess interior grid point values = 200°C

(2,2)  
(3,2)  
(2,3)  
(3,3)

$$f_{2,1}^0 = \frac{4f_{2,2}^0 - f_{2,3}^0}{3} = \frac{4 \times 200 - 200}{3} = 200^\circ C$$

$$f_{3,1}^0 = \frac{4f_{3,2}^0 - f_{3,3}^0}{3} = \frac{4 \times 200 - 200}{3} = 200^\circ C$$



So for example you have to guess interior grid point values as usual. So let us continue with the old approximation of 200 degree centigrade for the interior grid points. Which of our points if you recall there where 4 points 2, 2 3, 2 2, 3 and 3, 3 these, where the interior grid points. Let us guess their values as this like as we did this earlier. This time we have to also guess values at the lower boundary point or the south boundary points.

So let us apply that finite difference formula in order to estimate them. So, both turn out to be 200 degree centigrade so this is the initial guess. And you may remember that all the initial guesses where mark with the 0-iteration level.

**(Refer Slide Time 19:46)**

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

*1st iteration*

$$f_{2,2}^1 = \frac{f_{3,2}^0 + f_{1,2}^0 + f_{2,3}^0 + f_{2,1}^0}{4}$$

$$= \frac{200 + 100 + 200 + 200}{4} = 175^\circ\text{C}$$

Having, said that we continue with the iterative formula coming from the Jacobi method. Of course now we know that we can have much better methods to do these calculations. But we are just taking the old formula in order to just compare between what we saw happening with our Dirichlet based boundary conditions. And now the mixed Dirichlent Neumann boundary conditions so in the first iteration let us see how the values change.

So we go to the interior grid points and use this formula for each grid point and try to work out the values.

**(Refer Slide Time 20:52)**

$$f_{3,2}' = \frac{f_{4,2}^0 + f_{2,2}^0 + f_{3,3}^0 + f_{3,1}^0}{4}$$

$$= \frac{400 + 200 + 200 + 200}{4}$$

$$= 250^\circ\text{C}$$


$$f_{2,3}' = \frac{f_{3,3}^0 + f_{1,3}^0 + f_{2,4}^0 + f_{2,2}^0}{4} = 200^\circ\text{C}$$

So the calculations are routine calculations only difference in this case was that in the boundary where we apply the normal condition, we have applied a finite difference expression for the derivative to compute the functional value. So that was the different from how it was for Dirichlet condition because for a Dirichlet condition the functional value was directly available. Here we have to compute it and then of course it depends on the order of accuracy of the scheme that you are using to compute it.

So in this case you have used a 3 point stencil based formula which is second order accurate. You could other seems and check for yourself how they work out when you try to use different orders of approximation. So this will turn to be 200 degree centigrade.

**(Refer Slide Time 22:01)**

$$f_{3,3}^1 = \frac{f_{4,3}^0 + f_{2,3}^0 + f_{3,4}^0 + f_{3,2}^0}{4}$$

$$= \frac{400 + 200 + 300 + 200}{4} = 275^\circ\text{C}$$


And now we are left with the point 3, 3 this will turn out to be 275 degree centigrade. This is the first equation level. And if you check with what you have got with the pure Dirichlet boundary condition the numbers would be quite different, just by changing one of the boundary condition the values would change significantly than can you check later. Let us go to the second derivation and see what happens.

If you remember the complete Dirichlet boundary condition we could reach convergence in just one step because the second iterations step essentially confirmed that we have reach convergence already. So here is it that way.

(Refer Slide Time 23:07)

2nd iteration

$$f_{2,1}^1 = \frac{4f_{2,2}^0 - f_{2,3}^0}{3} = 167^\circ\text{C}$$

$$f_{3,1}^1 = 242^\circ\text{C}$$

1st iteration

3rd iteration

$f_{2,2}^2 = 179^\circ\text{C}$	$f_{2,1}^2 = 168^\circ\text{C}$	$f_{2,2}^3 = 187^\circ\text{C}$
$f_{3,2}^2 = 273^\circ\text{C}$	$f_{3,1}^2 = 268^\circ\text{C}$	$f_{3,2}^3 = 284^\circ\text{C}$
$f_{2,3}^2 = 213^\circ\text{C}$		$f_{2,3}^3 = 217^\circ\text{C}$
$f_{3,3}^2 = 288^\circ\text{C}$		$f_{3,3}^3 = 297^\circ\text{C}$
		$f_{2,1}^3 = 180^\circ\text{C}$
		$f_{3,1}^3 = 280^\circ\text{C}$

So in the secondary iteration first thing we will do is we will update the boundary condition for the lower boundary points, at the end of the first iteration without which we would not be able to compute the secondary iteration level. So what we are doing now is essentially part of the first equation itself before we can start doing the second iteration. So this will turn out to be 167 degree in a similar manner  $f_3$  will turn out to be 242 degrees.

So this is part of first iteration itself. So what we did was that we iterated the inner grid point values in the first iteration level and use those, value to update the boundary values at the south boundary at the end of the first equation. Getting ready to initiate or secondary iteration. In the secondary iteration if you do repeat the calculations, now with the update values of the inner grid points, and the updated value of the south boundary grid points you will be able to see that this will become 179, this is 273.

And at the end of the second iteration when you update the values of the south boundary points they will turn out to be 168 and 268 respectively. And if you compare between the first iteration and second iterations level you can make out that the values have not converged. They are different in the secondary equation when compared to the first iteration which indicates that we have to go for more iteration's subsequently.

And then if you were to do another iteration let us quickly look at the results what that will produce. Third iterations would produce  $f_2$ , 2 at third iteration equal to 189 3, 2 284. Now if you compare again all of them still continue to change. Which means it will take us more iteration to converge. Now the message from this exercise essentially is that we had made a comment that if you have purely Dirichlet boundary conditions.

Then it is usually it takes less number of iterations to reach convergence compared to boundary conditions where you have mixed Dirichlet Neumann. And through this simple problem that we have set out to solve last time to the pure Dirichlet boundary condition where we could reach convergence in; just 1 iterations. Here we found even with 3 iterations we do not seem to be fairly close to the convergence.

There could more changes coming over next few rounds of iteration before we actually reach convergence. So as a practical exercise one could write a simple computer code to do these

iterations in a recursive manner. And then finally reach convergence and then check for convergence whether we are able to satisfy a particular convergence criteria. But in general presents of a Neumann boundary one of the 4 boundary conditions seem to bring in a lot of change not only to the numbers, how they vary in the domain but also on the convergence characteristic of the solution.

So you may have notice that now you may have impose the south boundary as a Neumann boundary the temperature in general are rising at the boundary. Last time it was just 0 degree centigrade at the south boundary. And this time you notice that the south boundary temperatures are way higher compare to what it was last time. So the 2 grid points at the boundary are showing 180 and 280 degree centigrade.

To reach the 0 gradient conditions the temperature needs to be rising there. It will later on saturate and then reach an equilibrium distribution but it seems to be taking more iterations.

**(Refer Slide Time 28:51)**

#### Alternating Direction Implicit scheme

$$f_{i-1,j}^{k+\frac{1}{2}} - 4f_{i,j}^{k+\frac{1}{2}} + f_{i+1,j}^{k+\frac{1}{2}} = - \left( f_{i,j+1}^k + f_{i,j-1}^{k+\frac{1}{2}} \right)$$

$$f_{i,j-1}^{k+1} - 4f_{i,j}^{k+1} + f_{i,j+1}^{k+1} = - \left( f_{i+1,j}^{k+\frac{1}{2}} + f_{i-1,j}^{k+1} \right)$$

Convergence acceleration is possible through over relaxation

So with this we complete our discussing on elliptic partial differential equations. Thank you.