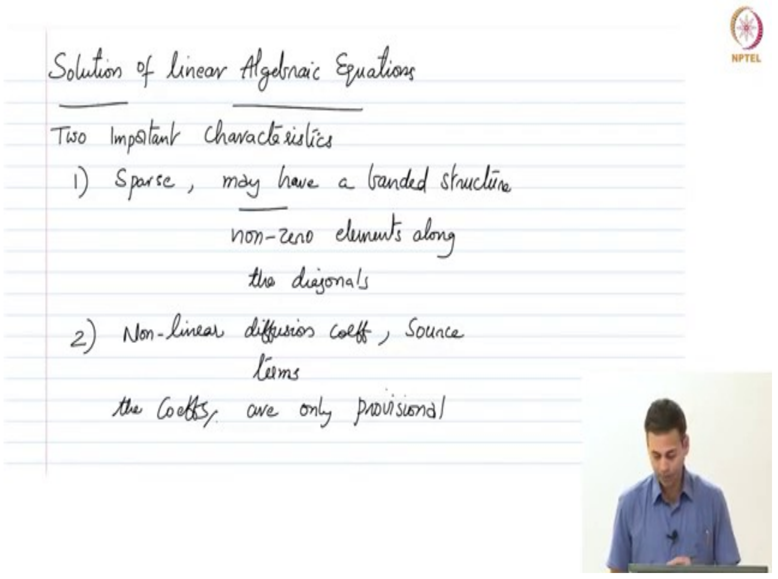
Computational Fluid Dynamics Using Finite Volume Method

Prof. Kameswararao Anupindi
Department of Mechanical Engineering
Indian Institute of Technology, Madras

Lecture – 09
Overview of Numerical Methods: Solution of linear algebraic equations

Good morning, welcome to today's lecture. So, we were discussing a Solution of Linear Equations in the last class.

(Refer Slide Time: 00:26)



Solution of linear Algebraic Equations

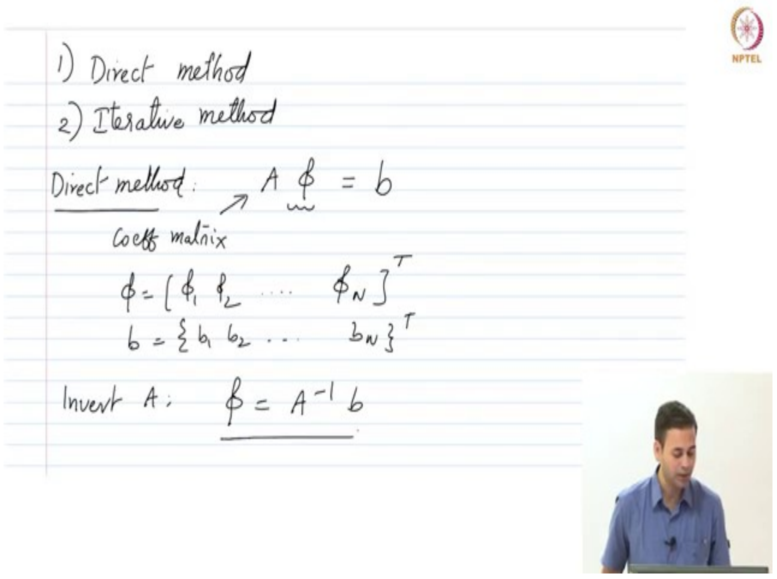
Two Important Characteristics
1) Sparse, may have a banded structure
non-zero elements along the diagonals
2) Non-linear diffusion coeff, Source terms
the Coeffs. are only provisional

So, let us get started with that, so this is a Solution of Linear Algebraic Equations. So, what we noted was there are two important characteristics that these linear equation set has; the first one is that the matrix, the resulting matrix is sparse, that means there are lot of 0s in the matrix that we assemble.

And then the, so the matrix is sparse and also it may have, may have a banded structure right; that means all the nonzero elements right, all the nonzero elements could be aligned along the diagonals of this matrix, right. So, nonzero elements along the diagonals ok, and we also noted that if we have non-linear problems, we have non-linear diffusion coefficient or source terms; then the coefficients that we get are only provisional, right.

So, these are the two main properties that we kind of noted for the matrix A, ok.

(Refer Slide Time: 02:20)



1) Direct method

2) Iterative method

Direct method: $\underset{\nearrow \text{Coeff matrix}}{A}\ \underset{\sim}{\phi} = b$

$\phi = [\phi_1\ \phi_2\ \dots\ \phi_N]^T$

$b = \{b_1\ b_2\ \dots\ b_N\}^T$

Invert A: $\underline{\phi = A^{-1} b}$

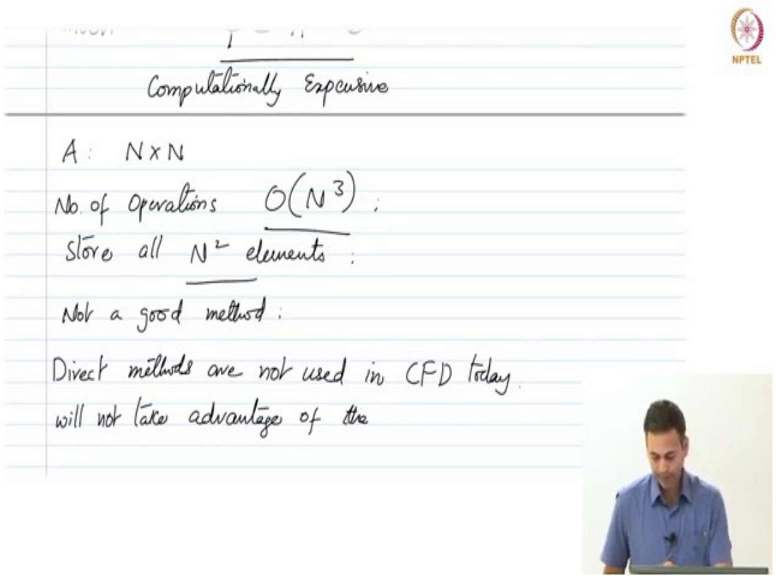
So, there are mainly two different methods that can be used to solve the system of linear equations; the first method is a direct method and the other one is an iterative method. So, let us look at each of the advantages and disadvantage for each of these methods.

So, in the direct method what we do is, essentially once you assemble these equations right; you have a system of linear equations that we can write it as $A\phi=B$ where A is the coefficient matrix. And what is phi? Phi is the vector of unknowns right; that is what we set out to solve.

So, phi is your phi 1, phi 2 and so on all the way to phi n. So, this is the vector of unknowns that we want to solve. And B is what? B is a known right hand side vector right, essentially it is a known right hand side vector that would be some b 1, b 2 all the way to b n and transpose, ok.

So, this is what we set out to solve, and then in the direct method you essentially invert the matrix, ok. So, that is what you would use; you invert A and then you want to solve for phi equals A inverse B, ok. But we realize that inversion is a very what; is it a very easy process or cheap or expensive? It is expensive right, computationally it is expensive.

(Refer Slide Time: 03:54)



Computationally Expensive

A : $N \times N$

No. of Operations $\underline{O(N^3)}$;

Store all $\underline{N^2}$ elements ;

Not a good method :

Direct methods are not used in CFD today.

will not take advantage of the

For example if we of a system of N unknowns, what would be the size of the matrix A? If we have N unknowns.

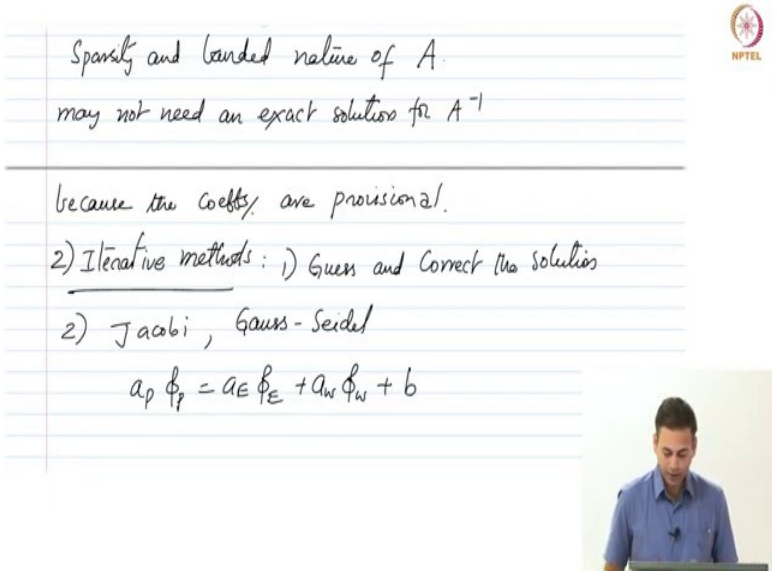
Student: (Refer Time: 04:10).

N cross N ok, so we have N by N. And what would be the number of operations involved, if we want to invert this matrix? Order of N cube ok, so that is a lot; if we have a thousand cells, then you are doing a billion operations right, 10 to the power 9. So, which is very time consuming and also we need to store all N square elements right; because out of which many of them could be 0s.

So, we will be like storing lot of 0s as well. So, store all N square elements and also do lot of operations. So, this is not a good way, at least for the matrix that we get in order to solve for phi, ok. So, not a good method; because it is not considering the sparsity or the banded nature of the matrix that we have, ok; so, that is why we do not kind of prefer direct methods.

So, that is why the direct methods are not used in simulations of let us say fluid flow and heat transfer, so in CFD today ok; that is because the direct methods will not take advantage of the sparsity and banded nature of matrix A, ok.

(Refer Slide Time: 05:40)



Sparsity and banded nature of A

may not need an exact solution for A^{-1}

because the coeffts. are provisional.

2) Iterative methods : 1) Guess and Correct the solution

2) Jacobi , Gauss - Seidel

$$a_P \phi_P = a_E \phi_E + a_W \phi_W + b$$

And also of course, we kind of discussed that the matrix A, the coefficients are provisional if we have non-linear diffusion coefficient or non-linear source terms; in which case, we probably do not need to invert this matrix exactly, ok.

Because the coefficients are anyway provisional, we could probably get away with getting an accurate answer; like an approximate answer instead of getting an exact answer. So, that is another motivation. So, essentially we, so may not need, may not need an exact solution for A inverse; because the coefficients are anyway provisional, ok. That it we would kind of go about two iterative methods; iterative methods work on guess and correct the solution.

So, the philosophy is basically guess a particular solution and then correct it and then keep doing it ok; that is the philosophy for iterative methods. Now, what are the examples of iterative methods that you know to solve for A phi equals B? Jacobi ok, so Jacobi and.

Student: Gauss Seidel.

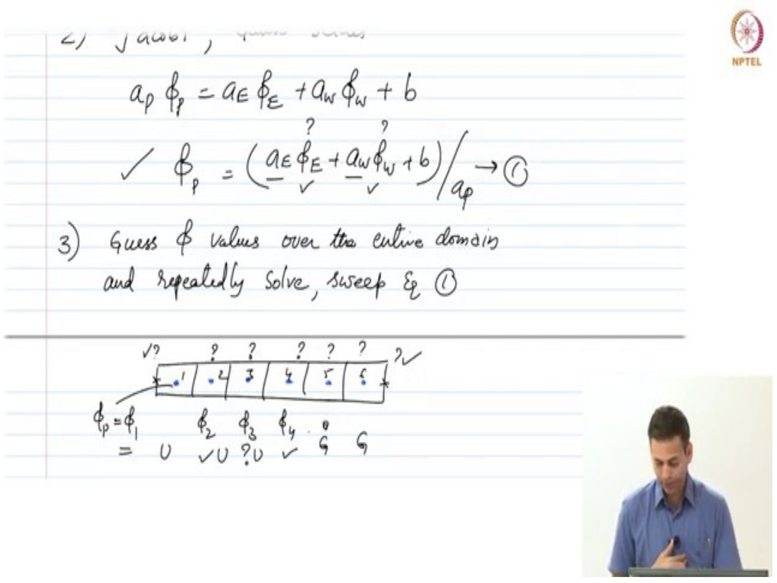
Gauss Seidel ok. So, Gauss Seidel and we have variants of this; using relaxation and things like that; somebody said tri diagonal as well, right.

Student: (Refer Time: 07:41).

Tri diagonal algorithm as well. So, what is how is it different from Gauss Seidel? It will be the same right; it will be the same, we just have a re composition and then you would do it, ok. So, well, it will not be the same in the context of Gauss Seidel right, because in Gauss Seidel you have a iterative method; whereas a tri diagonal one would be kind of a direct method, but it will take advantage of the banded nature and this sparsity, ok.

We will come to the tri diagonal matrix method later ok; once we kind of finish this discussion on iterative methods; we will come back to that, ok. So, essentially you have a guess and correct philosophy for these iterative methods. So, what we do is, we start off with the equation that we have derived; thus the equation we derived was$a_p \phi_p = a_E \phi_E + a_w \phi_w + b$, let us say if you are looking at a 1D problem a east phi east plus a west phi west plus some source term we call it let us say b, ok.

(Refer Slide Time: 08:40)



$$a_p \phi_p = a_E \phi_E + a_W \phi_W + b$$

$$\checkmark \ \phi_p = \left(\underset{\checkmark}{\underline{a_E} \overset{?}{\phi_E}} + \underset{\checkmark}{\underline{a_W} \overset{?}{\phi_W}} + b\right) / a_p \rightarrow (1)$$

3) Guess ϕ values over the entire domain and repeatedly solve, sweep Eq (1)

So, what we do is, we essentially rewrite this equation for the unknown, the unknown is the unknown for the p itself right that is what we are trying to solve, ok; $\phi_p = a_E \phi_E + a_w \phi_w + b$¿$_{ap}$ We rewrite this equation for the unknown phi p. Now what we do is, we guess phi values over the entire domain and repeatedly solve or sweep the equation one which is this one.

So, essentially you repeatedly apply this equation to cell after cell and then you keep sweeping it; by sweeping I mean, we just go from one point to another point, that is what we do in a Gauss Seidel or a Jacobi method, ok. So, how does it work? Essentially what we are talking about is we have let us say, we have let us say several cells, ok. So, we start off with. So, these are the cell centroids.

So, we apply for the first point phi p would be equals phi 1; let us say if we call it as 1, 2, 3, 4, 5, 6. So, phi p equals phi 1. So, you would guess the values of phi 2, ok. So, you would guess some values initially everywhere and then you would also guess the values on the boundaries or this could be already given depending on the boundary conditions, ok.

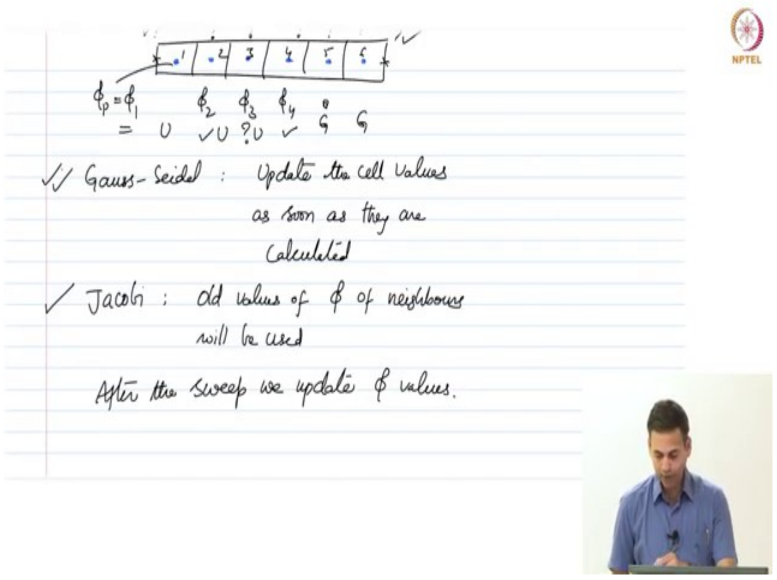
So, using that you have written one equation for phi p right, you got a linear equation algebraic equation for phi p or phi 1; you would rearrange that in this equation form of 1 and then you would solve for this using the guessed values on the right hand side, ok. These are all the guesses and you already know what is b and what is a p and the coefficients are all known, ok.

So, essentially solve for phi p and then you go to the next cell, cell 2 and then you again solve for that corresponding equation for phi 2 in terms of phi 1, phi 3, where you would use phi 1, phi 3 values right and you would solve for phi 2, ok. And then you go to phi 3, use the guessed values of these two and then solve for this, ok.

So, that is what we do in Gauss Seidel. Now in Gauss Seidel the unknowns to the right hand side or to the direction which we are sweeping are all guessed values, ok. So, these are all guesses, let us me call it G. And then all the unknowns that we have already traversed, they are all updated ok; let us call these are updated. So, U, U, U; let us say if you are solving for q 4, you will have updated values for q 3 phi 3, phi 2 and phi 1, right.

Whereas the unknowns for 5 and 6 would be phi 5 and phi 6 would be the guessed values, ok. So, as soon as you found a value for a particular cell, you would update that with what you have found, ok.

(Refer Slide Time: 11:46)


1 2 3 4 5 6
$\phi_p = \phi_1$ ϕ_2 ϕ_3 ϕ_4
= U ✓U ?U ✓ G G

✓✓ Gauss-Seidel : Update the cell values as soon as they are calculated

✓ Jacobi : old values of ϕ of neighbours will be used

After the sweep we update ϕ values.

So, that is what you do in Gauss Seidel's. So, essentially in Gauss Seidel you update the cell values as soon as you calculate, as soon as they are calculated, ok. Now that brings out the difference between Jacobi and Gauss Seidel. In Jacobi you would not update them ok; you would leave, you would still use the old values for all the neighbours, although you know that you have already calculated an updated value, ok. So, in Jacobi old values of phi of neighbours will be used and all the new values will be updated in one go after you sweep the entire domain.

So, essentially once you, so after the sweep, we update phi values ok; that is the difference between Jacobi and Gauss Seidel, otherwise it is pretty much the same ok, you go from one cell to another cells and then keep updating this. Now ok, so you may have a question; like then why do we need Jacobi right usually. So, actually it actually started in the other way around ok; it was Jacobi the first method and then Gauss Seidel was an improved method.

The way it was found that people started using Jacobi was proposed and then method was used and then they realized that; if you use the latest updated value, the convergence has a kind of that, the convergence rate has doubled, ok. So, until you would get the solution in half the number of iterations, you would need per Jacobi, ok. So, that is how this was kind of accidentally found and that is how the Gauss Seidel kind of came about from Jacobi method, ok.

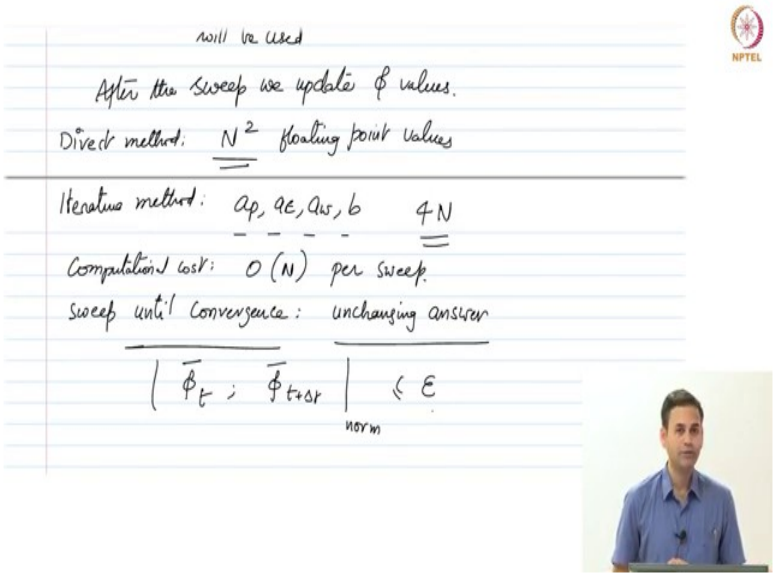
Of course, if you look at it from memory point of view; which of these methods do you think would require more memory in if you compare these two?

Student: Jacobi.

Jacobi, because you have to store the old values and the new values; whereas if you have Gauss Seidel, you just have one array of phi right and then you can just keep updating it, ok. So, in that sense Gauss Seidel is faster requires lesser memory compared to Jacobi method, alright.

Now, unlike, unlike direct methods, now these iterative methods only require, they do not need a storage of the full matrix right; all you need is you need to store all these coefficients that is a east, a west and a p and b of per cell that is all, right.

(Refer Slide Time: 14:27)



will be used

After the sweep we update ϕ values.

Direct method: $\underline{\underline{N^2}}$ floating point values

Iterative method: $\underline{a_P}, \underline{a_E}, \underline{a_W}, \underline{b}$ $\underline{\underline{4N}}$

Computational cost: $O(N)$ per sweep.

$\underline{\text{sweep until convergence}}$: $\underline{\text{unchanging answer}}$

$$\left| \bar{\phi}_t \; ; \; \bar{\phi}_{t+\Delta t} \right|_{\text{norm}} \leq \varepsilon$$

So, you do not need. So, in the direct method we have stored N square floating point values right; whereas in the iterative method how many do we need? We have a p, a east. Let us say we are talking about a 1 D problem and b right, we have these 1, 2, 3, 4 coefficients per cell and how many cells do we have? N cells right, we have N cell. So, how many do we need? 4 N that is all, right. So, this is much smaller compared to.

So, if you have let us say a 1000 cells, you just need 4000 floating point values to be stored; whereas in the direct method you have to store 10 to the power 6 right, so many elements have to be stored. What about the unknown? Unknown is the same in direct method and the iterative method right; you still have to store phi for N cells right that is anyway the common between these two, ok.

Also the computational cost is only order N per sweep; that means if you do let us say 50 sweeps you are doing 50 times N is the computational cost, whereas in the direct method it is order N cube which is huge, alright. So, what do we do? We have to kind of sweep until convergence; well we have not discussed what is convergence, but I mean we will discuss in a little while.

So, what we mean is, we basically keep updating these values right; once you are done from 1 to N you will again start from 1 to N and keep using these values until you reach a an unchanging answer right, that is what we mean by convergence in this particular context, ok.

So, until we reach an unchanging answer, now I am a kind of assuming that we are looking at a steady state problem ok; if it is transient, we will discuss that later, ok. So, that that is what we mean by convergence and or unchanging answer or we cannot really talk about an unchanging answer in the context of floating point values.

So, what we say is that, we kind of the solution that we obtained between two sweeps would differ by some epsilon, ok. For example, we have a solution vector that is phi, phi bar right, phi bar would contain basically phi 1, phi 2, phi 3 all the way to n. So, this is at a time level t and the next time level we got phi bar t plus delta t and the difference between these two in some norm ok, in some norm would be lesser than or equal to some epsilon, ok. We will kind of say then that we have reached some kind of a convergence or a steady state ok; we will discuss all these things little later, yes.

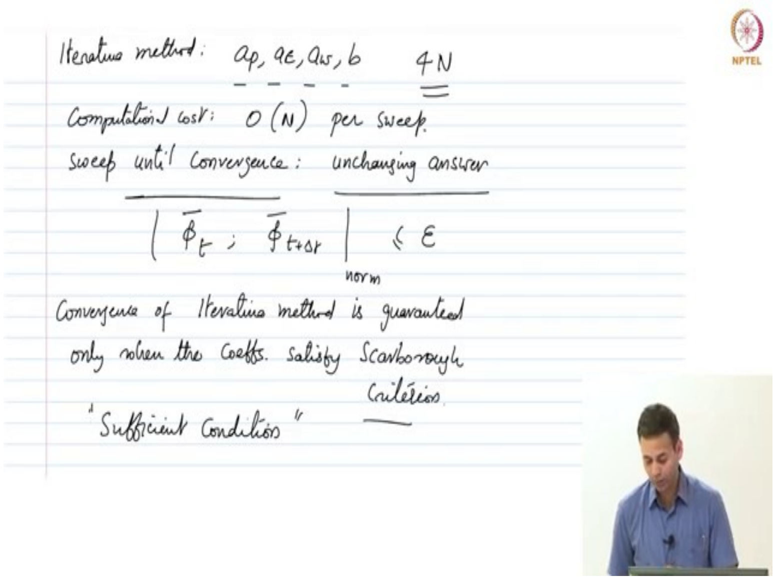
Student: (Refer Time: 17:40).

So, only for the sake of convergence you need to still store these values, only for the sake of this thing, ok. So, in that context you are storing them ok, for the sake of checking for the convergence, ok. Other questions; that is a good question. So, we said we just discussed that we do not need to store; but then we you need to store not from the point of view of the algorithm, but from the point of view of checking whether we have reached convergence or not ok, yes.

Student: How do we know that the net iteration is going to (Refer Time: 18:18) diverge?

Exactly, so that is the question that is the question we can do going to address, ok. The question is how do we know that the next iteration it is going to diverge right; how do I know that I will get some physical values out of it, ok?

(Refer Slide Time: 18:37)



Iterative method: $\underline{a_P}, \underline{a_E}, \underline{a_W}, \underline{b}$ $\quad \underline{\underline{4N}}$

Computational cost: $O(N)$ per sweep.

sweep until Convergence: unchanging answer

$$\left| \bar{\phi}_t ; \bar{\phi}_{t+\Delta t} \right|_{norm} \leq \epsilon$$

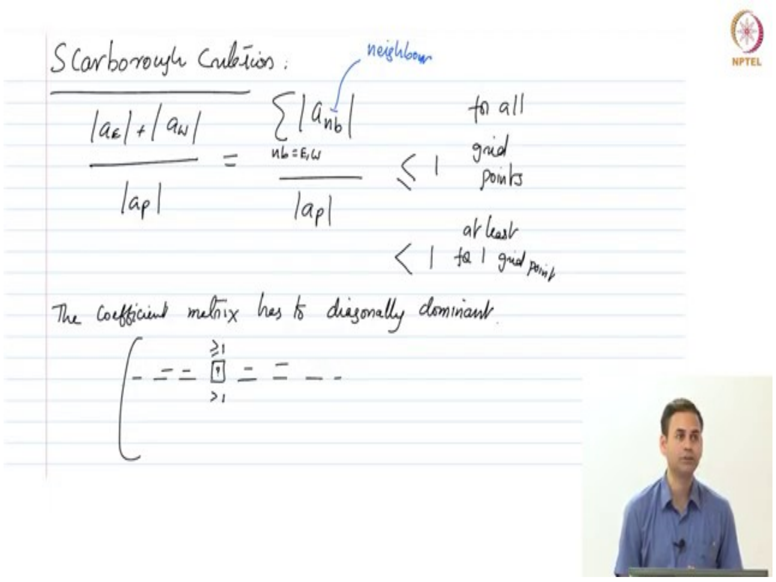
Convergence of Iterative method is guaranteed only when the Coeffs. satisfy Scarborough Criterion.

"Sufficient Condition"

So, that is the next point we are going to look at. So, essentially the convergence of any iterative method is guaranteed only when the coefficients satisfy something known as Scarborough criteria, ok. So, if the coefficients of the matrix satisfy something known as Scarborough criteria, then the convergence is guaranteed for an iterative method, ok. Now this is a sufficient condition, ok. So, this is a sufficient condition, ok.

It is not a necessary condition, it is a sufficient condition; that means even if your coefficients do not satisfies Scarborough criteria, it may still converge ok, but if it satisfies Scarborough criteria, it will definitely converge, ok. So, that is what Scarborough criteria is. Now what is Scarborough criteria, ok?

(Refer Slide Time: 19:54)



Scarborough Criterion:

neighbour

$$\frac{|a_E|+|a_W|}{|a_P|} = \frac{\sum_{nb=E,W}|a_{nb}|}{|a_P|} \leq 1 \quad \text{for all grid points}$$

$$< 1 \quad \text{at least for 1 grid point}$$

The coefficient matrix has to diagonally dominant.

$$\left(- \; - \; - \; \overset{\geq 1}{\underset{> 1}{\boxed{1}}} \; - \; - \; - \; - \right.$$

So, we will be making lot of references to this criteria as we go along. So, let us look at it. So, Scarborough criteria says that, the absolute values of the neighbours, that is absolute a E plus absolute a west upon absolute a p. So, essentially we are looking at the absolute values of the coefficients of the neighbours, their sum divided by the absolute value of the coefficient of the primary cell ok; that this I can rewrite it as absolute value of n b, where n b I would like to kind of a short term for a short form for neighbour which I will use.

So, sigma for all the neighbours; here in this particular case neighbours are east and west upon mod a p, ok. So, we have only two neighbours. So, the sum of the absolute values of the neighbouring coefficients upon the central coefficient should be less than or equal to 1 for all grid points ok, and should be less than 1 at least for 1 grid point, ok. So, that is Scarborough criteria.

So, this needs to be satisfied; if this is not satisfied, your iterative scheme may not converge ok, may not give you a, they are not it is not guaranteed to converge, alright. So, what does this mean? This means that, the coefficient matrix has to be in different terms, this has to be the coefficient matrix has to be diagonally dominant ok; that means, the terms on the diagonal should be, in an absolution should be larger than the sum of the all the off diagonal elements, right.

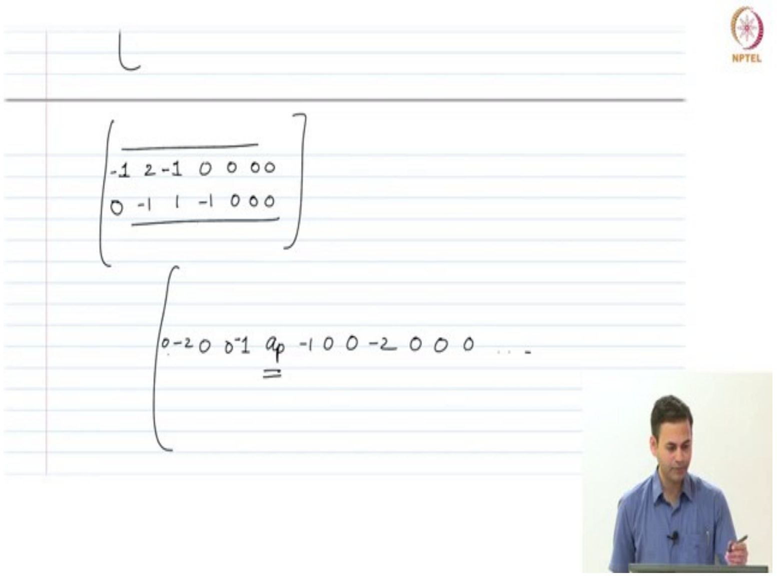
Because if you write this in a matrix form, I have some element and I have a diagonal right; let us say this is my diagonal here, then this should be greater than or equal to or greater than

1, for this this value here should be sum of all these neighbouring values, right. So, this should be much larger than or at least equal to or larger than all the neighbouring values. We know that we do not have many neighbours; we have only like 2 neighbours in the context of 1D; because everything else is 0, that is why we have this a east and a west, fine.

Questions; so do we need to check for this Scarborough criteria before we solve, once we discretize our system? Yes we need to right, otherwise it will not work right; we come back and then instead of worrying about it later, we should first check for it and then see whether it is scad is satisfies Scarborough or not, ok. We will do that for pretty much all the discretization that we will do in this particular course.

You can, yes you can make the matrix. So, the question is if a particular discretization does not you will like diagonally dominant coefficient matrix, can I still work with this? Yes you can, there are certain ways with which you can kind of improve the diagonal dominance of a particular system; we will see that little later ok, it uses of source terms and things like that, ok. Other questions, diagonal dominance, ok. So, let us say, question is explain the diagonal dominance again?

(Refer Slide Time: 23:33)



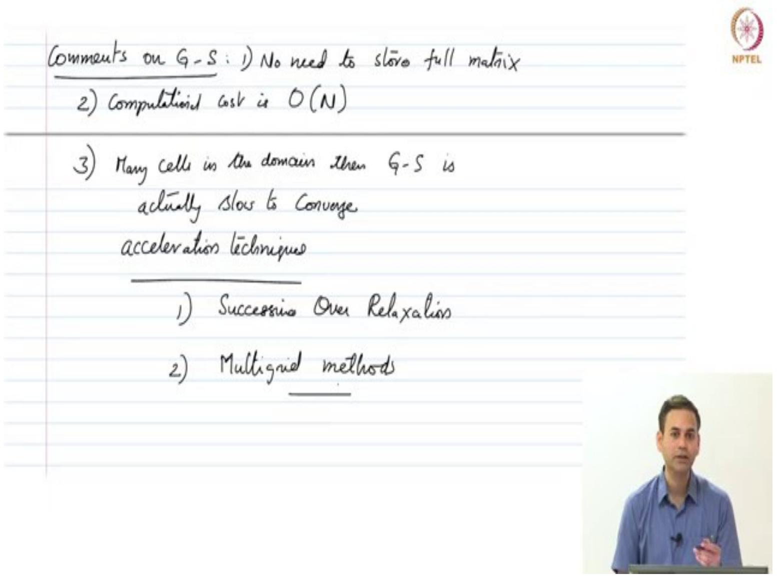
So, essentially what we mean is, for example, I have a set of rows, ok. So, I am looking at the second row ok; second row the first element is a off diagonal element, let us say it is 1 ok, this is a diagonal term 2 and this would be 1 0 0. So, this is diagonal dominant, because 2 is equal to the sum of the neighbours, right. What about this? Is this diagonal dominant? Yes, it is still diagonal dominant, because the absolute value of this diagonal coefficient is larger than this.

So, similarly you can have, you can have any value equal to 2 or greater than 2 to be diagonal dominant; this cannot be lesser than the diagonal off diagonal element. So, if you have a term which is, let us say the next term is 0, minus 1, 1, minus 1, 0; this is a problem, ok. You will not, if you get like this; then you are not satisfying Scarborough at one cell, ok. Now this only says that it is not guaranteed to converge, but it may still converge, ok.

It is only a sufficient condition. Yes, so the question is if we have more neighbours what shall we do? We have to take all the neighbours, ok. So, for example, let us say we go to two dimensions, then our matrix, if you put in the matrix; then you have, let us say I am looking at some interior row ok, where I have this is my a p term and then I have some nonzero terms here. So, I have some nonzero term here which is let us say 1, I have 0, 0 let us me call minus 1, minus 2, 0; then I have then I have minus 1, 0, 0, minus 2, 0, 0 and so on, right.

Then what would be a p? A p should be greater than or equal to 1 plus 1 plus 2 plus 2 right; it should be greater than or equal to 6 right, sum of all the elements in that row, ok. We will come to that when we do the 2 D diffusion equation. Other questions; no clear, ok, fine. So, let us have kind of few comments on the Gauss Seidel method.

(Refer Slide Time: 25:39)



Comments on G-S : 1) No need to store full matrix

2) Computational cost is $O(N)$

3) Many cells in the domain then G-S is actually slow to converge

acceleration techniques

1) Successive Over Relaxation

2) Multigrid methods

So, few comments are, no need to store the full matrix right; we just need to store only the nonzero elements, ok. So, computational cost is only order N, fine. So, but if we have a large cells; not large cells, we have, we have let us say many cells in the domain, then Gauss Seidel is actually kind of slow to converge, ok. It is kind of slow to converge, this is kind of a negative point for Gauss Seidel; but we will look at few techniques which will accelerate it ok, accelerate the convergence.

If we have a large number of cells, for example let us say you have hundred cells, Gauss Seidel might work quickly; but if you have let us say thousand cells or a let us say million cells, then Gauss Seidel will take a long time to converge to within that epsilon, ok. So, as a result this is slow to converge, we will look at some acceleration techniques later on in the course; these are one is the successive over relaxation and we will also look at something known as multi grid methods.

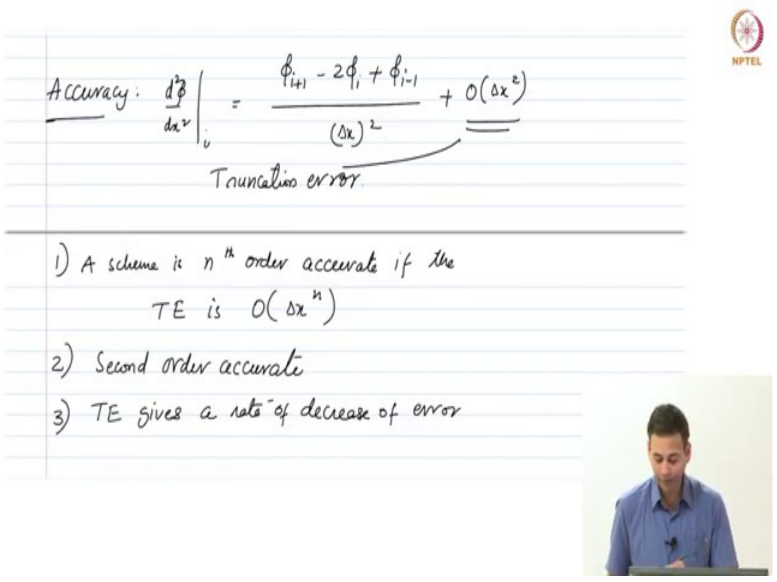
These will help to alleviate the problem of slow convergence of Gauss Seidel, ok. So, now will you be able to code for a Gauss Seidel with this discussion or no? Essentially we have this equation a p phi p equals sigma all the neighbouring terms plus some b right, we basically have a x equal to b. All you have to do is you have to go to every cell and then update that cell unknown value, update that cell phi value from the neighbouring phi values which are guesses, right.

You could start off with phi equal to 0 everywhere right, and phi could be specified on the boundary cells. So, we have one discrete equation for each of the cells. Let us say if we have five cells in the domain, we have five equations; you visit each and every equation and update it, right. And then once you keep updating it in another loop, you would have; you would check for is my previous values are very different from the new values I have obtained.

If it is very different, you go back keep doing this until they are within certain epsilon ok; that is all you have to do. So, basically two loops kind of; one loop is to go from cell to cell, another loop outside this one would be where you check difference between these two in an individual manner or in a collective manner, right. We will look at a example problem after that I think you will get a little more confidence; but that is the philosophy of Gauss Seidel, fine.

So, let us look at few concepts that kind of arised yesterday, which was related to accuracy, consistency, stability and convergence.

(Refer Slide Time: 29:07)



Accuracy: $\left.\frac{d^2\phi}{dx^2}\right|_i = \frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{(\Delta x)^2} + \underbrace{O(\Delta x^2)}_{\text{Truncation error.}}$

1) A scheme is n^{th} order accurate if the TE is $O(\Delta x^n)$

2) Second order accurate

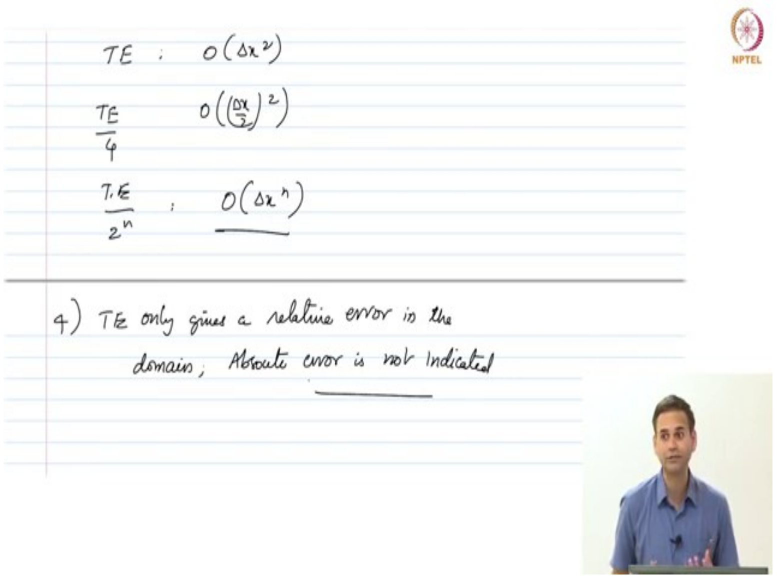
3) TE gives a rate of decrease of error

So, let us look at what is known as accuracy? Accuracy is we have defined using the Taylor series expansion right; for example, if we have $\frac{d^2\phi}{dx^2} = \frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{(\Delta x)^2} + 0(\Delta x^2)$ at certain point i, we had an expression yesterday in the context of finite difference methods, what where we wrote it was as plus we said we have neglected something of the terms order delta x square, ok.

This we have called it as what, truncation error; because we are truncating these terms, ok. This is truncation error. Now accuracy is basically defined by the truncation error that we have, ok. So, essentially if we have a scheme is nth order accurate, if the truncation error is order delta x to the power n, ok.

That means this particular scheme is a second order accurate; of course now the truncation error or accuracy, the truncation error gives a relative error decrease or we can say, we can say rate of decrease of error, ok. So, it is not truncation error term or the accuracy of the scheme you have used, gives something known as rate of decrease of error, ok.

(Refer Slide Time: 31:12)



TE : $O(\Delta x^2)$

$\frac{TE}{4}$ $O((\frac{\Delta x}{2})^2)$

$\frac{T.E}{2^n}$: $O(\Delta x^n)$

4) TE only gives a relative error in the domain; Absolute error is not Indicated

What does that mean? That means, if I employ my truncation error is of the order delta x square, ok. Now what will be my truncation error? If I take a cell, all the cells in my domain are now delta x by 2 right; if I take let us say if I have 5 cells, I will make it a 10 cells ok, that means each of my delta x is now halved right, smaller and smaller. So, what will happen to truncation error? It will go; it will decrease or increase?

Student: Decrease.

It will decrease by one fourth right; essentially it will become $\frac{TE}{4}$, right.

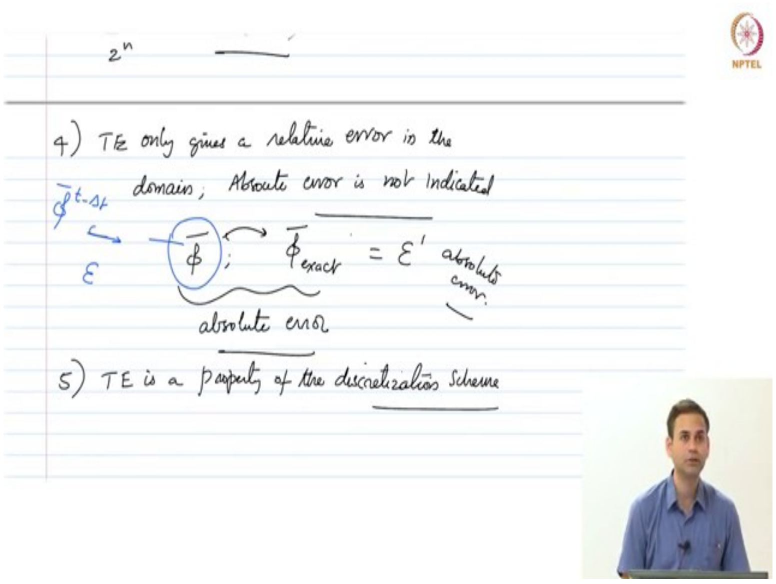
So, essentially if you half the grid; the truncation will go down by one fourth, if you have a second order scheme, right. Similarly if I halve the grid; the truncation will go by 1 by 2 to the power n, if I have an nth order scheme right and so on.

All this is of course, assuming the delta x is less than 1 ok, fine alright ok. So, what will happen to; so that means, my truncation error would go down by 2 to the power n, if my order of the scheme is delta x to the power n, right. If I halve the grid, it will go down by one fourth and so on ok, so fine.

So, that is the about the truncation error. Now this only gives, the truncation error only gives a relative error in the domain; it does not talk about the absolute error, ok. So, the absolute error is not indicated; this only tells you if you kind of refine your cells or if you make them bigger, this is how the error that you are committing would increase or decrease.

Now, what about the absolute error? How do we check for that? In order to check for absolute ever, we should know the exact solution right; only then I can check for the absolute error, right.

(Refer Slide Time: 33:23)



2^n

4) TE only gives a relative error in the domain; Absolute error is not indicated

$\bar{\phi}^{t-\Delta t}$ ε

$$\underbrace{\bar{\phi}\,;\quad \bar{\phi}_{exact}}_{\text{absolute error}} = \varepsilon' \quad \text{absolute error}$$

5) TE is a property of the discretization scheme

That means if I know, let us say I have found some phi bar in a domain; this is what I computed from my finite volume method, then if I know what is phi exact, I can go back and then calculate phi exact at those grid points and then find the difference between these two that would give me something known as absolute error,.

So, truncation error does not talk about anything related to absolute error ok; it only tells you as you refine your mesh your truncation error will go smaller and smaller, and that means you are committing lesser and lesser error,. Now this truncation error is a property of the discretization scheme; that means if we the discretization scheme, instead of second order if you go for third order accurate, you have the truncation error much smaller, ok.

It could be, let us say if we have a fourth order accurate scheme; then your truncation error will be of the order of delta x to the power 4, ok. So, that is why this is a property of the discretization scheme, fine ok.

So, truncation error is basically the question is, explain the difference between truncation error and the absolute error? Truncation error only gives you a error in the context of in a relative sense or rate of decrease of error. For example, you take a mesh of delta x size, you make it you make it delta x by 2; then the truncation error goes down by one fourth right, if we have a second order scheme.

That is what it tells you; but if you use a second order accurate scheme and you converge to some solution right and that solution is let us say phi bar. So, this is the solution you have converged, this phi bar you have converged to by comparing phi bar with phi the previous time step ok. Let me call it as t minus delta t bar and you found that the difference between these two is less than some epsilon that we have said.

Now, you say that we have reached convergence; but when you go back, somehow you know there is an exact solution, ok. You go back and compare this converged solution with the phi exact; then you realize that there is some error between these two, this could be let us say some epsilon 1, ok. Now this is your absolute error, fine.

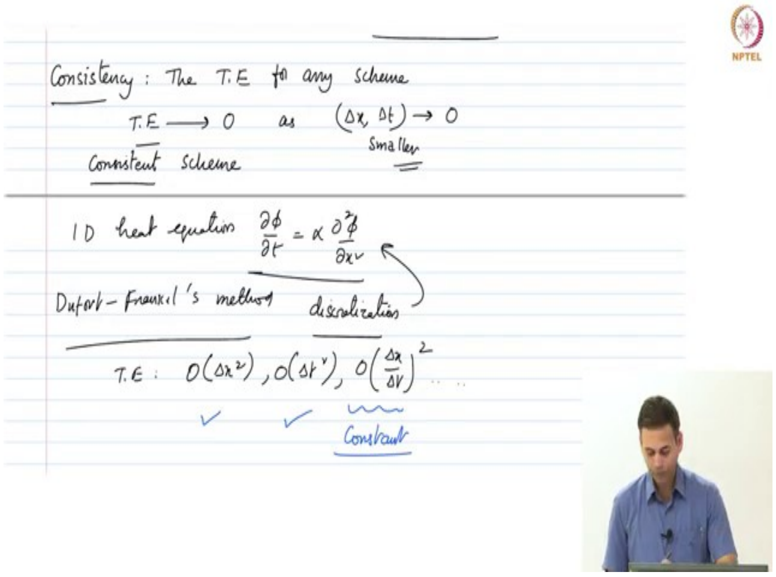
Now one if you want to take, if you realize that this epsilon 1 is much larger than what you had anticipated; what you would do? You either go back and change your cell size, you would make it smaller right and reiterate and again converge to some value and then you compare again, right. Or you could have instead of decreasing your delta x, you could use a higher order accurate scheme, right.

Because on the same delta x, a higher order accurate scheme will have lesser truncation error and thereby can give you little more accurate result, ok. So, there are two ways to obtain this; one is to keep decreasing delta x, or keep going to a higher and higher order accurate scheme on the same grid, right.

Of course both have their own differences and advantages, we will discuss that later; is that clear. We will probably do this as part of the assignment ok, we will have an exact solution; we will compare and then see difference between absolute error and the relative error. Other questions please; no, alright.

Then let us move on to the next concept that is accuracy is done, and we look at something known as consistency.

(Refer Slide Time: 37:04)



Consistency : The T.E for any scheme

$T.E \longrightarrow 0$ as $(\Delta x, \Delta t) \rightarrow 0$

Smaller

Consistent scheme

1D heat equation $\frac{\partial \phi}{\partial t} = \alpha \frac{\partial^2 \phi}{\partial x^2}$

Dufort – Frankel's method discretization

T.E : $O(\Delta x^2), O(\Delta t^2), O\left(\frac{\Delta x}{\Delta t}\right)^2 \ldots$

✓ ✓ Constant

So, all these are we are again looking from the context of overview of numerical methods ok; we will come back to accuracy, consistency all these things when we actually do the finite volume method for diffusion, convection diffusion and so on, ok. We will revisit that; here we are only kind of giving an introduction to what these mean, so that I can freely use them you know without you getting doubts in your minds, ok.

Essentially I, if I say accuracy and truncation error you will be able to kind of connect to them and we will visit these later, ok. We will use a Taylor series and kind of calculate the particular values, ok. Now what is consistency? Have you heard of this term consistency? Consistency is used to denote that the truncation error that we have for any scheme, if it goes to, if the truncation error tends to 0; that means, it becomes smaller and smaller as your delta x or delta t are made smaller and smaller, ok.

So, as you go delta x refine the values of the delta x delta t, your truncation error that is what you expect right; the truncation error should go to 0. So, the error this should go to 0. Now how do we know that? We know that from the scheme we have used, ok. So, if a scheme displays this property, then we say that the scheme is consistent, ok. Now you may have a question; well all the schemes will be consistent is not it, why should we care about inconsistency in this context? It will not be, some of the schemes are not in not consistent, ok.

For example. So, what does this mean? This means that your delta x as you refine it, your truncation error is not, is decreasing then it is fine; that means, you have obtained a solution on five cells, you got some solution. Now you have solved it on let us say hundred cells, you got a solution that is worse than the previous one right; that can only happen if the scheme used is not consistent right, that is not expected.

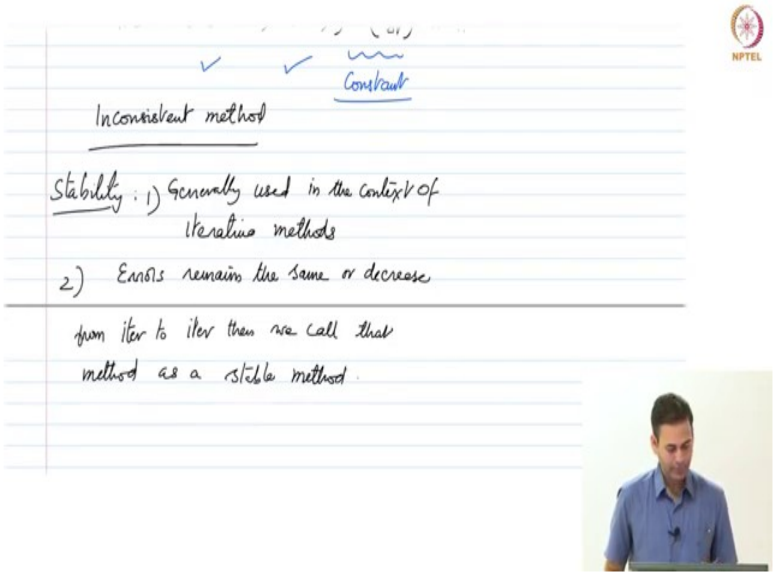
So, this can happen if let us say for example, we are not deriving this thing; I am just stating it as of now. Let us say if we take the one dimensional wave equation, 1 D wave equation that is first order or let us say I will take the heat transient, heat conduction equation, ok. So, I will take the 1 D heat equation, ok. So, essentially $\frac{\partial \phi}{\partial t}=\alpha \frac{\partial^2 \phi}{\partial x^2}$

If I use a particular method, which is known as Dufort Frankel's method; so this is a particular method, it has its own discretization way of discretization ok, this is a particular discretization, it is called Dufort's Frankel method. If we use this method for the 1 D transient heat conduction equation; the truncation error you would get would have terms which will have, which will contain$o\left(\Delta x^2\right), o\left(\Delta t\right)^2, o\left(\frac{\Delta x}{\Delta t}\right)^2$.

So, now do you see the problem? You see the problem right, essentially the first term is fine, second term is fine; but there will be some terms which will have this ratio of delta x by delta t, this is not good, right. That means, if you keep refining your delta t and delta x in such a way at the same rate; then this error will remain the same right, order delta x by delta t will remain the same. And even though you are refining your delta x and delta t, your truncation error will not tend to 0, right.

It will only tend to a particular constant value and you cannot get rid of that error from your solution, ok. So, such schemes are inconsistent and we do not want such a scheme, ok. We will probably do this later on as an example.

(Refer Slide Time: 41:15)



Constant

Inconsistent method

Stability : 1) Generally used in the context of
iterative methods

2) Errors remain the same or decrease
from iter to iter then we call that
method as a stable method.

So, that is an inconsistent method, ok. Questions; now you may ask like why ok, then why do we care about this method? Like you know the thing is it was proposed, it was used and later on it was found that it is inconsistent method ok; that is how this has come about. But there are certain other properties which are useful for this method; that is why it is used by knowing that this is an inconsistent method, ok.

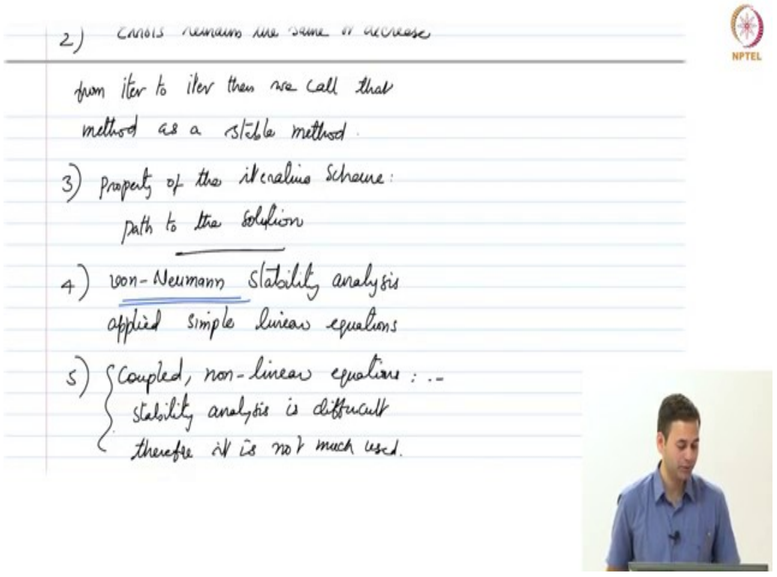
But we want to work with consistent methods. So, we want the truncation error to go to zero as we refine our delta x and delta t, fine. Let us look at something known as stability, ok. Stability is generally used in the context of iterative schemes or iterative methods. Any guesses on what is stability? I we just said that it is used in the context of iterative methods.

Student: Converges.

Converges, essentially we do not want the errors to grow from iteration to iteration, right.

If the errors remain the same or decrease, then we call a particular iterative method as stable, ok. So, if the errors remain the same or decrease from iteration to iteration; then we call that method as a stable method, alright.

(Refer Slide Time: 43:12)



2) errors remains the same or decrease

from iter to iter then we call that method as a stable method.

3) property of the iterative scheme:

path to the solution

4) von-Neumann stability analysis

applied simple linear equations

5) Coupled, non-linear equations : ..

stability analysis is difficult

therefore it is not much used.

So, essentially this is a property of the iterative scheme, we are employing; that means this is a, this is related to the path to the solution ok, that means it depends on the path we are taking based on the iterations, ok.

So, we mean path basically we are going to a set of intermediate values of phi ok, that is the path we are talking about here. So, property of the iterative scheme; so we want only; we want only of course work with the stable methods or stable schemes such that our iteration, our errors do not grow from iteration to iteration.

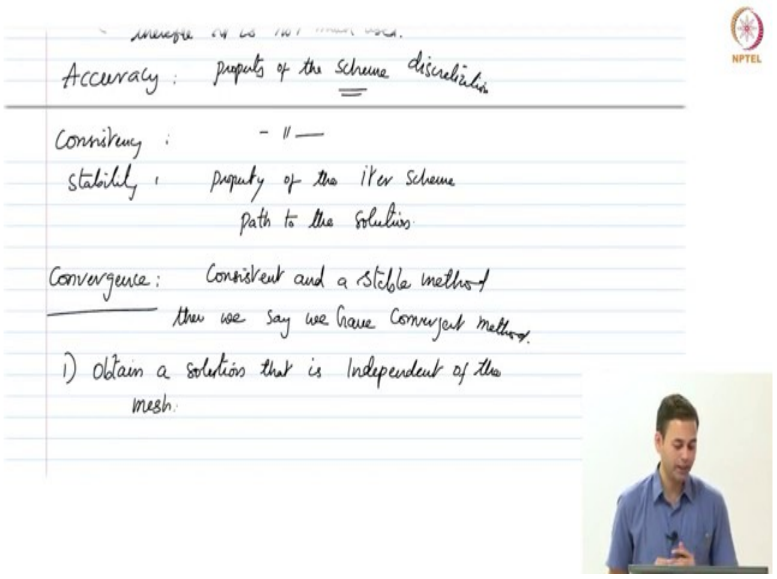
Now how do we know whether the iterations are, whether the errors are going to go grow or decrease? There is one classic tool that is known as a von Neumann stability analysis that can be applied to simple linear equations ok; simple linear equations, simple linear differential equations ok, which will tell you whether the particular scheme is stable or unstable and so on, ok.

We will use, we will be doing stability analysis little later after we kind of discuss the diffusion equation, ok. But if you have real life problems, let us say if you have a coupled non-linear equations and so on; then the stability analysis is difficult, therefore it is not much used, ok. So, we have to kind of work with some linear counterparts of the non-linear equations we are working with, and then come about with some kind of measures, which will hopefully give a solution to our coupled equations and so on, ok.

Otherwise this gets very very difficult to know whether a particular method is stable or not, ok. So, that is why there are no general methods that can be used for knowing the stability of coupled non-linear equations, ok. So, we kind of work with the counterpart equations, linear equations or with some experience we would kind of guess them, ok. So, but we will doing a von Neumann stability analysis later on after the diffusion is done, ok.

Now, we kind of did not see. So, the, we said the accuracy is a property of the scheme right; accuracy is a property of the scheme. And the, what about convergence; is convergence also a property of the scheme sorry, consistency?

(Refer Slide Time: 46:18)



Accuracy : property of the scheme discretization

Consistency : – " —

stability : property of the iter scheme

path to the solution

Convergence: Consistent and a stable method

then we say we have convergent method.

1) obtain a solution that is Independent of the mesh.

So, we said, we said accuracy is a property of the scheme, right. Similarly we said consistency; what about consistency? Excuse me; is it also a property of the scheme? It is a property of the scheme; because if we use a higher order accurate scheme, so this is also a property of the scheme.

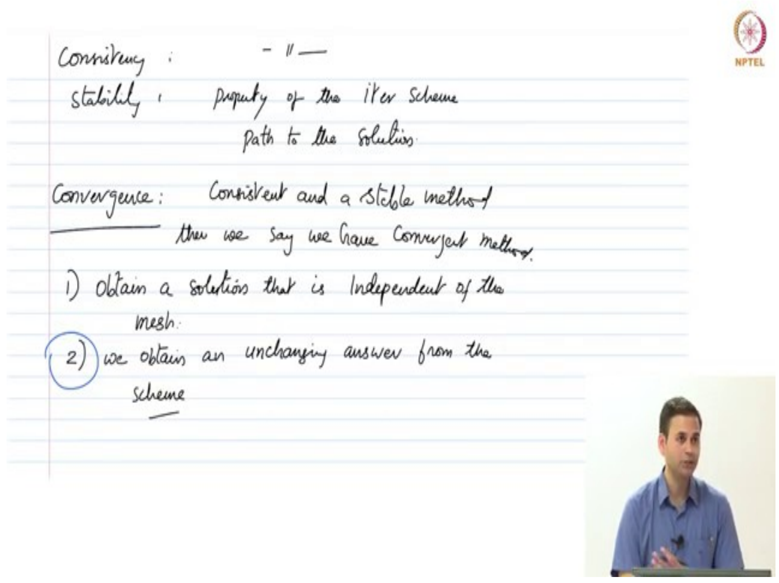
What about stability? Stability is a property of the iterative method right, not the discretization scheme. So, these are for the discretization scheme ok, this is a property of the iterative scheme that we use, or it is basically a path to the solution, right. So, which path do you take in order to obtain a stable solution, right? So, essentially stability is measures the path you are taking; you may take a path which will diverge, then it is not stable, right.

It has certain do with the discretization scheme you have used, you could use a second order scheme; but if you use an unstable method in the time stepping scheme, then it may kind of not converge, ok.

Now we will come to the final thing known as convergence; convergence basically if you have a consistent and a stable method ok, then we say that we have, then we say we have a convergent method. That means we should have a consistent and a stable method to have a convergence scheme, everything we work with will be convergence schemes.

Now, convergence also has different, couple of different meanings; one of them, one of them is basically obtaining a solution, obtain a solution that is independent of the mesh, ok. You might have heard of this term grid independency or mesh independent solution; that is also in some context convergence is used to indicate that the method converges as you take different finer meshes, ok.

(Refer Slide Time: 48:46)



Consistency : — " —

stability : property of the iter scheme
path to the solution

Convergence: Consistent and a stable method
then we say we have convergent method.

1) obtain a solution that is independent of the mesh.

2) we obtain an unchanging answer from the scheme

And in another context convergence means that, we obtain an unchanging answer from the scheme; that means converging to an unchanging answer, ok.

So, in this particular course, we will be using convergence in the context of the second one ok; that means if we reach an unchanging answer, that means we say that we have converged our solution, ok. That is what we use in this particular course; but you would see that people use convergence in the context of obtaining a solution that is independent of the mesh.

For example you have taken 100 cells, 1000 cells and 2000 cells; the solution you obtain from 1000 and 2000 may not be very different ok; that means we have converged the solution onto a particular mesh independent solution. So, these are the four concepts accuracy, consistency, stability and convergence. All these properties we would make use of as we go along, and also the Scarborough criteria that we have discussed in the context of iterative schemes, ok.

Thank you.