

**Computational Fluid Dynamics**  
**Prof. Sreenivas Jayanti**  
**Department of Chemical Engineering**  
**Indian Institute of Technology, Madras**

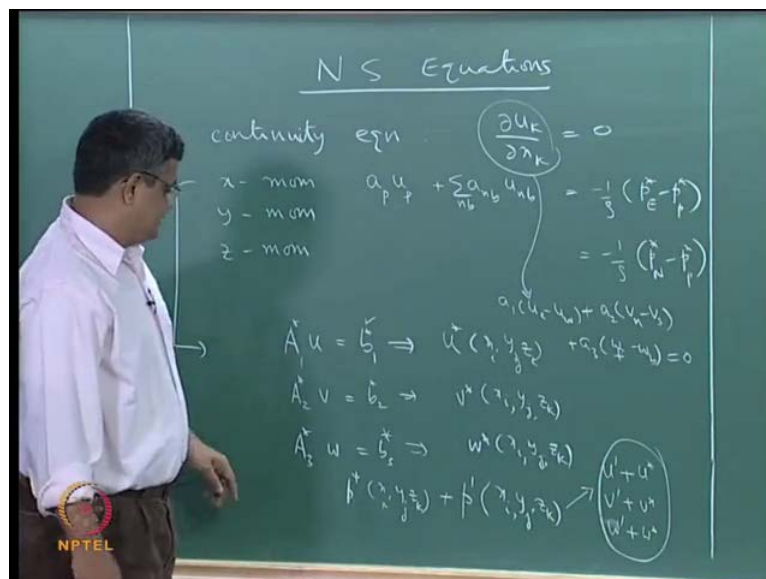
**Module No. # 05**  
**Solution of linear algebraic equations**  
**Lecture No. # 09**

**Need for efficient solution of linear algebraic equations**  
**Classification of approaches for the solution of linear algebraic equations**

Let us now take a look at the complete flow sheet for the solution of the Navier-Stoke's equations, so that we have **an** a good idea of not only how we are solving together but what is actually needed to get a solution. We are of course looking at first of all the steady case and we can see how that is going to be changed. When we look at unsteady case, where we want to get a time accurate resolution of the velocity field and the pressure field for a general three-dimensional flow.

We are restricting ourselves as of now to Cartesian flow geometry. In the next part of the course, we will look at the general case of complicated geometry, which does not necessarily fit in a Cartesian coordinate frame. So, we will look at an outline of the calculation procedure using the simple algorithm. Let us recall, what the simple algorithm is.

(Refer Slide Time: 01:18)



We have the Navier-Stokes equation for incompressible flow consisting of the continuity equation, which we can write as  $\sum_k \rho u_k = 0$ . We have the x momentum equation, y momentum equation and z momentum equation and each of these will have terms representing the rate of accumulation of x momentum or y momentum, the z momentum. The net advection of the x momentum and the contribution of the pressure gradient and the contribution of the viscous forces, these are the terms that appear in this. In the method, a simple method, we have a discretized x momentum equation, which can be written in this form discretized about point p for the velocity at that particular point plus  $a_{nb} u_{nb}$  where  $n_b$  represents the neighbouring points. What those neighbouring points depends on the specific discretization scheme that we adopt for the advection term and **and** for the diffusion term.

If we take upwind scheme, you have one set; if you take a central scheme, you have another effect and if you take a second order or third order upwind, you have a different set of neighbouring points. So, depending on what kind of discretization scheme we adopt, we will have different neighbouring point, but the point that we are making is that the velocity at a particular node will have contribution coming from the neighbouring points, the effect of which is linearized into this coefficient.

So, we are looking at a **linearized x momentum equations** the linearized discretized x momentum equation, in which the effect of the neighbouring contributing points coming from the advection and diffusion terms is incorporated already here and this is written as **the** **as** you have the pressure gradient term. So, we can write that as  $1 \text{ by } \rho_p \epsilon - \rho_p$ .

So, this is the discretized form of the x momentum equation for a particular point. If we write it for all the points, then we can **can** write a corresponding matrix equation  $A_1 u = b_1$ , where  $b_1$  **ah** contains all the terms coming from the pressure gradient terms here and  $A_1$  has the component for this and then the coefficients for the neighbouring terms also. We notice that, if we want to solve this equation, then we need to know the neighbouring values and not only that for the general case, we **this** these coefficients here, this linearized coefficient also contain the velocity components in the y momentum and z momentum.

So, in that sense, this  $A_1$  here contains approximations from linearization and approximations from the current estimate of the values of the v and w components and the x momentum equation.

So, we have this and we can solve this provided, we know all the coefficients of  $a_1$  and all the coefficients of  $b_1$ . So, therefore, because we have linkage of the x momentum with the y momentum and the z momentum through the velocity components, which are entering in this linearized, in these coefficients here and also because of the pressure which is coming here, we **we** start the solution with a guest pressure field. When we say guest pressure field, we are saying that pressure at every point; wherever pressure is evaluated is specified.

You may start with 00 or 11 whatever it is or you can take the inlet estimate and the outlet estimate of the pressure and you can do a linear interpolation. So, whatever it is, you start with a guest pressure field here and once you specify the guest pressure field, then  $b_1$  is known and  $a_1$  is already known for some sort of linearization with some assumed values of **ah**  $u$  and  $v$  **ok**.

So, now we are in a position to solve this and when we solve this we get  $u$  at all the points at which we have applied this formula here, this template here. So, we know  $u$  as a function of  $x$ ,  $y$  and  $z$  or  $x_i$ ,  $y_j$  and  $z_k$ , at those discrete point at which  $u$  is going to be evaluated. Now, what this  $u$  is evaluated based on a guest pressure field, so we call this as  $u$  star. This is not a fully correct one because this had some guest values of pressure and also some guest values of  $v$  and  $w$  and similarly we can discretize the y momentum equation to get  $2v$  equal to  $b_2$ .

Even in the y momentum equation, you have a corresponding contribution from pressure coming as  $p$  nought minus  $pp$  like this and the same pressure field that we have used in the y momentum, in the x momentum is also used here to get this and this is based on a star thing. This is also based on a starred velocity field, especially for the unknown components **ah**.

(Refer Slide Time: 08:34)

$$\begin{aligned} A_1^* u &= b_1^* \Rightarrow u^*(x_i, y_j, z_k) \\ A_2^* v &= b_2^* \Rightarrow v^*(x_i, y_j, z_k) \\ A_3^* w &= b_3^* \Rightarrow w^*(x_i, y_j, z_k) \end{aligned}$$

The image shows a chalkboard with three equations written in white chalk. Each equation is of the form  $A_i^* \text{velocity} = b_i^* \Rightarrow \text{velocity}^*(x_i, y_j, z_k)$ . The first equation is for  $u$ , the second for  $v$ , and the third for  $w$ . In the bottom left corner of the chalkboard, there is a small circular logo with the text 'NPTEL' below it.

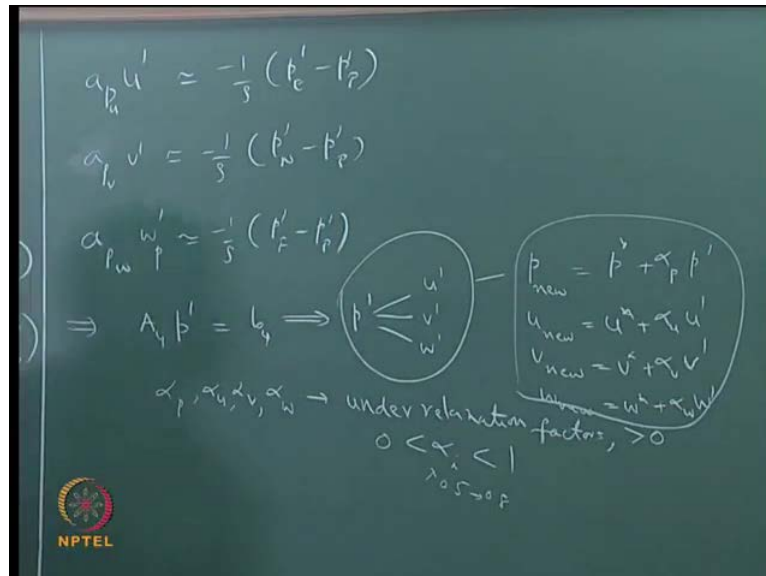
So, we solve this to get  $v$  star again as a function of  $x_i$ ,  $y_j$  and  $z_k$ . We noticed that, if you are using a staggered grid, then the  $x_i$  coming in the  $u$  momentum equation is different from the  $x_i$  coming in the  $v$  momentum equation. We can do the same thing for the  $z$  momentum and we can come up with  $A_3 w$  equal to  $b_3$ , which is the linearized form of the  $z$  momentum equation with the guess pressure field we evaluate the  $w$  to get  $w$  star at  $x_i$ ,  $y_j$  and  $z_k$ .

Now, we introduce a pressure correction, the  $p$  star of  $x$ ,  $y$ ,  $z$  again at discrete points  $x_i$ ,  $y_j$ ,  $z_k$  is now being supplemented with a guess with a correction, which we call as  $p$  prime and this again is to be specified at  $y_i$ ,  $x_j$ ,  $z_k$ .

As of now, this is not known. We want to introduce this pressure correction in such a way that this pressure correction introduces velocity correction in the three-velocity components:  $u$  prime,  $v$  prime and  $w$  prime, such that the new pressure field, the new velocity field that is  $u$  prime plus  $u$  star,  $v$  prime plus  $v$  star plus,  $w$  prime plus  $w$  star is the new velocity field satisfies the continuity equation. So, the condition for evaluating the pressure correction is that the induced velocity corrections in this will satisfy the continuity equation, the discretized form of the continuity equation. We can write this as, let us just put it as  $a_1$  times  $u_e$  minus  $u_w$  plus  $a_2$  times  $v_n$  minus  $v_s$  plus  $a_3$  times  $w_f$  minus  $w_b$  equal to 0, where this  $u_e$   $u$ 's are the new velocities and this we know are all  $x_i$ ,  $y_j$  and  $z_k$ . So, you can put these things.

So, based on the substitution of these things, we can get a **get a** modified continuity equation in terms of the known velocities  $u$  star,  $v$  star,  $w$  star and  $u$  prime,  $v$  prime and  $w$  prime.

(Refer Slide Time: 11:29)



So, that equation cannot be solved directly because we do not know about this  $u$  prime,  $v$  prime and  $w$  prime are. So, what we do in the simple method is that based on **on** this discretization, neglecting the contribution of the neighbouring points, we say as a first approximation this  $u$  prime,  $a$  prime,  $ap_u$  prime is roughly equal to minus 1 by  $rou$   $p$  prime  $e$  minus  $p$  prime  $p$ . Similarly, **ah** we will call this as  $ap_u$  for the  $u$  equation and  $ap_v$  for the  $v$  prime equation is roughly equal to minus 1 by  $rou$   $p$  prime north minus  $p$  prime  $p$  and  $ap_w$  is the component, is the coefficient of the  $w_p$  is roughly prime is equal to minus 1 by  $rou$   $p$  front minus  $p$   $p$ . So, we use these approximate relations to get  $u$  prime,  $v$  prime and  $w$  prime and once, these are substituted into this equation along with the  $u$  star, then we get an equation; a pressure correction equation which can be written as  $a_4 p$  prime equal to  $b_4$  with all the coefficients of  $b_4$  are known and all the coefficients of  $a_4$  also can be derived.

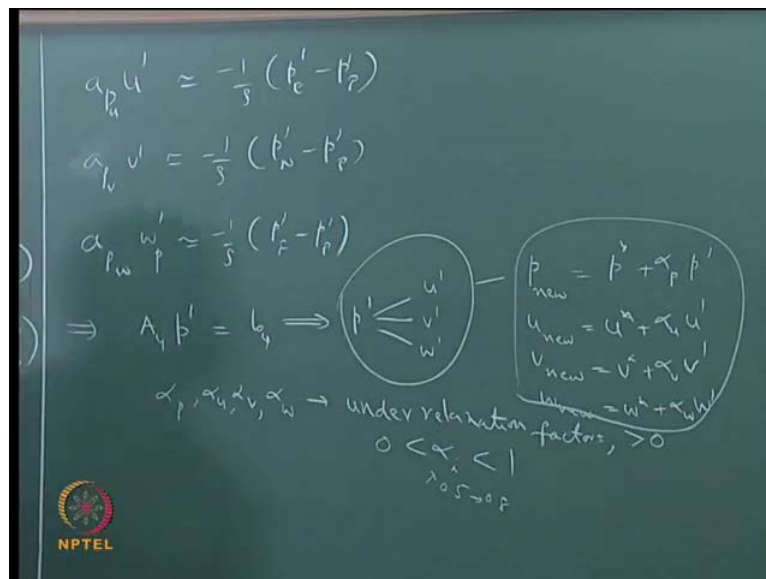
So, this we can solve for  $p$  prime and once we get  $p$  prime here, then from these relations we evaluate  $u$  prime,  $v$  prime and  $w$  prime from these approximate relations. So, once we have **these** all these corrections to the pressure and velocity field, then from these things we derive  $p$  new as  $p$  star plus  $alpha$   $p$  times  $p$  prime. Similarly,  $u$  new is  $u$  star plus  $alpha$   $u$  times  $u$  prime and  $v$  new is equal to  $v$  star plus  $alpha$   $v$  times the estimated  $v$  prime and  $w$  new is  $w$

star plus alpha w times alpha w where, alpha p, alpha u, alpha v and alpha w are under relaxation factors.

They are positive, greater than 0 and they lie between 1 between 0 and 1. If you make them 0, then that means that p new is the same as p old p star. If you make them 1, we are doing the entire thing. Once we do this, then it is **ah** known that the resulting iterative scheme does not converge. So, we typically have them as of the order of 0.5 to 0.8 like this **ok**.

So, what we have done is through this discretization here, making the approximations for this influence of the pressure correction on the respective velocity corrections on a simplified form of the **momentum** discretized momentum equation; we are able to go from the old values to the new values.

(Refer Slide Time: 15:40)



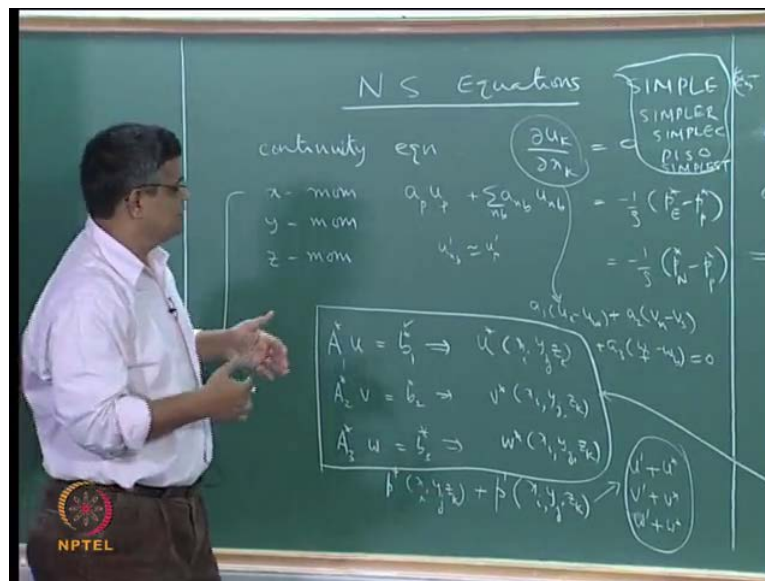
So, these new values are not fully correct because these are based on a guest velocity, guest pressure field and also the coefficients that are coming in the a1 in the discretized individual momentum equations have some estimated values of y and u, v and w **ok**.

Similarly, this one will have estimated values of u and **ah** w and so on. So, this is not necessarily fully correct. It would not satisfy the momentum equations and the continuity equation. It would not satisfy the continuity equations because these are less than 1 and also because these are only estimated values.

So, we have to start this process of evaluating these with the new values. So, we are starting with this and then we go through this process of the pressure correction equation and then get the corrections for the four components here and then we come to this and then we go back to this. With the new values of pressure and velocities, we re-evaluate b1, we re-evaluate the coefficients of a1 and of course b2 and a2 and b3 and a3. We again solve for new values here.

We put these things into this and into this to get new a4 and b4. We again solve for p prime and get, we can do this iteration several times. So, and this is how we use, we solve the Navier-Stokes equations, using the method which is known as simple semi-implicit method for pressure linked equation. The semi implicitness idea is coming from the fact that the velocity correction for a pressure correction are **are are** obtained by neglecting, not by completely solving this but only by neglecting the contribution of this. So, it is only partly implicit. This part is not implicit, this is in a way explicit step, but the fact that we are solving this together several times through this makes it that we are looking at an overall implicit scheme, but part of which is explicit **ok.**

(Refer Slide Time: 16:43)



So, this method is what we want to suggest as the method for the simultaneous solution of the Navier-Stokes equation for incompressible flows.

There are methods, whereby this assumption that is neglecting this is avoided. For example, you can say that the velocity correction in the neighbouring points is roughly the same as the velocity correction point at this. Instead of completely neglecting this, you can make a

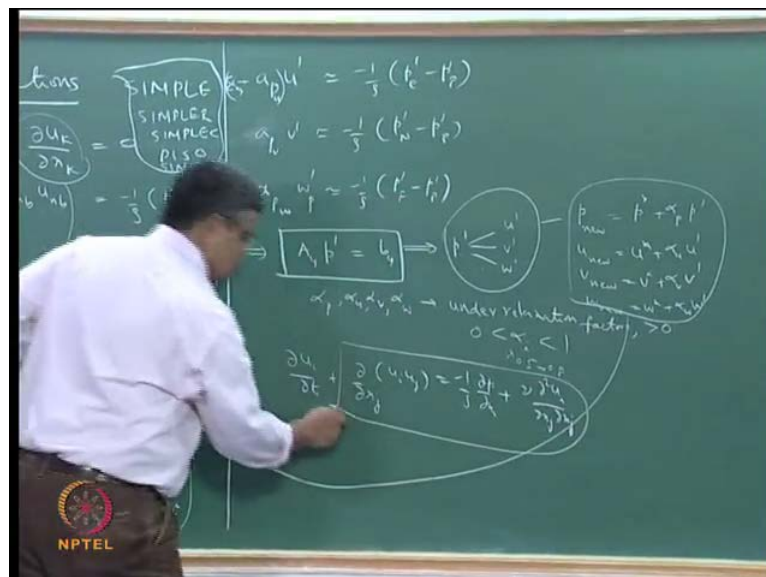
contribution of this term which is equal to this. So, you can say that instead of  $u$  and  $b$ , you can say  $u'$  and  $b'$  here, so that  $u$  and  $b$ ,  $u'$  and  $b'$  is equal to  $u + u'$  and  $b + b'$ . So, that **the** this expression here will become  $x$  plus  $a$  and  $b$  **ok**.

So, that gives you a different estimate for this  $u'$ . So, we can do those kind of things and you can also having known this  $u'$ ,  $v'$  and  $w'$  here, we can now come back to these estimate here. We can substitute these values in this and get a different estimate of  $u'$  **by** from this equation and then we can put it back again in to the pressure correction equation and you can do it.

There are many variants that are possible in **in** this process and they are Simple, Simpler, Simple C, PISO, Simplest. These are all variants of the same idea with modifications centred around the evaluation of the pressure velocity corrections coming from the pressure corrections. How many times you do this before, you do this entire over this outer iteration and these are the kind of things, but essentially this is one good way of solving the Navier-Stokes equations.

So, with this we can claim that we know how to solve the **the** Navier-Stokes equation.

(Refer Slide Time: 21:21)



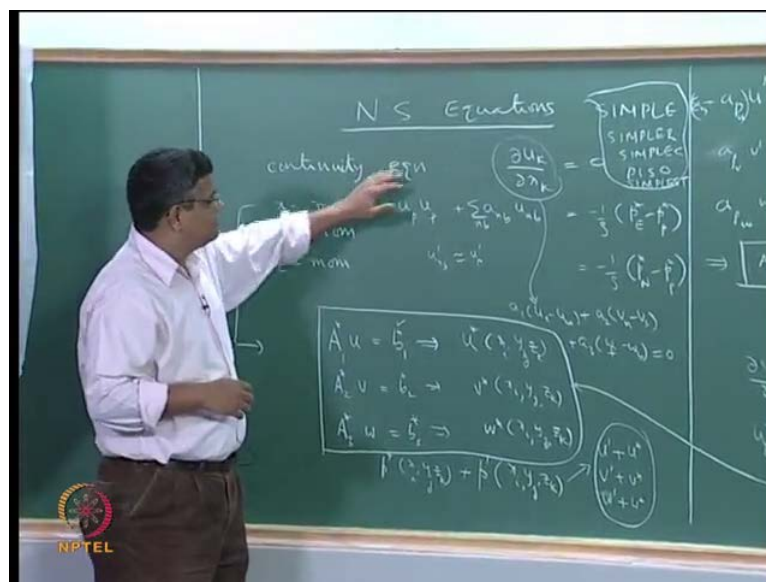
Now, what we have looked at is for a case of steady flow. If you have unsteady flow, then you **you** have an equation like  $\frac{du_i}{dt} + \frac{\partial}{\partial x_j} (\rho u_j u_i) = -\frac{1}{\rho} \frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j^2}$ .



So, far we have looked at this part. For this part alone, we have gone through this and we can write this whole thing as  $u_{i,n+1} - u_{i,n}$  by  $\Delta t$  depending on what kind of discretization scheme plus the discretized form of this whole thing is equal to minus of the discretized form of  $\frac{\partial u}{\partial t}$  this thing like this.

So, this becomes also a part of this up and this becomes part of the unknown coefficient term on this side. So, in **in** an unsteady flow, the discretized momentum equation will change slightly. The  $a_p$  term will not only have contribution from the advection term and the diffusion term; it will also have a contribution from the unsteady term and the right hand side will have contribution from the previous time steps value at that particular point.

(Refer Slide Time: 23:04)



So, that is the only modification to the discretized form of the momentum equations. With the continuity equation is exactly of the same form for steady and unsteady flows. So, when we are doing an unsteady calculation using a simple method, then the momentum equations will be modified slightly, but they still are of the same form that is, we have  $a_1 u$  equal to  $b_1$  where,  $a_1$  has contribution from the main point plus these things and the contribution here is now the same contribution that we have from advection diffusion term plus for example, this will give us  $1$  by  $\Delta t$ . So, you will have  $a_p$  plus  $1$  by  $\Delta t$  will be coming here and here, we will have plus  $1$  by  $\Delta t$ ,  $1$  by  $\Delta t$  times  $u_{i,n}$  will be coming here.

So, in this way we have a modification to the coefficients here, but otherwise the solution method will be the same and if we follow the solution method, then the ultimately the

velocities that we get will be the velocities at the new time steps. Then with those time steps, we have to come back to the discretized form and go from  $n + 1$  to  $n + 2$  and  $n + 2$  to  $n + 3$  and so on until we stop the calculation.

So, in an unsteady calculation, we have to go through the solution of this iterative solution of a guess pressure field, estimated velocity field, estimated pressure and velocity corrections to the new values. This kind of looping, we have to do several times at each time step, at the end of which we get the new values for the flow variables at that time step and then, we have to go to the next time step.

So, solution method is not very different and so the extension of the simple method that we have looked at for steady flows to unsteady flows is relatively straightforward. So, with this we can claim now, that **we have we know** we know how to solve; not only we know what equations to solve, but we also know how to discretize them appropriately, so that we get a satisfactory solution. Not only that, we know how to club them together especially for incompressible flows in such a way that we can through a lot of effort we can get the velocity field and pressure field for a given transient three-dimensional case. All that is not finished yet unless, we have an efficient way of solving a matrix equation of this type of form.

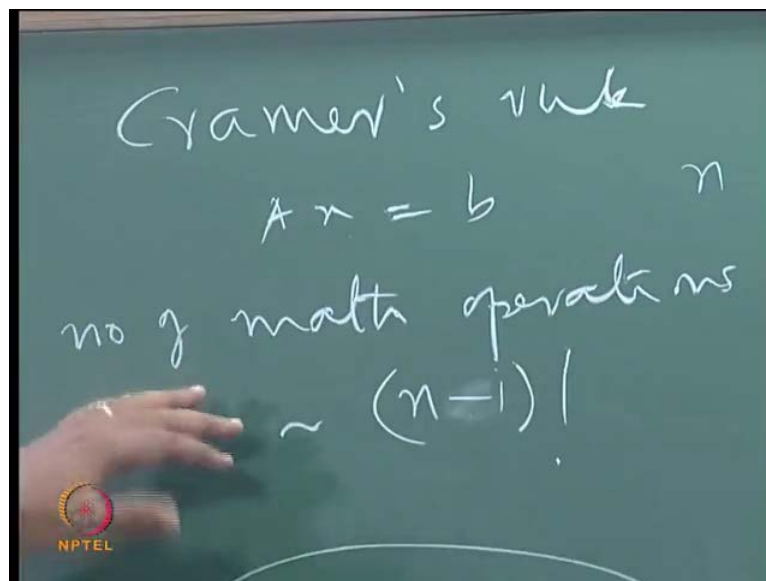
We need to have an efficient scheme because in each step from going from star quantities to the new quantities, we have to solve four matrix equations and we have to solve them many times and each time with an updated  $a_1$ ,  $a_2$  and  $a_3$ . So, we are not solving the same **ah** matrixes all the time.

So, that puts some extra care that we have to do to make sure that we have an efficient method and in a time dependent case, we have to solve them many times at each time step to make forward. So, the computational effort for a general transient three-dimensional case is not trivial and it will be extremely prohibitive, unless we have an efficient way of solving this. So, that is why it is important. The second reason why this is an efficient way of solving these things is important because the accuracy of the discretization given that we are not solving the exact equations in cfd; we are solving only the discretized form of the equations.

So, the accuracy of the discretization depends on the truncation terms and we know that the truncation terms depend on  $\Delta x$ ,  $\Delta y$ ,  $\Delta z$  and  $\Delta t$ .

So, the smaller the delta x is, the smaller the delta t is, the more accurate will be the approximation of the governing equation and therefore, the more accurate will be the computed solution. So, we need to make sure that we have sufficiently fine grid, a fine discretization in order to get an accurate estimation of the flow variables, but for the same domain if we want to have finer grid, then we have to put a more number of points, so that it will reduce the delta x. When delta x is reduced, we have more number of points, the size of this matrix will become large and once this becomes large, the computational time required for the solution also becomes large.

(Refer Slide Time: 28:37)



Just to give an example, if we use the Cramer's rule for the solution of  $ax$  equal to  $b$  with  $n$  as the number of variables, then the number of mathematical operations is  $n$  plus 1,  $n$  minus 1 factorial. So, when  $n$  is 3, 4 like that, then **it is** it becomes good method, but when  $n$  is of the order of 100, 1000, 10000, 10000000 these are not unheard of numbers in terms of the number grid points and that is the kind of number grid points that you need to have for a large domain grid, where we want to get an accurate solution.

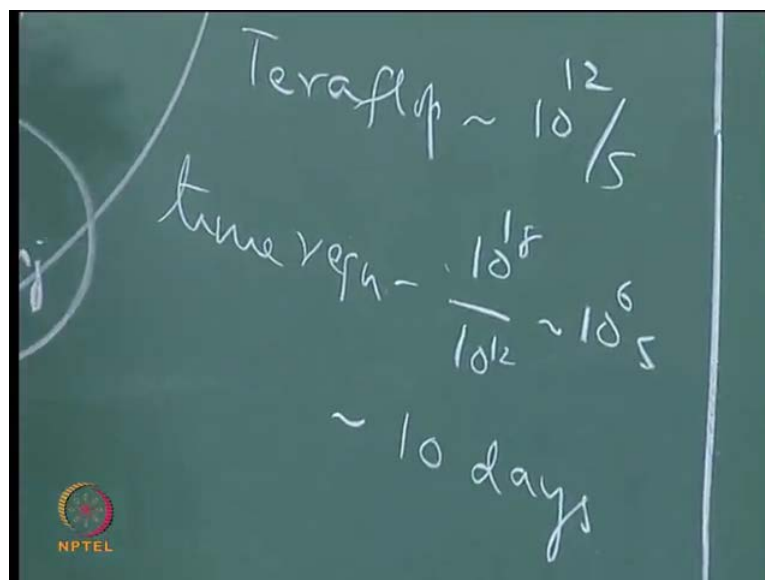
So, from that point cfd is the root method of **ah** getting a solution to the Navier-Stokes equations and **it** thrives only when you have raw computing power **ok**.

So, that is why as the computing power becomes more and more we can drive, we can go for more and more accurate solutions. So, if you want to have an accurate solution, then we need to have a computer which can do lot of mathematical operations. When  $n$  is large, this

becomes humongous. There are other methods like Gaussian elimination method, we will look at them then where the number of mathematical operations varies as  $n^3$  ok.

So,  $n^3$  is not as prohibitive as this, as factorial as the  $n$  factorial and, but even this can be very large. For example, if you take  $n$  million so, number of grid points as million, then the number of mathematical operations that you have will be, let us call this as capital  $N$ , so capital  $N$  will be of the power of  $n$  to the power of 6 cube. So, that is of the power of 10 to the power of 18 mathematical operations are needed.

(Refer Slide Time: 31:04)



Teraflop  $\sim 10^{12} / 5$   
time reqn  $- \frac{10^{18}}{10^{12}} \sim 10^6$   
 $\sim 10$  days

So, when you say mathematical operations multiplication division like that type of thing. A good computer may be, until a few years ago a good teraflop machine was considered to be a good fast number crunching machine. A teraflop machine can do 10 to the power operations per second, so 10 to the power of 12 mathematical operations per second.

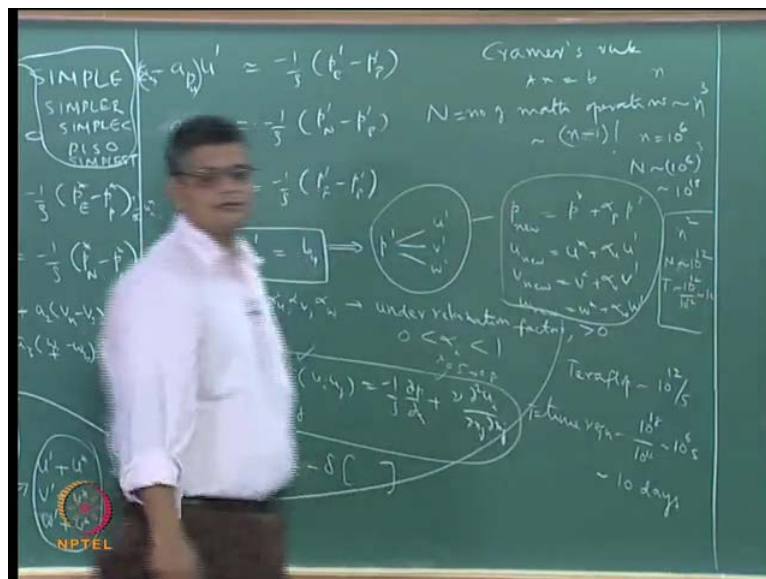
So, if you were to use a teraflop machine to solve a million grid points, then the time required will be 10 to the power 18 divided by 10 to the power 12. So, that is 10 to the power 6 seconds and how much is 10 to the power 6 seconds, it is about 10 days.

So, if you have a grid of one million and if you are using that a teraflop machine to solve this  $a1u$  equal to  $b1$ ; this computation alone would take 10 days. Then we have to do this and then we have to do this another day, another 10 days **10 days**. So, that is about one month for getting this. Then we have this one, that is another 10 days and after 40 days, we have to, we

get a new velocity field and then we have to come back and do this. It easily takes a lifetime of one particular person to get a solution for **for** a decent **(())** ok.

So, when you have the number of operations varying as n cubed, then as n becomes larger and larger, the number of mathematical operations becomes even more **ah** larger at a much faster rate. So, that is why we need to be very vary of making the problem size large, but it is necessary to have large number of grid points in a cfd computation and these should be distributed all over the control volume, so as to get a proper solution in a general three-dimensional case.

(Refer Slide Time: 33:13)



In such a case, we have to have a method which is much faster than much more efficient than n cubed in order to get a decent computational time. So, if **if** you had a method in which the number of mathematical operations varies as n square, so if it is n square, then the number of mathematical operations required for the same million grids will be about 10 to the power 12 and the time required for one computation will be 10 to the power 12 divided by 10 to the power 12. So, that is one second which is much more **ah** affordable. So, in 1 second we can solve million equations, another second another second four seconds for the four values and we can do thousands of those things, then we can get a solution in 4000 seconds say it is about an hour.

So, **how the** how efficient the computation is? How efficient the computation of the solution of this matrix equation is very important. If it varies as the number of operations varies as n

cube which is considered as an efficient method which is characteristic of for example, the Gaussian elimination method. That is not good enough for us because that would take a very large number of time; even for a single equation. So, if you have  $n$  square, then we can do easily one million operations, one million grid points even one million is probably not too high.

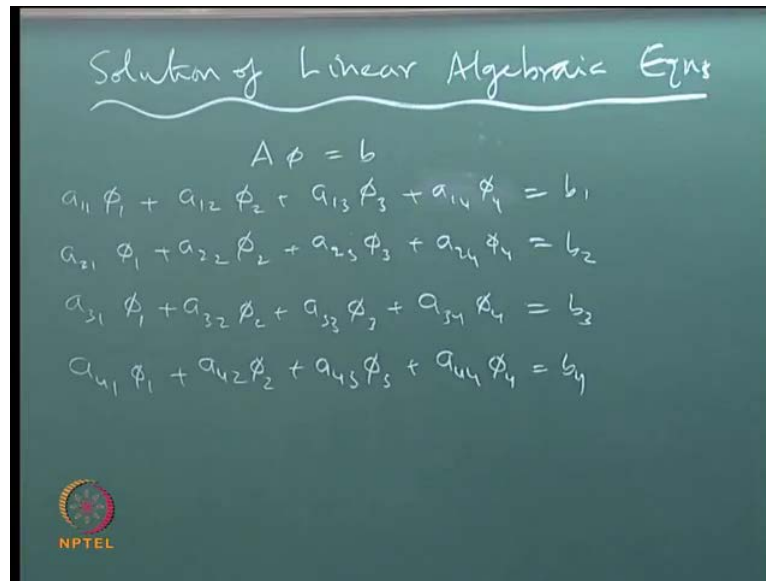
These days people are running with even ten million or even with, even in some cases Navier-Stokes equation are done even with much more larger number of grid points because the more the computing power you have, the more realistic your representation of complicated geometry, the more the number of components physical components that are affecting the flow domain that can be considered.

So, the search for efficient and more efficient algorithms is continuing and these days, we have methods in which the mathematical operations varies almost linearly with **with the the** number of nodes that are to be resolved.

So, it is important to have a good grasp of, a good command of how to solve these equations together. The conventional method for the solution of simultaneous equations  $Ax$  equal to  $b$ , which is the Gaussian elimination kind of thing with  $n$  cube number of operation is not good enough. So, we have to have better methods and that is what we are going to do.

So, we would now like to identify a method of solution of  $Ax$  equal to  $b$  in a reasonably efficient way, so that once we club it with the overall computational scheme that is described here, then we will have a prescribed, a complete method for the numerical solution of the Navier-Stokes equation. So, without an efficient method for the solution of  $Ax$  equal to  $b$  equations, the method is not useful, the method is not complete.

(Refer Slide Time: 36:47)



Solution of Linear Algebraic Eqns

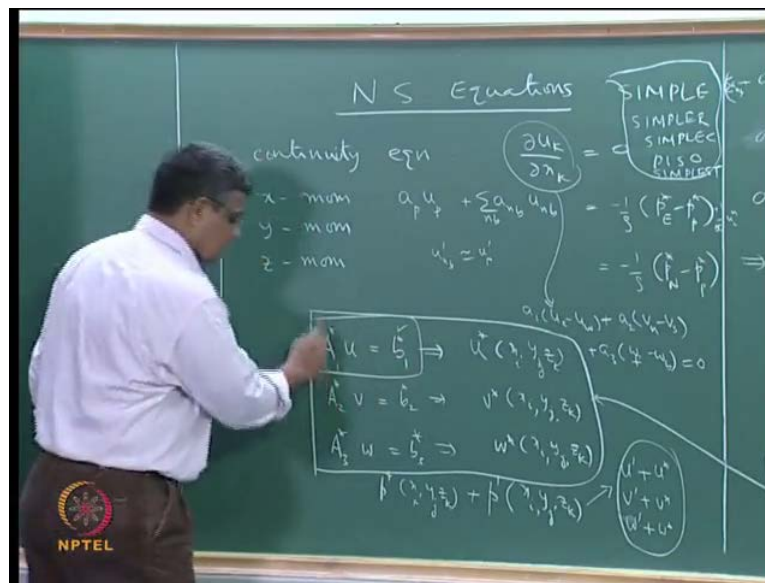
$$A\phi = b$$
$$a_{11}\phi_1 + a_{12}\phi_2 + a_{13}\phi_3 + a_{14}\phi_4 = b_1$$
$$a_{21}\phi_1 + a_{22}\phi_2 + a_{23}\phi_3 + a_{24}\phi_4 = b_2$$
$$a_{31}\phi_1 + a_{32}\phi_2 + a_{33}\phi_3 + a_{34}\phi_4 = b_3$$
$$a_{41}\phi_1 + a_{42}\phi_2 + a_{43}\phi_3 + a_{44}\phi_4 = b_4$$

NPTEL

So, we look at the solution of linear algebraic equations. What we mean by linear is the type of equation like  $a_{1i}u_i = b_1$ , in which all the elements of  $b_1$  and  $a_{1i}$  are known coefficients; they are numerical values.

We see the origin of these things. They are coming from the pressure gradient in the simple case, they are coming from the pressure gradient and may be from the previous time step value here and  $a_{1i}$  comes from **ah** the diffusion equation and also from the advection term in the simple laminar flow case. This constitutes a linear; this constitutes a non-linear term, whereas this constitutes a linear coefficient.

(Refer Slide Time: 38:26)

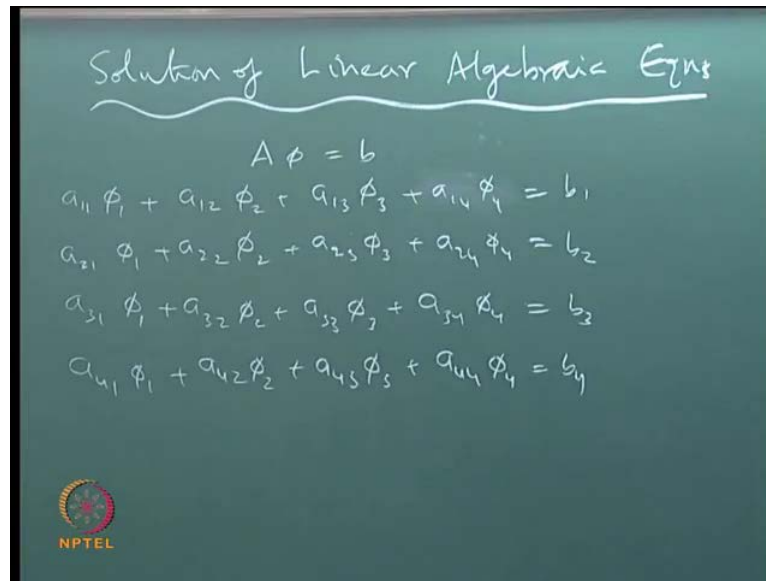


So, in a simple laminar flow case with constant property, then this will only give us linear values constant coefficients whereas, **this can give rise to** this can give rise to first of all non-linear coefficients. For example, we will have  $\frac{du}{dx}$  by  $\frac{du}{dx}$  of  $u$  square and it will also give rise to undetermined variable coefficients from iteration to iteration because, when you have  $\frac{du}{dx}$  by  $\frac{du}{dx}$  of  $uv$ , then which appears in the  $x$  momentum equations, the discretization for  $u$  here will have an assumed value of  $v$  **ok**.

So, that assumed value of  $v$  will come in as a part of that coefficient. So, the next time around when you come here, that assumed value will change because you got a new value of  $v$ . Therefore, from one iteration to another iteration the **the** coefficient here changes, but we are looking at when we come to the solution of this, we are looking at a linearized momentum equation. Therefore, the value all the elements of  $a_1$  and  $b_1$  are known and we have a linear equation.



(Refer Slide Time: 36:47)



Solution of Linear Algebraic Eqns

$$A\phi = b$$
$$a_{11}\phi_1 + a_{12}\phi_2 + a_{13}\phi_3 + a_{14}\phi_4 = b_1$$
$$a_{21}\phi_1 + a_{22}\phi_2 + a_{23}\phi_3 + a_{24}\phi_4 = b_2$$
$$a_{31}\phi_1 + a_{32}\phi_2 + a_{33}\phi_3 + a_{34}\phi_4 = b_3$$
$$a_{41}\phi_1 + a_{42}\phi_2 + a_{43}\phi_3 + a_{44}\phi_4 = b_4$$

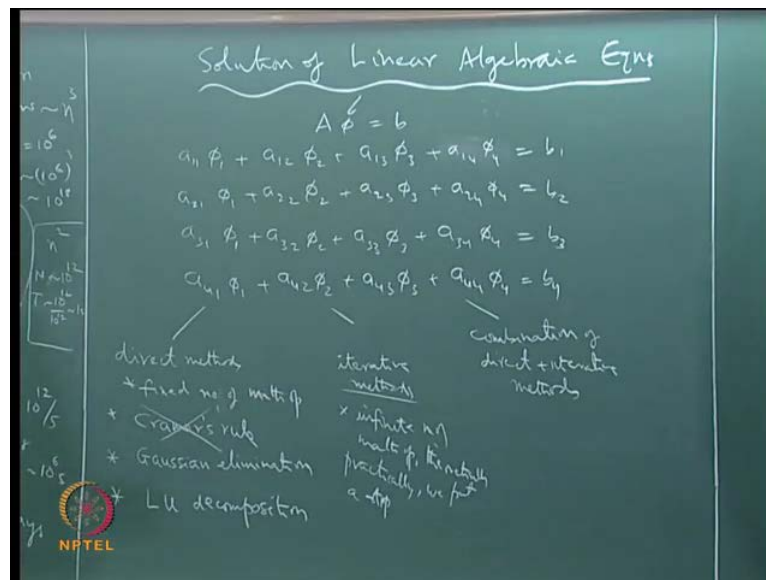
NPTEL

So, we can write it in the general term as  $a\phi = b$ , where  $\phi$  is the set of unknowns and  $a$  can be for example,  $a_{11}\phi_1 + a_{12}\phi_2 + a_{13}\phi_3 = b_1$ . We will put one more  $a_{14}\phi_4 = b_1$ .

So, we are looking at a system of four equations and four unknowns  $a_{21}\phi_1 + a_{22}\phi_2 + a_{23}\phi_3 + a_{24}\phi_4 = b_2$  and  $a_{31}\phi_1 + a_{32}\phi_2 + a_{33}\phi_3 + a_{34}\phi_4 = b_3$  and  $a_{41}\phi_1 + a_{42}\phi_2 + a_{43}\phi_3 + a_{44}\phi_4 = b_4$ .

If you have a million equations, you will have a million of these equations like this with million coefficients coming here. So, that is as big as set of equations that we are trying to solve. We are looking at a set of equations like this in which all the  $b$ 's and all the  $a_{ij}$ 's are known and what are unknown are all the  $\phi$ 's. So, this is the unknown, this is the variable, this contains the elements the variable elements, which you want to solve for known values of  $a_{ij}$  and  $b_k$ . The solution of this type of thing can be done in many ways, so there are one can say that there are three broad classifications.

(Refer Slide Time :41:31)



One is that, we can solve this using direct method and we can solve this using iterative method. As usual, when you have two classifications, there is one more classification which is combination of direct and iterative methods. So, we can have a large family of these methods and among the direct methods, when we say what we mean by direct method, is a method or an algorithm, whereby we do manipulation of these equations as per the given algorithm. After a finite sequence of those finite numbers and a sequence of these manipulations, we finally, get the values of  $\phi_i$  and that means, that a direct method has a sequence of **operations** mathematical operations that are needed to be carried out **in this** on these equations at the end of which we get a solution.

So, the number of mathematical operations by which we mean addition, subtraction, multiplication and division and when we look at the computational time, typically we take only the multiplication and division because these are more time consuming than this addition and subtraction. So, the number of mathematical operations that are to be done on this system of equations in order to get a solution is fixed in a direct method. So, this means that this has a fixed number of **of** mathematical operation and at the end, typically at the end of those fixed number of operation, we get these solutions. So, if you have a million numbers of operations to be done typically towards the end or at the end we get all the values and typically towards the end we get **ah** most of the values.

Iterative methods, they have a totally different principle. You start with a guest value and you try to improve upon it in such a way that these equations are somehow solved in their approximate solution.

So, you go from a guess value to an improved value and you put back that improved value in your algorithm to find an approximate value. Then you put that improved value back in to that black box, which describes that which comprises those kinds of operations that are needed to get the improved value. So, you can you put back the improved value in to the black box and then you get a further improved value and then you put it back like this. You do it many many times, until and until you get a more and more and more improved value. So, these are characterized typically by an ever improving value ok.

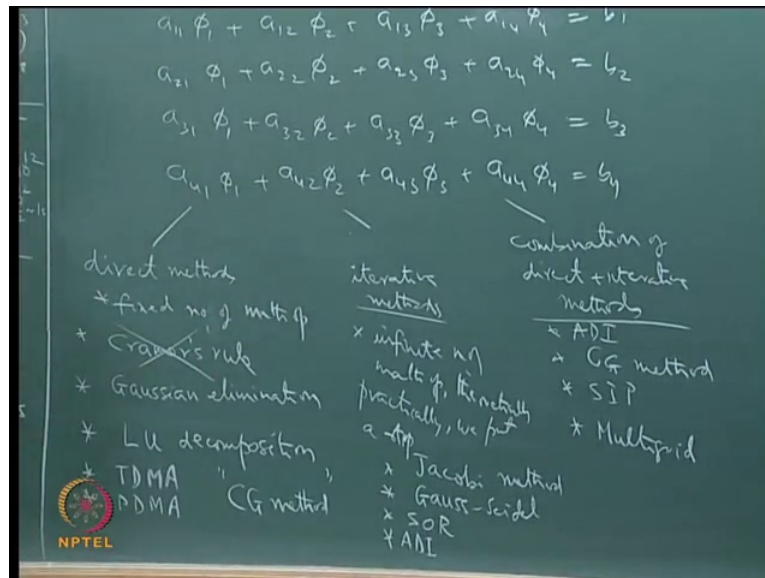
When we say ever improving value, we mean to say what is the most correct value? What is the correct value? Will it ever reach the correct value? Typically these Iterative methods asymptotically reach the correct value and any asymptotic thing is infinite number of steps ok. We do have a limitation in terms of the machine accuracy of computation, when we do it numerically using a computer, but typically we have a very large number of calculations that are required to get to the asymptotically exact values.

By definition, you do not reach the exact value when we are looking at the infinite accuracy of the computation, so we have to stop at a particular point; we say that a fourth decimal accuracy is good enough or a tenth decimal accuracy is good enough. So, we use that kind of judgment or another objective way of judgment expressed typically in terms of norms of the residual. So, we make use of some criterion like that to say that this is where we have got an accuracy which is good enough for us. So, the end is specified in terms of some estimate of the goodness of the accuracy that is required, otherwise there is no end.

So, we have an infinite theoretically, infinite number of math operations theoretically, but practically we put a stop to the equation after some number and where we put a stop is not necessarily easy to estimate. So, sometimes you may have to go through, that is by the time you start the computation. So, at a priori you do not know how long it will take to get to that satisfactory condition. So, that is a typical characteristic of Iterative methods, although one can do some estimates of that. So, in theory they have infinite number of mathematical operations, but practically we can if we put accuracy on the level of solution, then we can get some estimate of the number of mathematical operations required for this. So, there is in a

way no end to this, but we will end at some point. So, these are two very contrasting approaches to the solution of a phi equal to b. Let us identify, let us put some name to this, so that we can recognize these things.

(Refer Slide Time :41:31)



The Cramer's rule is one method one direct method, which we straight away discard in any cfd computation because of the number of mathematical operations that are required in this. Gaussian elimination is one very good way of solving system of equations and this is a direct method, you have a finite number of operations of which you get a solution. We have another method, LU decomposition and this is another useful method. We have also the Gauss-Jordan method and there are some more methods, which are more specific like the Tri-diagonal method for a specific type of a form of a coefficient matrix. Then you have the Cyclic method, which is again for a specific type of specific structure a matrix, of a coefficient matrix and so on. So, we can say there are also other methods which are which can be called as Direct methods in theory, but because of finite accuracy of computation numerically they become some Iterative methods and we are look at, we will mention those things later on, but we can identify these. Also, in the context of cfd, we can call this as Tri-diagonal matrix algorithms and Penta-diagonal matrix algorithms.

So, these are something in a way, these are variants of the Gaussian elimination for special structure and even LU decomposition can be considered as a variant of the Gauss elimination

method. So, these are some of the things that we may be using and among the Iterative methods, we have the Jacobi method.

We have already seen the Gauss-Siedel method and we have the Successive Over Relaxation method. We also have methods like ADI method (Alternating Direction Implicit) type of method and we also have other methods, which we can leave at this stage for example, the Conjugate gradient method. It is a fairly complicated method which is in principle a direct method. If we have infinite accuracy, but which is usually put as **ah** used in the context of an Iterative method. When we have finite method of calculation, finite accuracy of calculation and so, this may come as Iterative method and typically this comes as a part of the combination of direct and Iterative method.

So, we have identified already the Conjugate gradient method and of course, we have we can say that even ADI method is a Conjugate gradient. We can say that there are variants of Successive Over Relaxation which will come in this.

So, Conjugate gradient method and then we have some combination method like Strongly Implicit procedure and its variants where we try to take in the combination methods. We try to take advantage of specific features of the Direct methods and specific features of the Iterative method come up with a combination algorithm, which is better than both of these and then we can even say that we have already mentioned Conjugate gradient here. A Multigrid method, although it is **ah** it has a different philosophy to the overall solution is a method which is worth mentioning and it may be called as an Iterative method, but **it has** it is very difficult to put this into any of these classifications.

It is a method definitely it involves some element of Iteration and it is an extreme end and it may use some direct method also. So, we will just mention these things here and at this point, we would not go too much into the details that we will postpone for the next lecture, but we would like to identify these methods in terms of the number of mathematical operations that are required and especially what we are looking at is the exponent value.

We have seen from this example, a method which goes as  $n^3$  may take something like 10 days for the solution of a million unknowns and the same thing which takes as  $n^2$ , where the exponent goes from 3 to 2 will take only about a second. So, we have that much of variation just depending on what the exponent value is for large matrixes, for a large number

of grid points. We want to have a method which is good for which is good also for a large number of grid points. So, the exponent here becomes very crucial.

So, how do these exponents vary? We cannot fix a value of exponent for Iterative method because in theory they go up to infinity, but when we put a practical stop say at a 5 digit accuracy or 10 digit accuracy later, then we put a stop. Based on that, we have some estimates for exponent for this.

Gaussian elimination, it varies as  $n^3$  by three number of operation. So, this is a method which will take a large number a large amount of time, may be 3 days for a million grid points. It is most straight forward form and although, it takes such a long time for a large number of grid points; it is considered as the best direct method for a solution of general  $Ax = b$  equation. So, that is when the matrix here,  $A$  here is mostly filled that is it has very few zeroes and if when it has no specific structure, then this is considered as the best method in a general case; best direct method in a general case.

LU decomposition is in a way it is very similar to the Gaussian elimination method and this and it also takes  $n^3$  by 3 plus a small bit plus some  $n^2$  number of terms. So, for large values of  $n$  as this requires slightly more number of mathematical operations than this and the two are very closely related. Both these methods, Gaussian elimination method and LU decomposition method have a merit that they can be used under the most general conditions. There are no conditions on what this ' $A$ ' should be, except that it must be that the determinant is not 0. So, in that sense these these two methods can be applied for any matrix  $A$ .

(Refer Slide Time :57:27)

$$A \phi = b$$
$$\begin{aligned} a_{11} \phi_1 + a_{12} \phi_2 + a_{13} \phi_3 + a_{14} \phi_4 &= b_1 \\ a_{21} \phi_1 + a_{22} \phi_2 + a_{23} \phi_3 + a_{24} \phi_4 &= b_2 \\ a_{31} \phi_1 + a_{32} \phi_2 + a_{33} \phi_3 + a_{34} \phi_4 &= b_3 \\ a_{41} \phi_1 + a_{42} \phi_2 + a_{43} \phi_3 + a_{44} \phi_4 &= b_4 \end{aligned}$$

direct methods  
no. of multi p

iterative methods  
x inf

combination of direct + iterative methods  
ADT

The TDMA and PDMA are methods that can be used only when there is a diagonal structure. The Tri-diagonal matrix algorithm is applicable only when you have 3 adjacent diagonals. So, this one and then this one and this one are non-zero. So, that is typically the kind of matrix structure that we have for one-dimensional diffusion equation and PDMA is something that can be used when you have 5 adjacent diagonals which are non-zero **ok**.

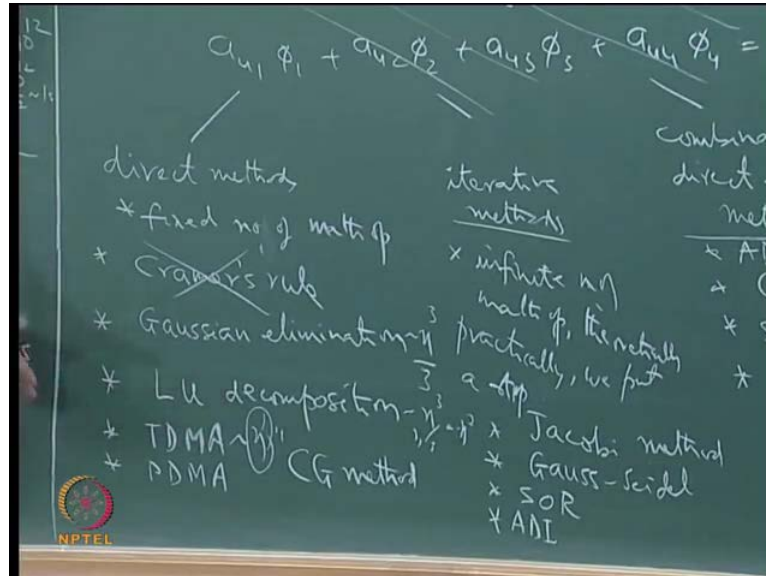
So, this kind of thing will be obtained for a one-dimensional diffusion equation with fourth order accuracy. With second order accuracy, we have 3 adjacent diagonals with fourth order accuracy, we have 5 adjacent diagonals.

When you go to two-dimensions, then you do not have either Tri-diagonal matrix or a Penta-diagonal matrix algorithm. So, only for restricted number of restricted case, you have this TDMA and PDMA thing and TDMA varies as  $n \times n$  to the power 1 with **ah** a constant with a coefficient here which is there, but still compared to the  $n$  cubed they vary linearly with a great advantage; it is a humungous advantage.

We can see that when you go to from  $n$  cube to  $n$  square for a large matrix, we have gone from days to seconds and if it is one, then it is second to microsecond type of thing, but one would be foolish to try to do a one-dimensional equation with such large number of grid points. It usually does not arise. So, the applicability of this as a direct method for an overall solution is in a way questionable, but if you are solving for a one-dimensional flow case with

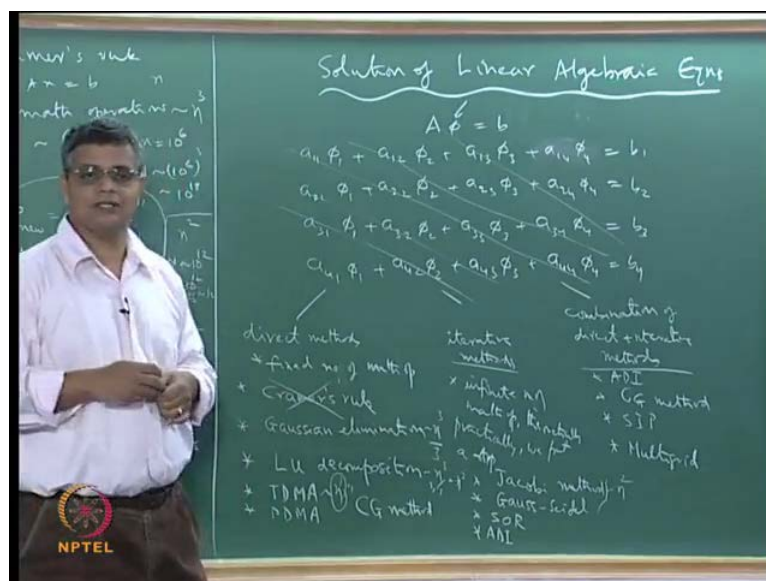
a diffusion term which gives you this type of matrix structure, then probably that is the best method to do.

(Refer Slide Time :59:37)



PDMA will also be something similar and this is even more rare because we are looking at a fourth order accuracy for a first one-dimension equation. So, that is even more rare for a one-dimension differential equation, typically we will look for a second order accuracy. So, this is for direct method.

(Refer Slide Time :1:00:04)



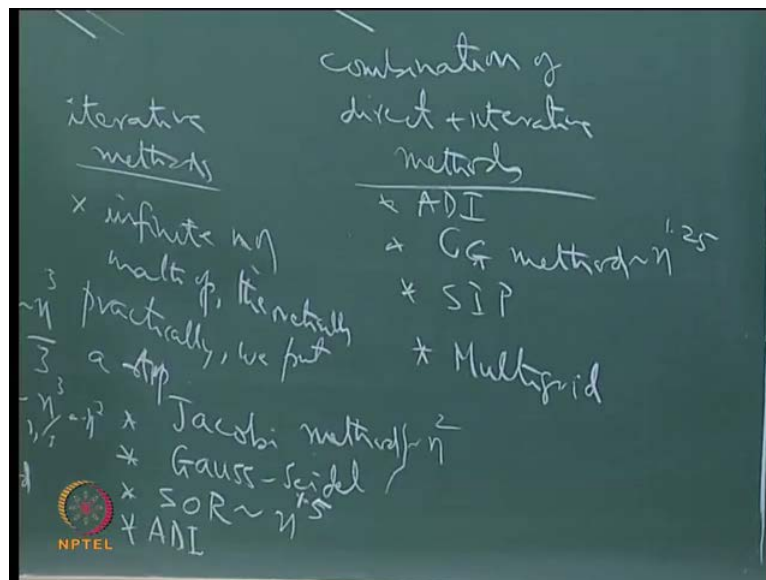


So, the best general purpose direct method has an  $n^3$  number of mathematical operations. Iterative methods, Jacobi method and Gauss-Seidel method typically, have  $n^2$  number of mathematical operations for reasonable number of solution. So, they constitute great improvement from days to seconds for the solution of the same equation, but there is a severe restriction on what kind of equation, what kind of matrix we can, for this we can apply.

These methods, the direct methods can be applied for all matrixes where there is a unique solution, whereas the iterative methods here can be used only when you have for example, the Scarborough criterion which I have mentioned here, the diagonal dominance condition is satisfied, only then we can say that, we can say with confidence that this is these methods will converge.

So, these methods will work in a context of cfd for a convection diffusion equation of the generic scalar type of equation on orthogonal coordinate system. On a non-orthogonal coordinate system, then you have the resulting discretized equations will not be satisfying the diagonal **dominant** dominance condition and in such a case these methods cannot be obtained **ok**.

(Refer Slide Time :1:01:37)



So, SOR Successive Overall Relaxation is an improvement on this and an optimal value in Successive Over Relaxation, there are Overall Relaxation Parameters just as we have Under Relaxation factors, we will make use of over relaxation factors and this will vary under optimal conditions as  $n^{1.5}$ . So, we are going down from 2 to 1 and a half. So, that again

constitutes a big improvement. From second, we can probably go down to millisecond or hundredth of a second for large kind of matrixes, but it is not easy to find the optimal set of parameters the optimal set of over relaxation factors, we have to do more work and there is something like that. You can also have other methods which are, which fall in between this where, we have since we cannot get ah optimal Over Relaxation Parameters, there are methods by which we can find approximate values for them and then get a get a mathematical convergence, which probably falls somewhere between  $n^2$  and  $n^{1.5}$ .

These methods have restricted applicability. The Conjugate gradient method is much more general, it can be one would say that is has been developed for symmetric matrixes, but there are variants of that method, which ah can be used even for asymmetric matrixes.

It is not necessary to have a diagonal dominance condition here; ADI is a combination of these two. It gives faster rate of convergence than say a simple Gauss-Seidel kind of thing, but probably not as fast as the SOR method. Conjugate gradient would be very good and in its simplest form, I think it can go up to 1.25. So, we are getting down closer to this, but that is very good, especially the rate of convergence is very good under optimal conditions.

SIP is probably more like ADI, this will be somewhere between  $n^2$  and  $n^{1.5}$ . Although, it is very difficult to pinpoint what exactly it is, but it is better than the Gauss-Seidel method kind of thing. Multigrid is a new philosophy, it can almost linear kind of ah computational efficiency is obtained with the Multigrid method ok.

So, this is a range a family of methods in which we can go from a general purpose method with  $n^3$  number which takes 10 days to  $n^2$ , which is a general purpose Iterative method with restrictions on what kind of 'a' that we can use for. We can go to seconds and, then we can go to even milliseconds using Conjugative Gradient Method, which are again with some restrictions and so on. With Multigrid method, where we have to do much more work and may be because all the work may be 1.1 or something like that. So, we have family of methods which can be used and wherever it is possible, we have to take advantage of the problem that we have and then come up with the most optimal way of solving this a  $\phi$  equal to b because we have to solve them so many times. We have to solve them here, here, here and then here and then back here so many times. So, we will look at some of the methods what how these methods work in the next couple of lectures.