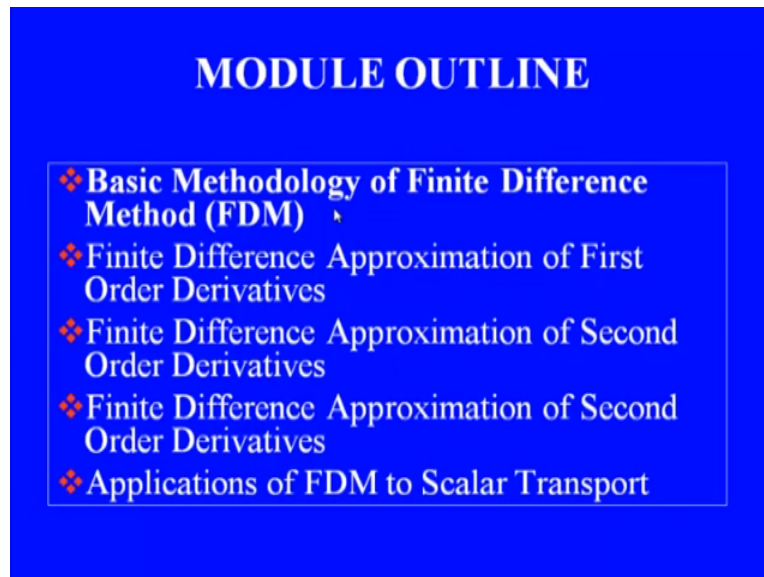


Computational Fluid Dynamics
Dr. Krishna M. Singh
Department of Mechanical and Industrial Engineering
Indian Institute of Technology - Roorkee

Lecture - 10
Finite Difference Method: Methodology and Grid Notation

Welcome to the module 03 on Finite Difference Method. In the previous module we had discussed a derivation of governing equation for fluid flow, which is the first step in the numerical simulation using computational fluid dynamics techniques. And this is the first module on numerical methods which is based on finite differences, so we would focus primarily on finite difference methods and its applications to different order of derivatives.

(Refer Slide Time: 01:00)



So module outline is we will first start off today with basic methodology of finite difference method, and then we will have a look at how do we derive finite difference approximation first order derivative, next finite difference approximation of second order derivatives, and applications of finite difference method to scalar transport problems. In today's lecture we would focus on finite difference method its basic methodology and the notations which we would use in subsequent lectures.

(Refer Slide Time: 01:44)

FINITE DIFFERENCE METHOD (FDM)

- ❖ Oldest method for numerical solution of PDE'S (Euler, 18th century)
 - ❖ Forward Euler and Backward Euler methods for IVPs
- ❖ Easiest method to use for simple geometries.
- ❖ Starting point is the conservation equation in differential form.

So now let us see what is finite difference method a brief look up its history, so this oldest method for numerical simulation of partial differential equations in fact it takes back to 18 century and created to Leonhard Euler, and the two methods which is still referred to by his name Forward Euler and backward Euler method for initial value problems, both of these methods are based on simple one step difference approximation of time derivative.

And we will have a detailed look at both of these methods when we come to time discretization aspects, now this is one of easiest methods to use for simple geometry in fact this finite difference method started off for solving partial differential equations on rectangular geometries and that it was cited as one of the basic limitations of finite difference method. But today we will just say that there is no more the case then it can be utilized for solving problems on any type of geometry.

And starting point is the conservation equation in differential form this is what we derived in the previous module, and we would use the techniques which we learn today for difference approximation today and the next few lectures the difference approximations of the derivatives in our governing equations and that would be used in obtaining a discrete form for numerical simulation.

(Refer Slide Time: 03:14)

ATTRACTIVE FEATURES OF FDM

- ❖ It is the easiest method to use for simple geometries, both in terms of formulation and programming.
- ❖ It can also be adapted for problems in complex geometries using boundary fitted grids or the concept of immersed boundaries (IBM).
- ❖ Domain decomposition based solvers can be easily adapted for solution of algebraic systems obtained from FDM, and thus, this method is uniquely suited for massively parallel architectures

For the attractive features of our difference method these are the other schemes, it is easiest method to use for simple rectangular geometries, it is very easy to formulate, very easy to derive difference equations starting from the partial differential equation for any problem, and writing a computer program in any language translating our difference formulation is straight forward.

In fact so much so that today we routinely give it to our undergraduate students to write computer programs for solving heat conduction problems and fluid flow problems based on finite difference method, but one of the limitations we decided earlier was that this finite difference method is primarily suited to simple rectangular geometries because the method is based on the use of local Cartesian grids but that is no more the case.

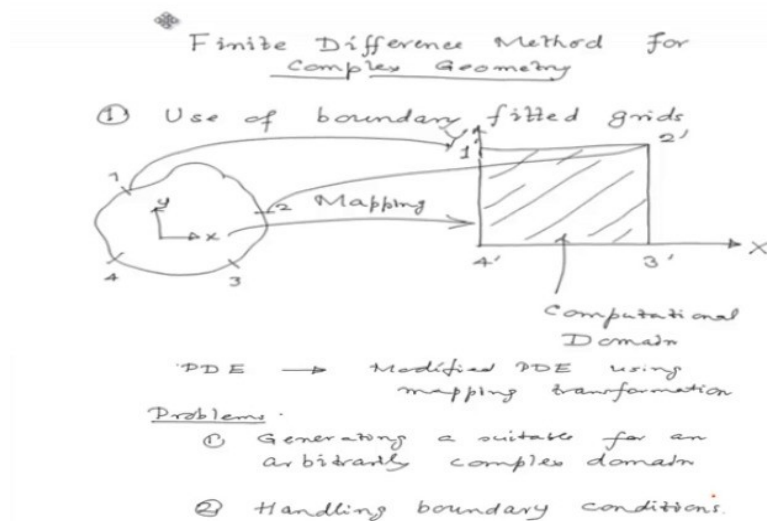
In the beginning to overcome this limitation people used for complex geometries what we call boundary fitted grids, but today the recent developments based on what we called the concept of immersed boundaries that had left behind the mapping which is required in boundary fitted grids and this limitations all together. And we can use simple Cartesian grids for modeling any complex applications, this point we would elaborate little later.

Other advantage which we have got is we can use domain decomposition based solvers, if you are dealing with very large scale a numerical simulation which we have to routinely do for a solution of industrial problems specifically the problems which involved turbulent flow. Now the

whole problem cannot be solved on a workstation or a single computer and we have got to use massively parallel computers for numerical simulation.

And for this parallel simulation we normally use what we call domain decomposition based approach, and finite difference method is one of the easiest to adapt to this domain decomposition approach.

(Refer Slide Time: 05:23)



Now let us have a brief look at the application of finite difference method to problems in complex geometry, so I mentioned 2 approaches. The first one is what we call use of boundary fitted grids, what do we do in this approach is suppose we have got an arbitrary domain and we have somehow got to map it to a rectangular domain so let us do these points 1, 2, 3 and 4. So each one of these wanted to that becomes one side, 2 to 3 is another side, 3 to 4 is another side and 4 to 1 is another side which has to be mapped okay.

We have to obtain a mapping, so that we can map it to a simple rectangular or a square domain, let us call the co-ordinate system here by capital X, Y and the co-ordinates system which we had for usual physical space is small x, y. So each point is mapped, let us say one goes to 1 prime, 2 goes to 2 prime and so on 3 prime and 4 prime, and this square domain which we have got square rectangular domain this is what becomes our computational domain on which our finite difference equation has to be solved.

So what do we do in this case our finite our partial differential equation which we have got that has to be modified, so we have got our modified PDE using mapping transformations. There are 2 problems associated with this approach, the problems which we had is generating a suitable mapping for an arbitrarily complex domain, so this is not a straight forward task it complicates the things a numerical solution quite a bit.

And the second problem was handling of boundary conditions, so this is not as complicated as the first one, but nevertheless the transformation from physical domain to computational domain it makes the boundary conditions a bit complex okay, so these two difficulties did put certain constraints in the application of finite difference method to the problems in complex geometries, but nevertheless this approach was used very often in 1980s and 1990s.

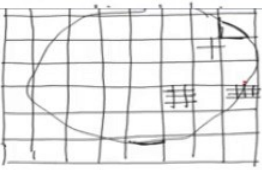
(Refer Slide Time: 09:34)

FDM for complex geometry

② Immersed boundary method (IBM)

⇒ Simple Cartesian grids (No mapping required)

- Curved boundaries are taken care of by introducing forcing fns. (similar to body force terms) in our governing equation.



- Peskin (1972) for bio-fluid problems

⇒ Most active research areas in CFD

⇒ Fluid-structure interaction problem can be easily as solving flow problems.

⇒ IBM can be used with adaptively refined Cartesian grids.

There is approach FDM for complex geometry, our second approach is based on what we call immersed boundary method or in sort we use the acronym IBM, so the name itself implies what we have to do is we have to immerse our arbitrarily complex domain into a simple rectangular domain. So we would use this is based on simple Cartesian grids, so that means we do not have to use any mapping no mapping required.

So suppose we had our arbitrarily complex domain all that we do is we just immerse it into a rectangular grid, so this is a simple rectangular grid, then the question arises how do we take care of these complex boundary segments which do not align themselves with these Cartesian grids. So here curved boundaries are taken care of by introducing forcing functions, now these forcing functions are very similar to boundary forces sorry similar to the body force terms and this is all we have to do.

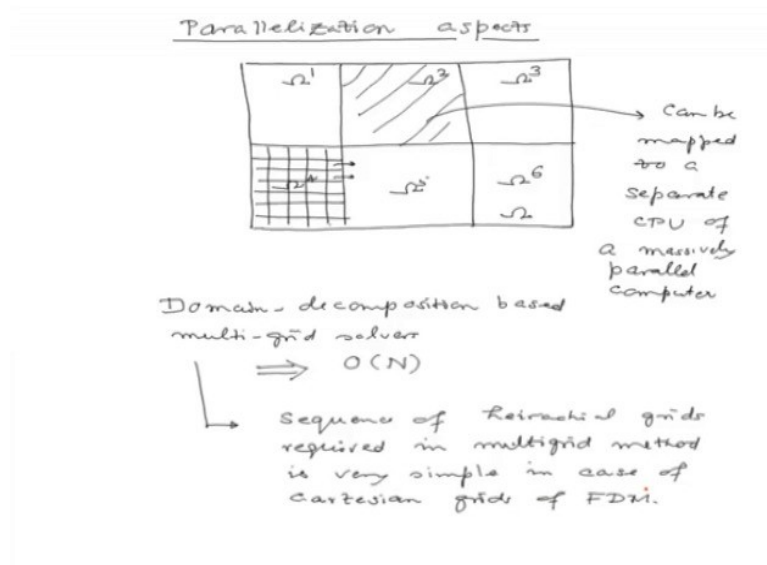
So introduce this term in our governing equation and that governing equation would be that of momentum or energy equation, so this method was proposed long long ago by a Peskin in 1972 for what we call today is bio-fluid problems, so naturally when we are dealing with let us say the blood flow in our arteries of heart we are dealing with fairly complicated geometries.

And we can easily now what Peskin did was immerse that complex domain into a simple rectangular Cartesian grid take care of the complex boundary parts by introducing forcing terms in momentum equation and solve the problem. Now this is one of the most fertile areas of research today, so most active research today, so most active research areas in CFD and the beauty of this method is that using IBM we can solve not just the fluid problem.

We can also solve fluid structure interaction problems, this can be handled as easily as solving a simple flow problem, and recently many new developments taking place in the case of IBM, and we can use this IBM together with what we call can be used with adaptively refined Cartesian grids, what do you mean by adaptive refinement let us suppose in a specific region of the problem domain, we would like to have very fine mesh based on some error criteria.

So that we can introduce, so we can have grids of different densities in different domains to take care of the boundaries maybe we can have a fairly fine grid close to the boundary, and these features are permitted in the context of immersed boundary method. Now let us have a brief look at the advantages which FDM offers in parallelization.

(Refer Slide Time: 16:12)



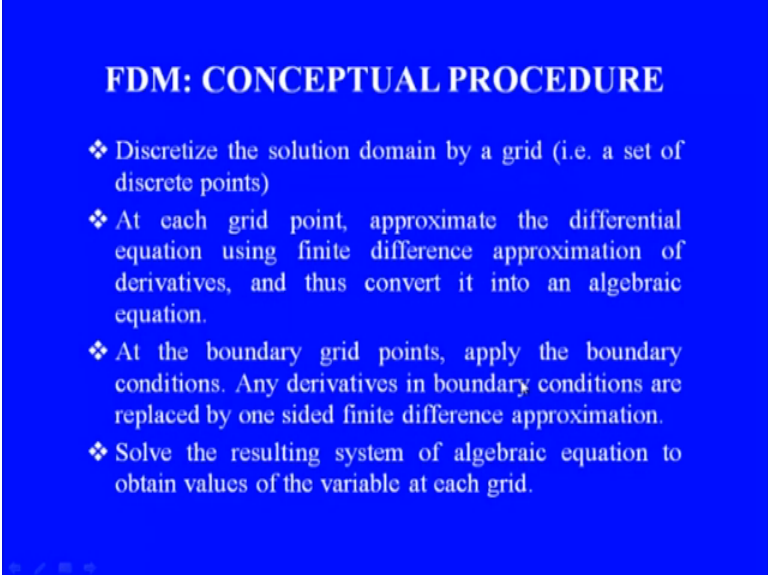
So here we can use simple rectangular decomposition of a problem domain, take for instance a simple rectangular geometry decomposition for a complex geometric would be fairly similar, so this is a big problem domain Ω , now this domain can be decomposed in smaller subdomains let us call the Ω_1 , Ω_2 , Ω_3 , Ω_4 , Ω_5 and Ω_6 . And each one of these subdomains they can be mapped to a separate processor or separate CPU of a massively parallel computer.

So we do not have to make use of any advanced or fairly complicated mesh partitioning algorithm in this case, we can simply decompose our problem domains a small small subdomains, we can have a different types of Cartesian grids or grids of different density on each subdomain, and the interface information can be easily passed on from one processor to another as a part of solution process.

Yet another important aspect which is permitted by this finite difference and domain decomposition based approach is, so we can have domain decomposition based multigrid solvers, which are very efficient and in fact they scale with the order n where n is the total number of nodes, and generation of what we call a sequence of grids a sequence of hierarchical grids required in multigrid algorithm is very simple or very straight forward in case of Cartesian grids of finite difference method.

So these important advantages are attractive features of finite difference method today make it one of the most widely used numerical discretization scheme for simulation of very complex turbulent flow and fluid structure problems.

(Refer Slide Time: 19:59)



FDM: CONCEPTUAL PROCEDURE

- ❖ Discretize the solution domain by a grid (i.e. a set of discrete points)
- ❖ At each grid point, approximate the differential equation using finite difference approximation of derivatives, and thus convert it into an algebraic equation.
- ❖ At the boundary grid points, apply the boundary conditions. Any derivatives in boundary conditions are replaced by one sided finite difference approximation.
- ❖ Solve the resulting system of algebraic equation to obtain values of the variable at each grid.

Now before proceed to details of the method let us have a look at what we called conceptual procedure, that how do we solve a given fluid problem using finite difference method. So first thing which we have said besides discretize the solution domain by a grid, that is our continuum problem is represented by a set of discrete points which we will call nodes, we will elaborate on this point little later.

And next, what we would do at each grid point we approximate the differential equation using finite difference approximation of derivatives by use of this finite difference approximation for the derivatives, thus converts a partial differential equation into an algebraic equation at each grid point, we repeat the same exercise at boundary grid points as well, that boundary conditions are fairly simple as of differential type where the solution variable itself is specified we do not have to do much.

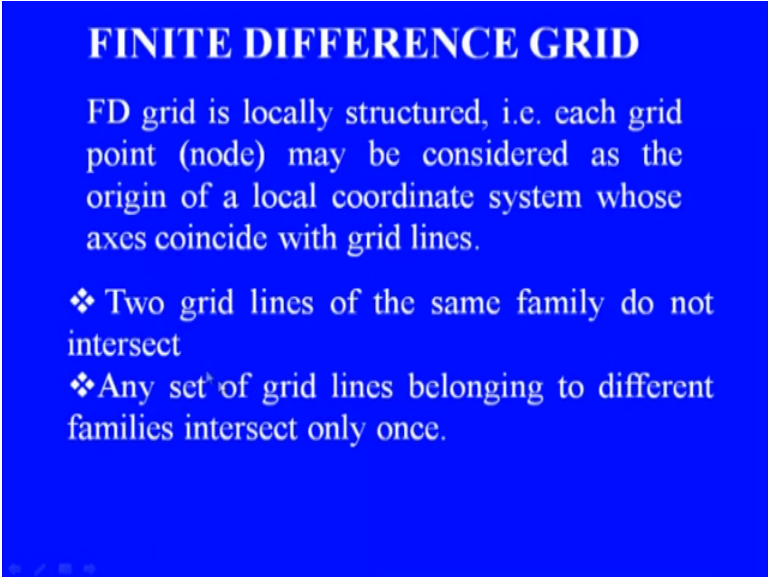
But if there are any derivatives in boundary conditions we replace or we approximate these derivatives by using one sided finite difference approximation formula, so now with these 2 steps we will now converted our partial differential equation or boundary conditions into discrete

algebraic equation at each node interior as well as boundary nodes. Next, what do we do collect the algebraic equations obtain at the integral node as well as boundary nodes to form a system of algebraic equation which will usually be a sparse algebraic equations.

And we just need to solve this resulting algebraic equation to find out the values of problem variable at each grid point, if you are dealing with time dependent problem then instead of an algebraic equation you would get what we call a discrete system of ordinary differential equations, and that has to be further discretized using a time discretization scheme which we will discuss in future module.

And once we have obtained the solution values of each of the grid points, obtaining solution at any point in the interior domain is very simple, we can obtain the derivatives of variable let us say you are dealing with flow field, we have got the velocity values and pressure, we can obtain the stresses at any point in a solution domain by numerical differentiation and interpolation process, so this is our simple conceptual procedure.

(Refer Slide Time: 22:53)



FINITE DIFFERENCE GRID

FD grid is locally structured, i.e. each grid point (node) may be considered as the origin of a local coordinate system whose axes coincide with grid lines.

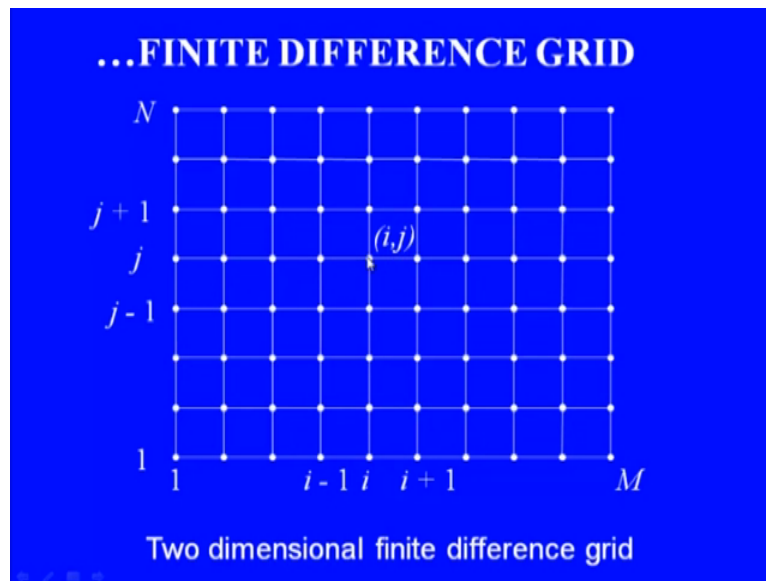
- ❖ Two grid lines of the same family do not intersect
- ❖ Any set of grid lines belonging to different families intersect only once.

Now we have mentioned grid, now let us have a look at few more things notation the grid which we see, what are the properties of a grid? The finite difference grid is locally structured, what do you mean by this term structured? It simply means that each grid point which would also called node may be considered as the origin of a local co-ordinate system whose axes coincide with our

grid lines, grid lines represent the lines or curves or maybe surfaces in the space at which a certain co-ordinate value remains constant.

Suppose, you have got grid lines of same family the properties are two grid lines of the same family do not intersect, and the next property of the grid is that any set of grid lines belonging to different families intersect only once.

(Refer Slide Time: 23:43)



So let us have a look at a typical two dimensional finite difference grid on a rectangular domain, so we have used M grid lines in x direction and N grid lines in j direction, each node can be identified by a graph indices corresponding to the intersection of the grid lines for since in this particular node, it belongs at the intersection of $x=x_i$ grid line and $y=y_j$ grid line, so we will use a simple symbol i,j to denote this finite difference node okay.

(Refer Slide Time: 24:25)

GRID NOTATION

Each node is identified by a set of indices which are the indices of the grid lines that intersect at it. Thus,

- ❖ Node i represents the grid point $x = x_i$ in one dimension.
- ❖ Node (i,j) represents the intersection point of grid lines $x = x_i$ and $y = y_j$ in 2-D.
- ❖ Node (i,j,k) represents the intersection point of grid lines $x = x_i$, $y = y_j$ and $z = z_k$ in 3D.

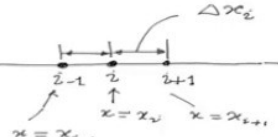
So what we do each node is identified by a set of indices which are the indices of grid lines that intersect at it. Thus, node i it represents the grid point $x=x_i$ in one dimensions, node i, j represents the intersection of intersection point of grid lines $x=x_i$ and $y=y_j$ in 2-D that is what we saw in the previous slide. Similarly, node i, j, k represents the intersection points of grid lines $x=x_i$, $y=y_j$ and $z=z_k$ in three dimensions. Now there are some shorthand notations which we use in our grids.

(Refer Slide Time: 25:24)

Shorthand Notations

Grid spacing

$$x_{i+1} - x_i \equiv \Delta x_i$$

$$x_i - x_{i-1} \equiv \Delta x_{i-1}$$


Multi-dimensional space:

$$y_{j+1} - y_j \equiv \Delta y_j$$

$$z_{k+1} - z_k \equiv \Delta z_k$$

Function values

① 1-D $f(x_i) \equiv f_i$

② 2-D $f(x_i, y_j) \equiv \underline{f_{i,j}} \equiv f_{ij}$

$$f(x_{i+1}, y_{j+1}) \equiv \underline{f_{i+1,j+1}}$$

③ 3-D : Use three subscripts

$$f(x_i, y_j, z_k) \equiv \underline{f_{i,j,k}} \equiv f_{ijk}$$

$$f(x_{i+1}, y_{j+1}, z_{k+1}) \equiv \underline{f_{i+1,j+1,k+1}}$$

Let us have a look at these two shorthand notations which will get spacing and function values at grid points. So shorthand notation the first want to denote what we call grid spacing, so let us have a look at simple one dimensional grid space or one dimensional line here, these are our points let us say a generic grid point we denote by index i which corresponds to $x=x_i$, the next

grid point we use a symbol $i+1$ this corresponds to $x=x_{i+1}$, and similarly, $i-1$ which corresponds to $x=x_{i-1}$.

Now how do we represent these spacing's, we use a symbol capital delta to denote the difference between these, so this x of $i+1-x_i$, this denotes the spacing between grid points x_{i+1} and x_i , and we adopt a simple convention we called as a delta x_i , so $x_{i+1}-x_i$ we would use a shorthand notation Δx_i , now following this notation the difference between i th grid point and one to the left of it that is $x=x_{i-1}$ that is our $i-1$ of grid point x_i-x of $i-1$ we will call or we will denoted by Δx_{i-1} .

Similarly, the same convention we can extend in multi-dimensional space, so in multi-dimensional space in other two directions we will use the symbol j to denote our y lines, so $y_{j+1}-y_j$ this grid spacing would be denoted by Δy_j . And let us use subscript k to represent index in k z directions, so $z_{k+1}-z_k$ we would use the shorthand notation Δz_k okay. The next shorthand notation function values.

The first in the case of one dimension at point x_i for a generic function f , $f(x_i)$ we would use a simple shorthand symbol f subscript i , so in our finite difference formula that is what we would use. Similarly, in two dimensional space in 2-D to represent the function value at point x_i, y_j we would use 2 subscript to symbol $f_{i,j}$, there is no confusion you would simply write it as f of ij , this comma mandatory in case of we dealing with slightly more complex indices for i and j for instance if you want to denote x of $i+1$, y of $j+1$ the function value here.

So now in this case we must use separation by comma between the subscripts, so $f_{i+1, j+1}$, so this shorthand notations make it very simple for us to write our difference equation in a compact form, and they also help us to transform or translate these formulae into a computer programming languages. Now let us see what do we do in 3-D, 3-D we will use 3 subscripts that is f at x_i, y_j, z_k this be would represent by $f_{i,j,k}$.

Or alternatively, we can also write it as omit the comma in this case f_{ijk} as I said earlier this comma you would require when this indices are longer than 1 symbol, for instances $f_{x_{i+1}, y_{j+1}, z_{k+1}}$, so now in this case we have to use comma separator between different indices, so

(Refer Slide Time: 31:56)

- ❖ Uniform Grid
- ❖ Non-uniform grid

(Refer Slide Time: 32:23)

1-D A grid is uniform if spacing between each pair of grid points is the same.

$$\Rightarrow \Delta x_v = \Delta x_{b-1} = \Delta x_{b+1} = \dots = \frac{\Delta x}{\dots}$$

Grid will be called uniform if spacing along each grid line is constant.

constant.

$$\begin{aligned}\Delta x_i &= \Delta x_{i+1} = \Delta x_{i+2} = \dots = \Delta x \\ \Delta y_j &= \Delta y_{j+1} = \Delta y_{j+2} = \dots = \Delta y \\ \Delta z_k &= \Delta z_{k+1} = \Delta z_{k+2} = \dots = \Delta z\end{aligned}$$

Uniform grid

If you are dealing with an one dimensional problems 1-D a grid is uniform if spacing between each pair of grid points is the same, so if you got a set of grid points let us say $i, i+1, i+2, i-1$ and

so on, so we can simply say that $x_{i+1} - x_i = x_i - x_{i-1} = x_{i-1} - x_{i-2} = \dots$ and so on each of these differences are equal or using our complex notation, we can write this as $\Delta x_i = \Delta x_{i-1} = \Delta x_{i+1}$ and so on, and we can use drop this subscript to Δx we can simply write this as Δx .

So now we have got a uniform grid spacing Δx and if this were not the case we will call the grid as non-uniform. In multi-dimensions the grid would be called uniform if spacing along each grid line is constant, now this constant does not mean that we have to have same spacing along each families of grid lines, all that this definition requires is that our Δx along each lines separately that is $\Delta x_i = \Delta x_{i-1} = \Delta x_{i+1}$ and so on this $= \Delta x$.

Similarly, $\Delta y_j = \Delta y_{j+1} = \Delta y_{j-1}$ and so on, and it is equal to a constant spacing Δy along the j grid lines. And similarly, $\Delta z_k = \Delta z_{k+1} = \Delta z_{k-1}$ and so on and that is equal to a constant increment or spacing Δz . So that is what we mean by this uniform grid in three dimensions, we need not be any correlation between Δx , Δy and Δz they can all be different, and as a corollary a grid is non-uniform if it is not uniform okay.

(Refer Slide Time: 37:01)

FINITE DIFFERENCE APPROXIMATION

Definition of the derivative:

$$\left(\frac{\partial f}{\partial x} \right)_{x_i} = \lim_{\Delta x \rightarrow 0} \frac{f(x_i + \Delta x) - f(x_i)}{\Delta x}$$

An approximate value of the derivative can be obtained from the finite difference expression given by

$$\left(\frac{\partial f}{\partial x} \right)_{x_i} \approx \frac{f(x_i + \Delta x) - f(x_i)}{\Delta x} \equiv \frac{f_{i+1} - f_i}{\Delta x}$$

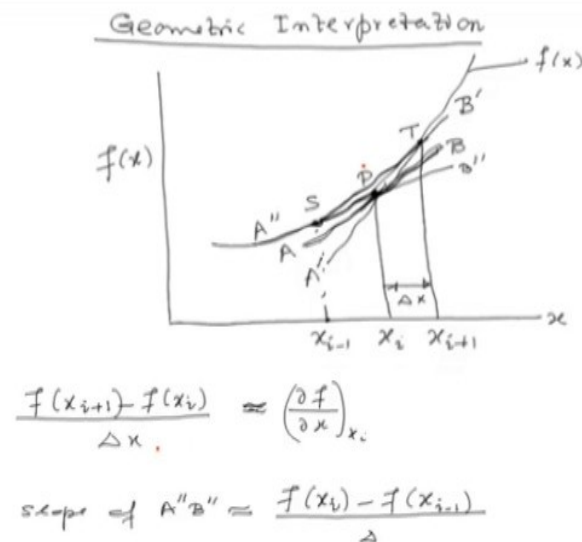
Let us come to the basic concept of finite difference approximation and as you can easily guess it is based on would we are using approximation for what? We are using approximation for a derivative, so let us go back to the basic definition of the derivative which we have learned in our

calculus class. The derivative of function f with respect to x at a given location x_i is $\frac{df}{dx}$ at x_i . How do we define it, we defined it as the limit of Δx tending to 0 $\frac{f(x_i + \Delta x) - f(x_i)}{\Delta x}$.

So that is the basic definition of derivative, now if you want to obtain an approximation to this derivative, what we can do that approximate value of the derivative can be obtained from the finite difference expressions where we do not let Δx become 0, we take finite Δx that is why we use the term finite difference, so $\Delta f / \Delta x$ the derivative of f with respect to x at point x_i , this can be approximated by a simple finite difference the value function f at point $x_i + \Delta x - f(x_i) / \Delta x$.

Or if you use a complex notation for these function values we can write it as $f_{i+1} - f_i / \Delta x$, now this is one approximation which is frequently referred to as forward difference approximation, why we called forward? Because we have used the value as a function at point to the right of point $x = x_i$ that is what this value f_{i+1} means. Now there is a geometry interpretation which we have in this case, let us have a brief look at the geometric interpretation for a function.

(Refer Slide Time: 39:01)



Suppose, we take a generic function this $f(x)$ and this is a curve which represents of function $f(x)$, we were to find out this derivative at point x_i , so derivative is given by the value of this tangent if you draw a tangent to this point, the slope of the tangent that is what gives us the derivative. Now suppose, we take this point x_{i+1} , so the expression which we saw $f(x_{i+1}) - f(x_i) / \Delta x$, this difference is Δf , now this is given by the slope of that is line the.

So the initial tangent let us say at point P was A, B the forward difference approximation is basically the slope of this line A prime B prime, the same thing we can say look at if you take instead of point to the right of point x_i we can choose one towards the left that is called x_{i-1} and if you see the values here these 2 values let us define this point and draw a different line let us say this A double prime B double prime, slope of A double prime B double prime that is $f(x_i) - f(x_{i-1})/\Delta x_{i-1}$ okay.

and this is what we referred to as a backward difference approximation, so that is geometric meaning of this slopes, and it is also very easy if we take the slope of these two points here at $x=x_{i-1}$ let us join this points let us call them as S and T, we can clearly see the slope of that line ST is a pretty close approximation it is all almost parallel to a tangent AB.

And later on we will learn that the so called central difference approximation which we can get by using the values on both the sides of our point P that gives us a much better approximation of the derivative compared to forward difference or backward difference approximation. Before we can conclude this lecture let us have a look at few useful definitions which would frequently use in coming lectures.

(Refer Slide Time: 42:42)

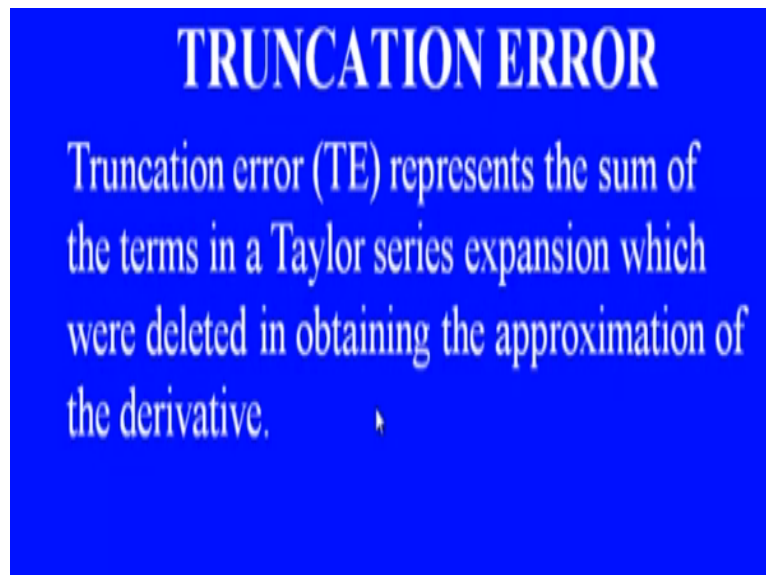
USEFUL DEFINITIONS

- ❖ Order of Magnitude
- ❖ Truncation Error
- ❖ Order of Accuracy

We will look at two definitions order of magnitude, truncation error and order of accuracy of a particular approximation. Now this order of magnitude is what we would use when we specified the truncation error or the order of accuracy, so what do you mean by the order of magnitude? Let us say that we have got quantity g_x is a function g of x , it is said to be of the order h to the power m where $m > 0$.

We ultimately write this $g_x \sim O(h^m)$ this capital O is used to represent order of so $O(h^m)$ if limit $h \rightarrow 0$ $g_x/h^m = L$, where L is a finite quantity. Now this definition is taking from the book of Niyogi et al and it tells us when we would say that the g_x of the order $O(h^m)$.

(Refer Slide Time: 43:41)



Next, the truncation error we just saw that we represent our an approximation of the derivative by using let us say finite differences, how much error we have committed? That we can obtained from Taylor series expansion, so truncation error represents the sum of terms in a trailer Taylor series expansion which were deleted in obtaining the approximation of the derivative. So let us have a look at what we just did and see what do you mean by this Tailors sorry truncation error in the case of Forward or backward differences?

(Refer Slide Time: 44:19)

Truncation Error

Taylor series expansion at $x = x_i$

$$f(x_{i+1}) = f(x_i) + \Delta x \left(\frac{\partial f}{\partial x} \right)_{x_i} + \frac{\Delta x^2}{2} \left(\frac{\partial^2 f}{\partial x^2} \right)_{x_i} + \frac{\Delta x^3}{3!} \left(\frac{\partial^3 f}{\partial x^3} \right)_{x_i} + \dots$$

$$\Rightarrow \left(\frac{\partial f}{\partial x} \right)_{x_i} = \frac{f(x_{i+1}) - f(x_i)}{\Delta x} - \left\{ \frac{\Delta x}{2} \left(\frac{\partial^2 f}{\partial x^2} \right)_{x_i} + \frac{\Delta x^2}{6} \left(\frac{\partial^3 f}{\partial x^3} \right)_{x_i} + \dots \right\}$$

$$\Rightarrow \boxed{\left(\frac{\partial f}{\partial x} \right)_{x_i} \approx \frac{f(x_{i+1}) - f(x_i)}{\Delta x} + (\tau E)}$$

For the sake of simplicity let us concentrate on forward difference on an uniform grid, so f at point x_i using Taylor series expansion we can write it as, so Taylor series expansion at $x = x_i$ that gives us that $f(x_{i+1})$ this can be written as f at $x_i + \Delta x$ times $\frac{\partial f}{\partial x}$ at $x_i + \frac{\Delta x^2}{2} \frac{\partial^2 f}{\partial x^2} + \frac{\Delta x^3}{6} \frac{\partial^3 f}{\partial x^3}$ the third derivative of f with respect to x at point x_i so on.

Now if you rearrange the terms that is what would we get for this $\frac{\partial f}{\partial x}$ at $x_i = \frac{f(x_{i+1}) - f(x_i)}{\Delta x}$ - the higher order terms which we had so we will get this $\left\{ \frac{\Delta x}{2} \frac{\partial^2 f}{\partial x^2} + \frac{\Delta x^2}{6} \frac{\partial^3 f}{\partial x^3} + \dots \right\}$ square at $x_i + \frac{\Delta x^2}{3!} \frac{\partial^3 f}{\partial x^3}$ at x_i so on. So if you want to use an approximation this forward difference approximation you will retain only the first term on the right hand side, and the remaining this terms in bracket we can say that this is what our truncation error okay.

So truncation error basically gives us that higher order terms which we omitted in obtaining a finite difference approximation of the derivative. Next, we would use the term very often let us say that an different scheme is such and such order occur it, what do you mean by that?

(Refer Slide Time: 47:25)

ORDER OF ACCURACY

If the truncation error of a finite difference approximation is $O(\Delta x^m)$, then it is said to have the accuracy of order m (or be m th order accurate).

A simple truncation error if the truncation error of a finite difference approximation is order of Δx to the power m , then this difference approximation is said to be to have the accuracy of order m or you would use alternative description as that the differences scheme is m th order accurate. Now there are various approaches which are used for approximation of derivatives.

(Refer Slide Time: 47:50)

APPROXIMATION OF DERIVATIVES

Most popular approaches are

- Taylor series expression
- Polynomial fitting
- Pade approximants
- Difference equations
-

Our focus: finite difference approximation of first and second order derivatives.

So I am just provided a list here of most popular approaches, the first one is Taylor series expression, the second one is based on polynomial fitting, the third one is based on pade approximants, we can use difference equations and there are many other approaches. We would focus in our lectures next few lectures on these two approaches Taylor series expression or expansion and polynomial fitting.

Now remember that when we derived our conservation laws or governing equations, we had mostly first order and second order derivatives, the first order derivatives we got in convective terms, and the diffusion terms on the right hand side we had second order derivatives. So we would primarily focus on finite difference approximation of first and second order derivatives in our next few lectures. We will stop here and we will continue our discussions on finite difference approximation of first order derivative in the next lecture.