

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

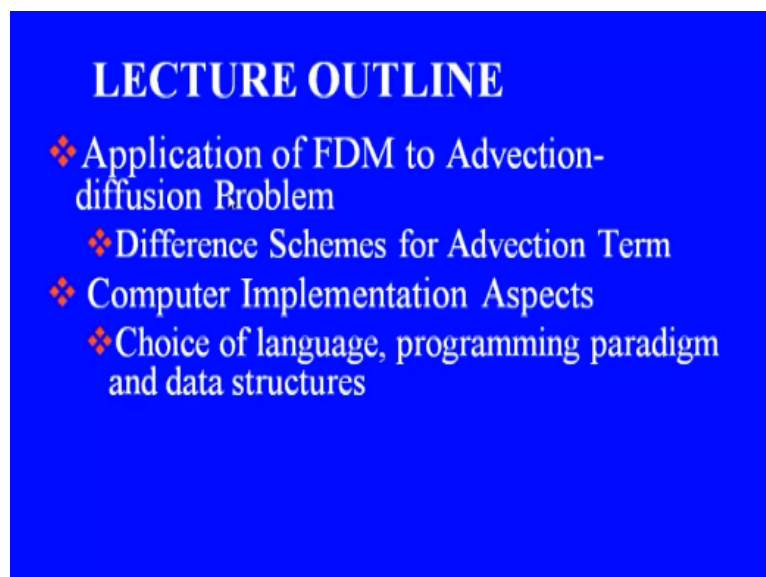
Lecture - 18

Application of FDM to Advection Diffusion and Computer Implementation Aspects

Welcome back to the last league of module 3 on finite difference method. We are working on the applications of finite difference method to scalar transport and its computer applications. Now let us have a recap of previous lecture, the lecture 8 which we had earlier. We completed self-centered grid development for finite difference discretization for 1-D heat conduction and we also discussed how this simple approach can be easily extended for multi dimensional problems.

Then we discussed application of finite difference method to 2 dimensional heat conduction transient heat conduction. And today we are going to focus in today's lecture on advection diffusion problem and computer implementation aspects.

(Refer Slide Time: 01:18)



LECTURE OUTLINE

- ❖ Application of FDM to Advection-diffusion Problem
 - ❖ Difference Schemes for Advection Term
- ❖ Computer Implementation Aspects
 - ❖ Choice of language, programming paradigm and data structures

The brief outline of what we will do today. Application of finite difference method to 1 dimensional advection diffusion problem, extension to multi dimensional should be fairly simple. We will have a brief look at different schemes which are available, different difference scheme for advection terms and we will then focus on the computer implementation of the algorithms, which we learnt so far for real life CFD.

Primarily, we will have a look at the choice of languages which are used, programming paradigm and data structures.

(Refer Slide Time: 01:55)

ONE DIMENSIONAL ADVECTION-DIFFUSION

One dimensional advection-diffusion problem for a scalar variable ϕ is governed by

$$\frac{d}{dx}(\rho u \phi) = \frac{d}{dx} \left[\Gamma \frac{d\phi}{dx} \right]$$

where u is specified velocity, ρ is density, and Γ is diffusivity. Suppose values of ϕ are specified at both ends of the domain of length L , then bc's are

$$\phi(0) = \phi_0 \quad \text{and} \quad \phi(L) = \phi_L$$

Now let us first take 1 dimensional advection diffusion case. So if we are dealing with steady state problem 1 dimensional advection diffusion for a scalar variable ϕ is governed by $\frac{d}{dx}(\rho u \phi)$. This left hand side this term we would call as convection term because this represents the advection or convection of a scalar ϕ due to velocity u . On the right side, we have got $\frac{d}{dx}$ within bracket $\Gamma \frac{d\phi}{dx}$, where we will presume that Γ may be a function of the spatial variables. So this term we refer to as diffusive term.

Now this usual symbols u stands for specified velocity ρ is density and what follows we are going to assume that these might be considered as constant and Γ is diffusive array. Now let us take this typical boundary value problem. Suppose ϕ values are specified at both ends with domain of length L , then boundary conditions are ϕ at $X=0 = \phi_0$ where ϕ_0 is a specified value and similarly the right end ϕ at $X=L$ is ϕ_L .

This ϕ_0 and ϕ_L they would both denote these specified values. Incorporation of gradient boundary conditions should be exactly done in the same way as we have discussed earlier for 1 dimensional conduction problem.

(Refer Slide Time: 03:26)

... One Dimensional Advection-Diffusion

- ❖ CDS scheme for FD approximation of diffusive term
- ❖ CDS/Upwind scheme for advective term
- ❖ Discretized equation for an interior node i can be represented as

$$A_P^i \phi_P + A_W^i \phi_{i-1} + A_E^i \phi_{i+1} = Q_i$$

Coefficients A contain contributions of both convective and diffusive terms, i. e. $A = A^d + A^c$

So let us see your choice of schemes which you can have. We will primarily choose central difference scheme for finite difference approximation of diffusive term that is the second order derivative which we had on the right hand side. For convective terms on the left hand side we can use CDS that is central difference scheme or upwind scheme. And once we have done our discretization.

A discretized equation for an interior node i can be represented in our standard template form that A_P superscript i that is coefficient A for current point P or i -th node $\phi_P + A_W \phi_{i-1} + A_E \phi_{i+1} = Q_i$. Instead of ϕ this should be capital A and this should be small ϕ . ϕ at $i+1 = Q_i$. Now let us note that these coefficients which we have put A_P , A_W and A_E they contain the contributions coming from the finite difference discretization of both convective and diffusive terms.

That is to say we can break it into 2 parts. $A = A^d + A^c$ and we will have a look at how do we get these components with different choices of our discretization scheme.

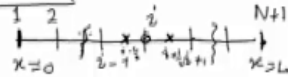
(Refer Slide Time: 04:48)

Advection-diffusion problem

Discrete eqn. at an interior node i :

$$A_P^i \phi_i + A_W^i \phi_{i-1} + A_E^i \phi_{i+1} = Q_i$$

$$A = A^d + A^c$$



Case 1 Diffusive term using CDS

$$\left\{ \frac{\partial}{\partial x} \left[\Gamma \frac{\partial \phi}{\partial x} \right] \right\}_i \approx \frac{\left(\Gamma \frac{\partial \phi}{\partial x} \right)_{i+1/2} - \left(\Gamma \frac{\partial \phi}{\partial x} \right)_{i-1/2}}{x_{i+1/2} - x_{i-1/2}}$$

CDS for first order derivative in numerator:

$$\left(\Gamma \frac{\partial \phi}{\partial x} \right)_{i+1/2} \approx \Gamma_{i+1/2} \frac{\phi_{i+1} - \phi_i}{\Delta x_i}$$

Now let us first have a look at the contribution which comes from the diffusive term. We have represented our discrete equation at node i as an interior node, node i let us draw a standard grid of the left end. $X=0$ and this is $X=L$ and if we use our vertex centre by a different scheme these nodes will be numbered 1, 2 and so on. So i -th node $i-1$, $i+1$ and the last node would be at $N+1$ if we had N number of divisions.

So we wrote our generic discrete equation at node i as A_P the subscript P stands for our grid node i to reinforce to put superscript i as well. So $A_P^i \phi_i + A_W^i \phi_{i-1} + A_E^i \phi_{i+1} = Q_i$ and we said any of these components at A they will consist of 2 parts. One which is contributed by the diffusive term and another one which is contributed by the finite difference discretization of convective term.

So now let us first take the case of diffusive term and how do we discretize it. We have already seen discretization in detail, so we will just put a summary here. We would use central differencing. So what we are looking at? We are looking at a term $\frac{\partial}{\partial x} \left(\Gamma \frac{\partial \phi}{\partial x} \right)$ at node i and we said look this we would discretize using the central difference formula. So we are focusing on our grid node i .

So let us chose these 2 intermediate node points $i+1/2$ and $i-1/2$. So in terms of these values, the outer derivative can be approximated as $\frac{\partial}{\partial x} \left(\Gamma \frac{\partial \phi}{\partial x} \right)$ at $i+1/2 - \frac{\partial}{\partial x} \left(\Gamma \frac{\partial \phi}{\partial x} \right)$ at $i-1/2$ / $x_{i+1/2} - x_{i-1/2}$. Now there are derivatives which appear in the numerator we would again use CDS. CDS for first order derivatives in a numerator so that will give us $\frac{\partial}{\partial x} \left(\Gamma \frac{\partial \phi}{\partial x} \right)$ at $i+1/2$.

This can be approximated in terms of the values at node point $i+1$ and i because $i+1/2$ is the midpoint i and $i+1/2$ $\gamma_{i+1/2} \phi_{i+1} - \phi_i$. Let me reward back to delta notation to delta X_i .

(Refer Slide Time: 10:12)

$$\left\{ \frac{\partial}{\partial x} \left[\Gamma \frac{\partial \phi}{\partial x} \right] \right\}_i \approx \frac{\left(\Gamma \frac{\partial \phi}{\partial x} \right)_{i+1/2} - \left(\Gamma \frac{\partial \phi}{\partial x} \right)_{i-1/2}}{x_{i+1/2} - x_{i-1/2}} \quad (2)$$

CDS for first order derivative in numerator:

$$\left(\Gamma \frac{\partial \phi}{\partial x} \right)_{i+1/2} \approx \Gamma_{i+1/2} \left(\frac{\phi_{i+1} - \phi_i}{\Delta x_i} \right) \quad (3)$$

$$\left(\Gamma \frac{\partial \phi}{\partial x} \right)_{i-1/2} \approx \Gamma_{i-1/2} \left(\frac{\phi_i - \phi_{i-1}}{\Delta x_{i-1}} \right) \quad (4)$$

$$\left(\frac{\partial}{\partial x} \left[\Gamma \frac{\partial \phi}{\partial x} \right] \right)_i = \frac{\left[\Gamma_{i+1/2} (\phi_{i+1} - \phi_i) \Delta x_{i-1} - \Gamma_{i-1/2} (\phi_i - \phi_{i-1}) \Delta x_i \right]}{\frac{1}{2} (x_{i+1} - x_{i-1}) \Delta x_i \Delta x_{i-1}}$$

Ex Simplify this expression (collect terms in ϕ with same subscript) together and obtain expression for A_P , A_W and A_E .

Similarly, $\frac{\partial \phi}{\partial x} \big|_{i-1/2}$ this can be approximated as $\gamma_{i-1/2} \phi_i - \phi_{i-1} / \Delta x_{i-1}$. We can substitute the expression for these expanded variables or the expansion for the derivatives appearing in numerator and thereby we can obtain the final form for the diffusive term $\frac{\partial \phi}{\partial x} \big|_{i-1/2} \gamma_{i-1/2} \frac{\partial \phi}{\partial x} \big|_{i+1/2}$ of i $\gamma_{i+1/2} \phi_{i+1} - \phi_i \Delta x_{i-1} - \gamma_{i-1/2} \phi_i - \phi_{i-1} \Delta x_i$ this whole thing divided by $1/2$.

This $x_{i+1/2}$ and $x_{i-1/2}$ their difference can be represented in terms of Δx_i and $\Delta x_{i+1/2}$ or rather we can simply write it as $1/2(x_{i+1} - x_{i-1}) \Delta x_{i-1}$. You can simplify this expression further and that I would leave it as an exercise to simplify this expression that is all that you need to do is collect terms in ϕ with same subscript together and obtain expressions for A_P , A_W and A_E of d .

So this I would leave as an exercise for you. It is a fairly simple exercise. So this A_P would basically you will obtain by collecting the terms which multiply ϕ_{i+1} and A_W would be the terms or the coefficient which multiplies after gathering the appropriate terms which multiply ϕ_i . Similarly, A_E that would represent the term which form the coefficient of ϕ_{i-1} .

(Refer Slide Time: 14:44)

... One Dimensional Advection-Diffusion

- ❖ CDS scheme for advective term leads to spurious oscillation if cell Peclet number > 2 .
- ❖ Upwind schemes for advective term
 - ❖ First order upwind scheme (highly diffusive)

Next let us move on to convective term and the convective term we have many choices available. We can use CDS scheme for an advective term when I just like to make a simple statement or generic statement that it can lead to what we call spurious oscillations that even though oscillation might be smooth function we would find lots of oscillations if the so called cell peclet number > 2 .

And to ameliorate the situation what we normally do is we use upwind scheme for advective term and there are various choices available. We will see one of them today that is the first order of upwind scheme. This term it eliminates the oscillations altogether, but it introduces what we call a diffusive term in the equation which gives us a highly damped profile so that is why people say that this particular scheme is highly diffusive.

What do you mean upwind scheme? Let us have a brief look at first CDS and then the upwind scheme.

(Refer Slide Time: 15:52)

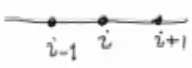
... Discretization of advective term

Option 5: central difference approximation

$$\left[\frac{d}{dx} (\rho u \phi) \right]_i \approx \frac{(\rho u \phi)_{i+1} - (\rho u \phi)_{i-1}}{(x_{i+1} - x_{i-1})} \quad (1)$$

Thus, $A_p^c = 0$

Ex. $A_w^c = ?$
 $A_E^c = ?$



Remark: Accuracy $\sim O(\Delta x^2)$ but it can produce oscillatory results if local (cell) Peclet number $Pe = (\rho u \Delta x) / \Gamma > 2$.

So discretization of advective term. There are various options. Option 1, what we did for the diffusive term we had used second order accurate CDS scheme. So shall we use the same scheme for our convective term as well. So if we use the central difference approximation then what we would get? The term which we are looking was d/dx of $\rho u \phi$ at node i . Since we are looking at node i in central difference approximation and terms involved would be at node $i+1$ and $i-1$.

So this can be approximated as $\rho u \phi$ at $i+1 - \rho u \phi$ at $i-1$ divided by the spacing between these 2 points that is $x_{i+1} - x_{i-1}$. So what is the contribution which we get for the convective term? At A_p^c and this case would be our contribution to the i -th node term $A_p^c = 0$ because on the right side of this equation there is no term which contains ϕ_i and how about ϕ_{i-1} -th term coefficient that is what we said that will give us the contribution to the A_w term what will that be? It will definitely be non 0.

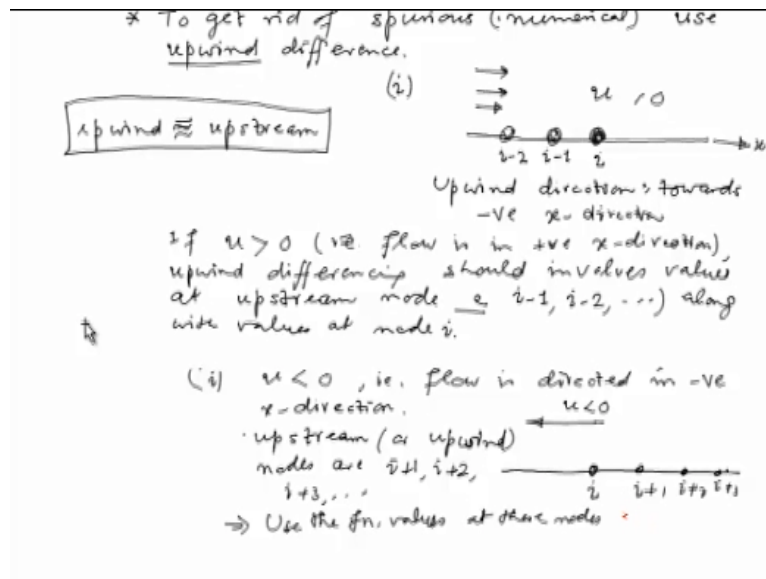
So can you rearrange and find the way out. This I would again leave as a very simple exercise. So find out what will be the contributions to this A_p^c and A_w^c and A_E^c if you use central difference approximation. Now the remark which we had earlier made was node though the CDS is second order accurate that is accuracy is second order, but it can produce oscillatory results.

If local we also use sometimes a term cell local peclot number. How do we define this local peclot number use a symbol Pe this is defined as $\rho u \Delta x / \gamma$. So this is definition of local peclot number. If this is greater than 2, we are in trouble when we use our CDS scheme.

So what is the solution then? To get rid of oscillations, rid of spurious numerical oscillations.

I want to reinforce the term spurious numerical because such oscillations are not actually the part of our exact solution of the problem. They come here simply because of our choice of the numerical approximation or finite difference approximation. We have to use what we call use upwind difference schemes.

(Refer Slide Time: 21:40)



Now here we have used the term upwind, what do you mean by upwind? Upwind comes from normal connotation of when we say the wind is blowing. So this is an advection diffusion problem and advection depends on the velocity. So I suppose this focuses on a specific point in a space, it could be a grid point i . Look at the velocity here. Now there are 2 possibilities. So X direction that u could be >0 or <0 . So $u >0$ would mean what? That is our velocity but flow is going from left to right.

So what would be our upwind direction? Upwind means that we go against the wind or against the flow velocity. So upwind direction would be towards negative x direction. So this is one case. So now in this case upwind difference scheme should involve what? This would involve values at the nodes which are not towards the right, but in a state they are towards the upwind or upstream side.

So some people also use in place of upwind. They also use this term very often upstream both are synonymously used. So in this case if u is >0 that is flow is in positive x direction, upwind differencing should involve values at upstream nodes that is which are the upstream

nodes. They will also involve the values of node i that is $i-1$, $i-2$ and so on along with values at node i .

So these are the nodes which we will have in this case which should be involved in our formulation and the physical logic given is, that is the effect of the velocity would be felt more because upstream whatever happens at the upstream nodes that affects the things happening at node i . So now our derivative approximation should involve the values at these nodes which are towards the left hand side in this case.

Now there could be another possibility. The second case would be that $u \leq 0$ that is flow is directed in negative x direction. So now in this case which ones will be our upwind nodes. This is the direction of the flow at a given node location i . So upstream nodes would be nodes $i+1$, $i+2$, $i+3$ and so on. So now in this case our upstream nodes or upwind nodes are $i+1$, $i+2$, $i+3$ and so on.

So use their values, function values at these nodes in finite difference approximation.

(Refer Slide Time: 27:05)

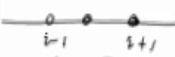
Upwind difference schemes

- First order upwind schemes (FDS or BDS)

$$\frac{\partial (\rho u \phi)}{\partial x} \approx \begin{cases} \frac{(\rho u \phi)_i - (\rho u \phi)_{i-1}}{x_i - x_{i-1}} & \text{if } u > 0 \\ \frac{(\rho u \phi)_{i+1} - (\rho u \phi)_i}{x_{i+1} - x_i} & \text{if } u < 0 \end{cases}$$

Advantage
 \Rightarrow No oscillations in numerical solution for any value Pe_i

Disadvantage
 * Excessive numerical diffusion (damping)



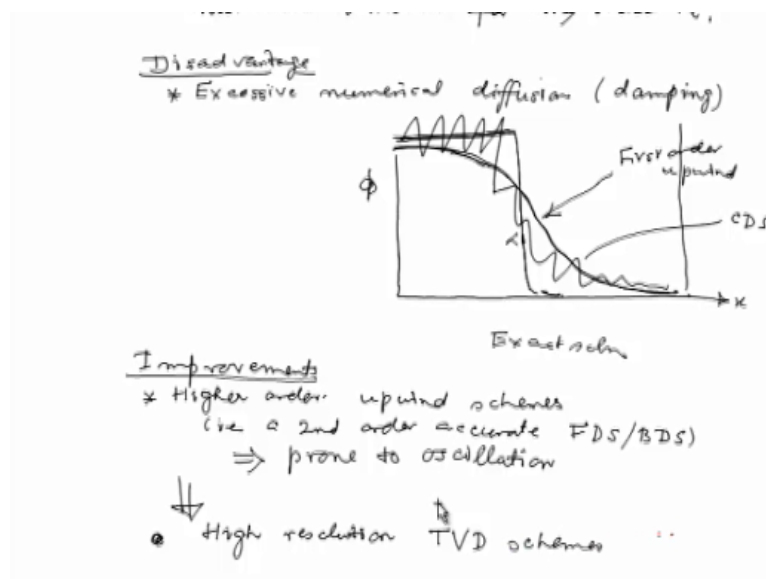
Now what are the choices which are available to us in the case of upwind difference schemes? The simplest choice is use of a single upstream node. First order upwind schemes which would essentially correspond to our first order what we call forward difference or backward difference schemes. So here basically we will approximate that d/dx at $\rho u \phi$ that should be approximated as what if our u is > 0 then we will have to take the values to the left of i that is we will have to go for backward differencing.

We will have $\rho u \phi$ at $i - \rho u \phi$ at $i-1/x_i - x$ of $i-1/$ and if u were >0 that is our flow is directed from right to left. So in that case we have to consider our upwind nodes becomes $i+i$ th node. So we will have the values $\rho u \phi$ at $i+1 - \rho u \phi$ at i/x of $i+1-x_i$. So in the first case we have used what we call at backward difference approximation and when used <0 we would have used what is exactly our forward difference scheme.

Now what are the nice properties of the scheme no oscillations. So this advantageous feature no oscillations in numerical solutions for any value of Pe . It does not matter how smaller how larger peclet number that is how strong is the convective term. Our first order upwind scheme would always give us a smooth solution and downside is the disadvantage which we should be aware.

That is excessive numerical diffusion or some people also prefer to call it a damping. Though it is very difficult to say that why is this particular term damping is used in this case.

(Refer Slide Time: 31:15)



And just to give you a view of graphical illustration. Suppose we are dealing with the advection diffusion of the sharp front, this is ϕ along this we have got x the sharp front is supposed holds to get a constant value up to certain point and suddenly takes a jump and the values go to 0. Now if you use so this is our so called exact solution. If we use a CDS based scheme it could give us this sort of a solution, lots of oscillations, but if we use upwind difference scheme we might have a fairly smoothed out solution.

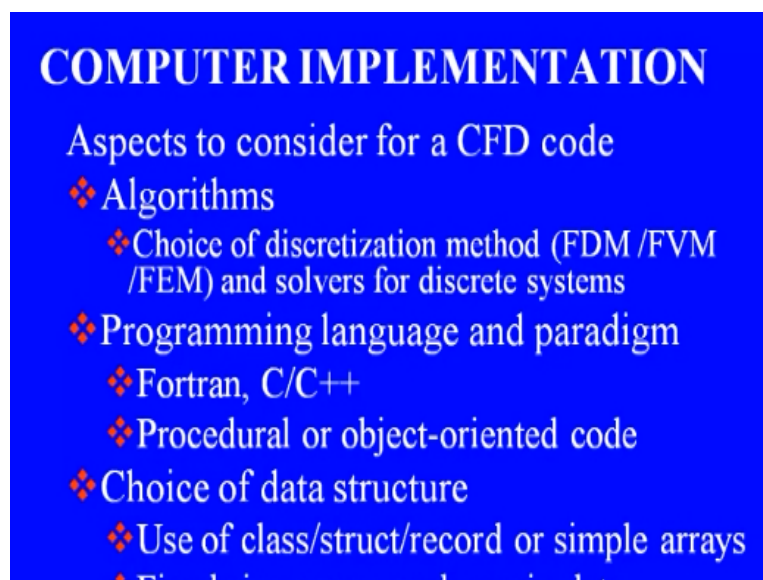
So you can easily say that this is as if we have introduced additional amount of diffusivity in our system. Now is there any way in which we can improve the situation. One-way improvement which you can immediately think of use a higher order upwind scheme. So improvements higher order upwind schemes. For example, a second order accurate FDS or BDS that is forward difference or backward difference scheme.

But surprisingly this is also prone to oscillations. So this is not real improvement then our next thing is to go for developed recently which are called high resolution TVD schemes. So this work similar to our upwind scheme, but we are not going to discuss these in detail at the moment. I would in fact leave it as reading assignment to you that is what is a summary that first order upwind scheme it works beautifully in damping out or in eliminating this spurious oscillation, but it is highly diffusive.

Second order upwind scheme based on a one sided FDS or BDS. It is again prone to oscillatory behavior. So in usual or commercial CFD course we go for what we call total variation diminishing or TVD schemes. This one possibility and other set of schemes are called essentially non-oscillatory schemes and our first order upwind scheme in fact is a member of these generic TVD schemes.

So for details, I would suggest that you refer to the books by Chung and Versteeg and Malalasekera on computational fluid dynamics. I will give you the full reference of these books towards end of the lecture.

(Refer Slide Time: 35:42)



COMPUTER IMPLEMENTATION

Aspects to consider for a CFD code

- ❖ Algorithms
 - ❖ Choice of discretization method (FDM /FVM /FEM) and solvers for discrete systems
- ❖ Programming language and paradigm
 - ❖ Fortran, C/C++
 - ❖ Procedural or object-oriented code
- ❖ Choice of data structure
 - ❖ Use of class/struct/record or simple arrays
 - ❖ Fixed size arrays or dynamic data

Now let us move on to our computer implementation aspects. So if you want to develop a CFD code, we have aspects which we must consider that is at first which algorithms we are going to choose to solve our partial differential equation governing a specific CFD problem. So we have primarily these 3 popular choices finite difference method, finite volume method and finite element. We have already discussed in this module in detail finite difference method.

So in the next lecture, we will discuss development of a simple one dimensional code based on finite difference. So in set of algorithm first choice is our discretization method and the next is the solvers for discrete system. So we have to make these choices. In fact, in our useful CFD code, we have got to provide for a set of solvers for the discrete systems then which language to choose and which programming paradigm.

The most popular languages in CFD applications have been Fortran and Fortran has been used since beginning since 1960s and more recently C and C++ have been claiming the space which was earlier exclusively occupied by Fortran. Then should we go for procedural approach which is the way the Fortran works or should we go for object oriented code which gives us what we call as easily modifiable program.

So we have to make a choice of these 2 and then the last thing which we have to think of is the choice of the data structures should we go for fixed size arrays or should we go for the simple records arrays or we go for what we call some more complicated classes, structures or records. Class and struct they are part of a C language. Records are now available as a feature in modern Fortran language.

So should we go for these ones bit more complicated ones or we go for simple arrays. The next one should we go for the fixed sized arrays that is to say we have when we are writing the program, we would think of a size or maximum number of nodes which we might use in discretization, make all arrays of that size or we can allow the array to grow that is for different problem different users might choose different grid sizes and then the grid size can be input and based on the grid size the sizes of these arrays, the structures or records could be considered.

(Refer Slide Time: 38:37)

... COMPUTER IMPLEMENTATION

ALGORITHMS

- ❖ Choose a discretization method (FDM /FVM /FEM) based on your preference / expertise
 - ❖ FDM and FVM are more popular in CFD than FEM (which is preferred in structural analysis)
- ❖ Solvers for discrete algebraic systems
 - ❖ Provide a set of iterative solvers and pre-conditioners
- ❖ Time integration schemes
 - ❖ Provide a set of one step and multi-step methods (both implicit and explicit)

Now as far as the algorithms are concerned whether you are going to choose finite difference finite volume or finite element. We will discuss finite volume and finite element in bit more detail in later modules. It would depend on your preference. Majority of the commercial codes in CFD are based on either finite difference or finite volume method. There are few codes available on FEM as well, but it is your call.

If you are developing your own code which one you are more comfortable with and this is for the common practice today that FDM and FVM finite difference and finite volume method, they are more popular in commercial CFD applications locations and finite element is preferred in structural analysis. There is no reason why one cannot go for FEM for writing a CFD code. Now next is solvers for discrete algebraic systems.

In our usual code we should provide a set of iterative solvers and of course these iterative solvers require what we call pre conditioners. Now these 2 aspects we will consider in our next module on solution of algebraic equations. Now I have not mentioned any direct solver because in our usual CFD when the grid size are pretty large number of nodes are very large we would rarely if ever use a direct solver.

Similarly, if you go to time dependent problem, we have to provide for a set of one step and multi step time integration methods both of implicit types and explicit types. So that depending on the problem, the user can opt for either implicit one or an explicit approach.

(Refer Slide Time: 40:26)

COMPUTER IMPLEMENTATION PROGRAMMING LANGUAGE AND PARADIGM

- ❖ Fortran or C/C++
 - ❖ Basic knowledge of both is required for a CFD code developer
- ❖ Procedural or object-oriented programming
 - ❖ Object-oriented is better software engineering (but has performance overheads)
 - ❖ Hence, procedural programming preferred

Now I would just like to point out here that as it regards the choice of their language you can chose either C, C++ or Fortran, but please remember if you want to become a CFD developer a basic knowledge of both these languages is required. The reason is very simple. Fortran has been used, in fact Fortran language was developed exclusively for scientific programming and so lots and lots of work is available in Fortran.

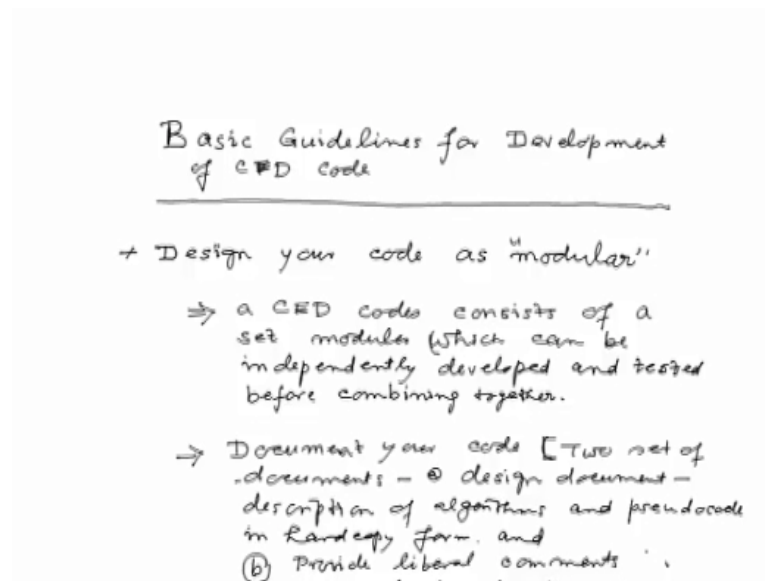
There are many libraries which are available in Fortran. Majority of the codes which are available in commercial and free domain they are based on Fortran and if you want to make use of those libraries, the user must know at least must have a working knowledge of Fortran and a good knowledge of C or C++. Now regarding procedural object oriented programming I would like to make one small comment that object oriented code is what we call it provides for better software engineering in the sense that the codes were easily modifiable, easily extensible because you will work in terms of classes.

Nuclear classes could be derived based on those classes without affecting the current implementation. So that is why it is called a better software engineering, but this object orientation has got severe performance overheads and CFD applications are what we call highly compute intensive. So we would like to minimize these non essential overheads to the extent possible.

Hence in CFD programming our prefer programming paradigm is procedural programming we would not use in fact majority of the software developers in CFD, they do not use object oriented code, procedural programming is preferred. Now few more guidelines or what I

would say the basic guidelines for software development whichever language you chose whether you chose Fortran or C and C++.

(Refer Slide Time: 42:53)



There are few basic thumb rules which you should follow. So basic guidelines for development of CFD code. In fact, these guidelines apply they are taken from software engineering. So they apply for development any computer code and they are equally pertinent to our CFD applications. So the first approach which we should use is what we call design your code as we call it modular that is to say the CFD code consists of a set of modules which can be independently developed and tested before combining together.

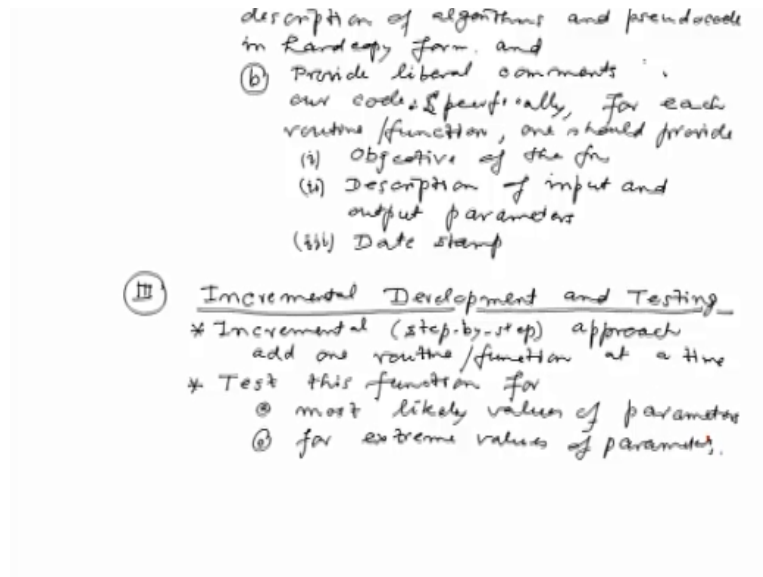
This is very similar to a normal engineering practice for instance if you want to develop or we want to design a car or manufacture a car what do we do. The entire car is not manufactured at one go. Different subsystems are designed separately, they are manufactured separately, tested separately and once each component has been tested then it is assembled together to get our final product that is our car.

Similarly, if we want to develop a nice CFD code we should break our big code into small, small modules or pieces, develop each module separately, test it and only then combine it together. And second guideline which you should follow is document your code. Now by documentation there should be 2 set of documentations. The first one is your design document which will contain a description of algorithm and pseudo code in hard copy form.

And the second and which is equally important is provide liberal comments. Comments or set

of statements or set of lines in a code which are there only for human programmer they are ignored by the compiler. So we have to provide liberal comments in our code that is specifically for each routine or function. In Fortran, we use the terms routine or sub routine in C or C++ we call them as functions. For each sub routine or functions one should provide.

(Refer Slide Time: 48:22)



In our first what that function does, objective of the function, second description of input and output parameters and you should also put a date stamp which would help you to keep track of when that particular routine was return or modified. So the first guidelines that you should develop your code as a modular code and the second aspect which we had was a documentation.

The third aspect which is equally important is incremental development and testing. So there are 2 ways you are very ambitious you know the knowledge of your domain, you thoroughly know your computer programming language, you also know your CFD algorithms and you might be tempted to go for write a big piece of code and then try and run it, it might in all probability 99% chances would be that your code would fail and you would find it very difficult to detect where you had made an error.

So what you should do is we should take an incremental step by step approach and its incremental which we can also call a step by step that is we add one routine at a time add one routine or function at a time. Test this function for a most possible or most likely values of parameters to verify that whatever algorithm has been coded in that function that works correctly and secondly for extreme values the parameters to detect which are the likely

situation and whichever algorithm might fail.

And what sort of error statements or exact cases we need to incorporate in our code. So if you follow this incremental approach you can start from a code of let say 10 lines you can easily build it step by step into a code which might run into millions or billions of lines. So even your very big commercial code which might contain millions of lines that has been built up using this incremental development and testing approach.

There might have been many programs who are involved in development, but each programmer was assigned task of developing a set of routines or set of functions or set of modules and in developing their modules or function they had followed or they had instructed in this industry to follow this incremental developmental and testing approach. The code has to be written in a structured format. It must be documented properly both in hard copy form as well as inside the code.

And then unit testing it ensures that our code has been debugged properly and it is ready for what we call final integration. So once we have unit tested thoroughly the task of integration would become a lot easier and we will get a reasonably error free code. We cannot say that it is a code which have developed or product which we have got that is perfect that is 100% error free that can never happen in a human endeavor, but by following this incremental approach we can minimize the occurrence of those errors.

So we can confidently say that we have developed a reasonably error free code. We are going to follow these guidelines and in developing a very simple one dimensional heat conduction code based on finite difference method that is what we are going to discuss in the next lecture. We will primarily focus on our approach so that you can use the same approach to develop bigger codes let say multi dimensional problems.

(Refer Slide Time: 54:49)

REFERENCES

- ❖ Chung, T. J. (2010). Computational Fluid Dynamics. 2nd Ed., Cambridge University Press, Cambridge, UK.
- ❖ Ferziger, J. H. And Perić, M. (2003). Computational Methods for Fluid Dynamics. Springer.

The references which we have refer today specifically for some algorithms. There are 2 which you can refer to computational fluid dynamic second edition Cambridge University Press by Chung T.J which is published in 2010. Similarly, you can also refer to Ferziger and Peric. This book is published in 2003 computational method fluid dynamics. In the next lecture towards end I will give you consolidated set of the books or references which would be very useful for learning finite difference method and the CFD application based in that.