

Computation Fluid Dynamics
Prof. Dr. Suman Chakraborty
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

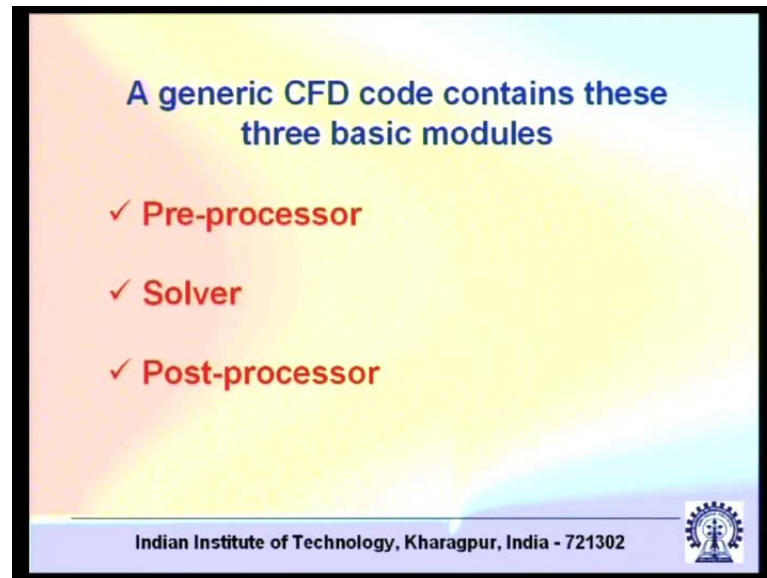
Lecture No. # 40
What is there in implementing a CFD Code

So far, we have learnt certain techniques for solving CFD problems. Now, the question is how to apply these techniques? CFD after all is a subject, where you want to apply the understandings that we have developed for solving practical problems. Once you are interested to do that, the first and foremost step will be to write a CFD code. These days, lots of CFD codes are available; some CFD codes are commercially available; some are open source code; and, some are in-house code, that is, code developed by various research groups for solving their tailor-made customised problem.

Now, whatever may be the case, it is often necessary to use the code rather than starting to write a code from the scratch. It is many times not a very bad idea though not a very idealistic approach, because nobody will give us a credit of reinventing a wheel. So, if somebody starts writing a CFD code at a particular stage and at the end, interested to solve a very challenging CFD problem, maybe it will take years to write the code and then solve the problem. And, one may be interested to use the code in a way to adopt the code to the particular requirement of the problem that he or she is interested to solve. And, that may be one of the practical approaches for modern day CFD, where you really have lots of CFD codes available.

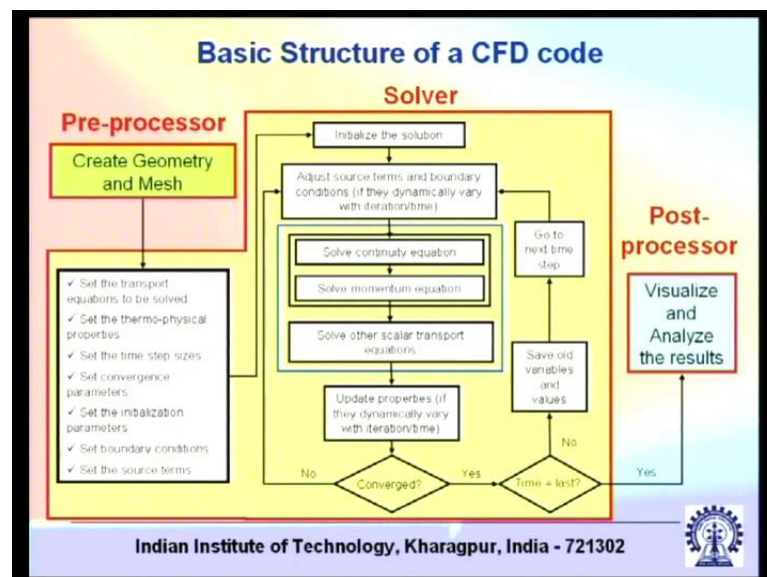
Question is if such a generic code is available with you, what are the intricate features that are common to most of these codes and how to go about the use of these codes for solving the problems? Now, because of obvious restrictions, we will not be considering any particular commercial code in this particular lecture for illustrating that how to go about that. But, we will consider a generic approach that can be used for any sort of CFD code and to go ahead with solving a problem.

(Refer Slide Time: 02:17)



Let us try to go through this presentation and let us see that what a generic CFD code contains. A generic CFD code contains three basic modules: one is a pre-processor; another is a solver; another is a post-processor.

(Refer Slide Time: 02:39)



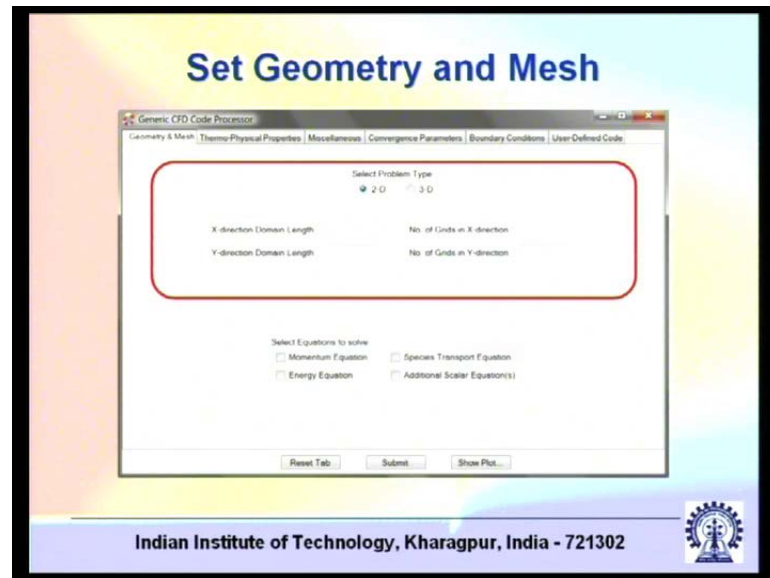
Let us go through the basic structure of a CFD code. Many times we are abstracted of this basic structure when we are using a code, because we are only using the user interface. But, if you see that, first if you consider the pre-processor, the pre-processor has different functionalities. One of the important functionality is to create geometry and

mesh for solving a problem. We will go through one or two examples to illustrate that. But then, what you need to do? You need to set the transport equations that need to be solved, because the code does not know that which equations you need to solve. So, you have to specify which equations you need to solve; set the thermo physical properties. If it is an unsteady problem, set the time step sizes.

Even if it is a steady state problem and you are using an unsteady mode, you can use a large single time step to convert an unsteady mode implementation into an equivalent steady state implementation. Set the convergence criteria or convergence parameters; set the initialization parameters; set the boundary conditions; and, set the source terms. These are certain pre-setting things that you could do. Then, you can initialize the solution when you go through the solver; then, adjust the source terms and boundary conditions if they dynamically vary with iterations. Remember that in CFD, mostly, we use iterative techniques. And, one of the common reasons is that in the cases in which we want to solve Navier-Stokes equations, which are non-linear equations, iterative techniques are much more suited than the elimination techniques.

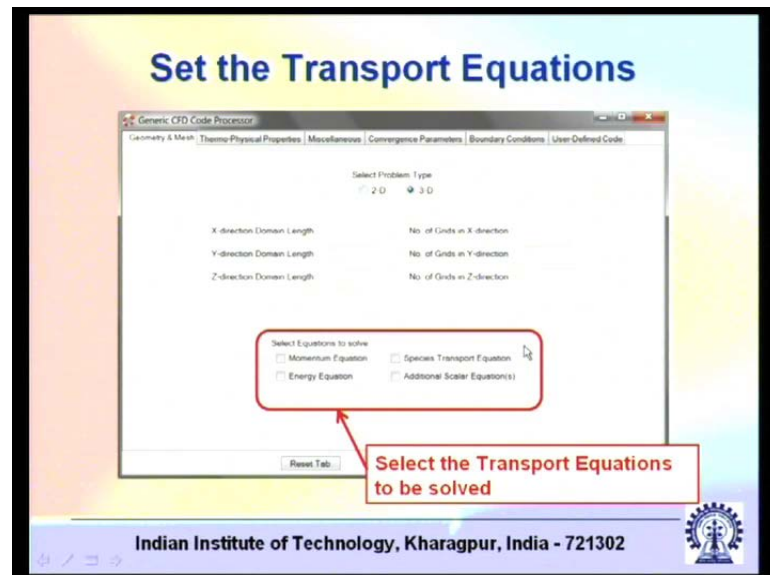
Then, for example, if we are interested for solving a fluid flow problem, then we must solve at least the continuity equation and the momentum equation. This is the basic requirement of solving the flow field. Then, you can solve other scalar transport equations depending on which other equations you need to solve. For example, if you are required to solve energy equation, then you can solve that; if you are required to solve the species conservation equation, you can solve another scalar transport; then, update the properties if they dynamically vary with iteration or time. So, some of the properties may not be constant, but they may vary with iteration or time and that you need to update; and then, you check for convergence. If convergence is not achieved, you go to the next time step or next iteration. In this way, iterations are done in a solver. And once iterations have converged and you have come to the end of the time domain, time equal to last; if it is yes, then you come out of this solver and you go to the post-processor, where you visualize and analyse the results in terms of graphical outputs like in terms of contour plots, vector plots like that.

(Refer Slide Time: 05:34)



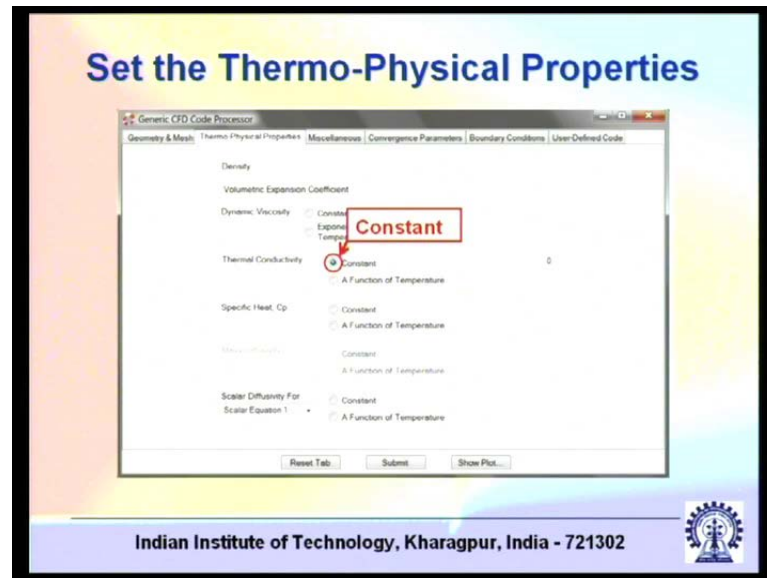
We just quickly go through these steps for a generic case; say, first you said geometry and mesh. So, for example, it is a g y for selecting whether it is a 2D or 3D problem. Then, the domain lengths along x and y direction; and, select the equations that you need to solve.

(Refer Slide Time: 05:50)



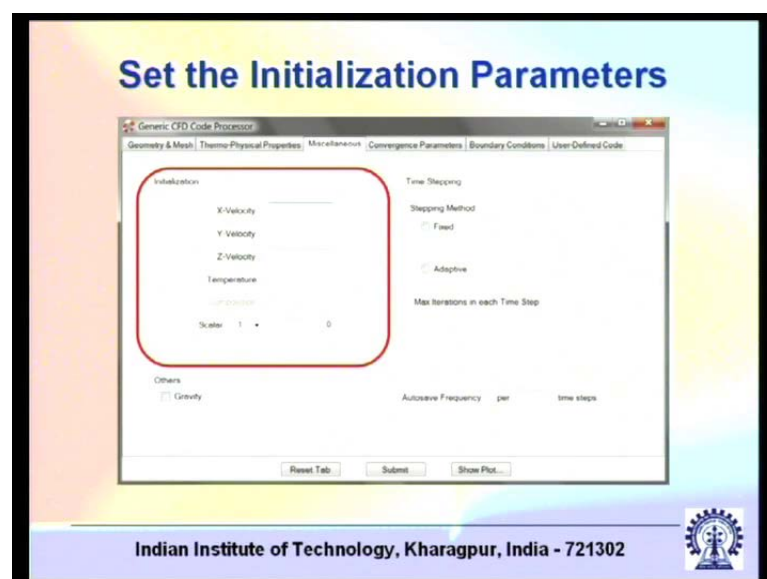
For example, here you solve momentum equation, energy equation and some additional scalar transport equation.

(Refer Slide Time: 06:02)



Then, set the thermo-physical properties. The thermo-physical properties may be constants; may be specified as function of the variable itself for example, the thermal conductivity; may be a function of temperature say linear function of temperature. So, these are certain facilities, which may be available with GUI; or even if it is not a GUI, it may be available with a standard user interface, not a graphical user interface. Then, it may be a programmable user interface through which you do or like this one where you have the graphical user interface. You can also have a step function of temperature, that is... These are some options, but you could have several other options.

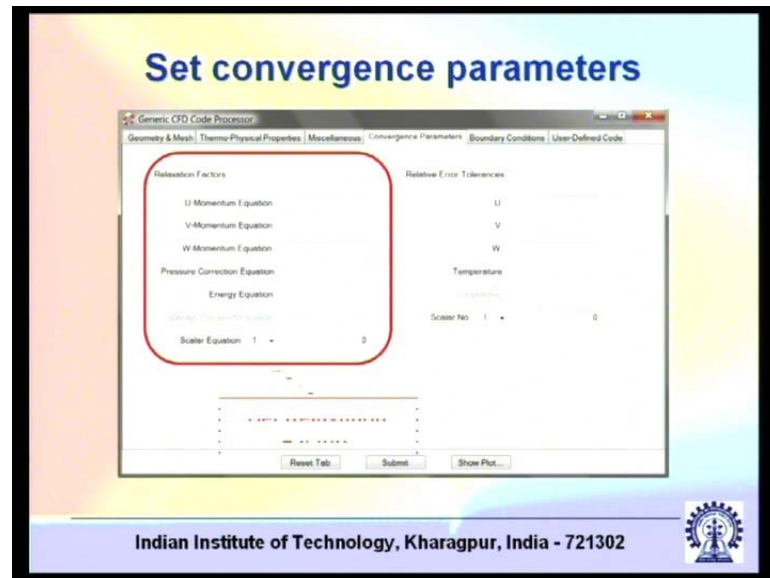
(Refer Slide Time: 06:43)



If none of these options fits, for example, if you say that you have a property variation, which is neither constant, non-linear nor piece-wise constant, but some other variations, then you have to write a user defined function for that. And, we will see later on that how to write user defined functions. Now, set the initialization parameters. For example, you can set the initial velocity and then you can set the time steps. So, you can have a fixed time step size or you can have an adaptive time step size. For example, a problem can have different time scales at different instants of time. For example, you have a domain; you suddenly heat the domain. So, what you see that initially, there will be rapid transients. So, you need to keep very small time steps to capture the changes within those small time intervals. But, later on when the system has adjusted that change to itself, then you may employ a larger time step size. So, you may use adaptive time step size.

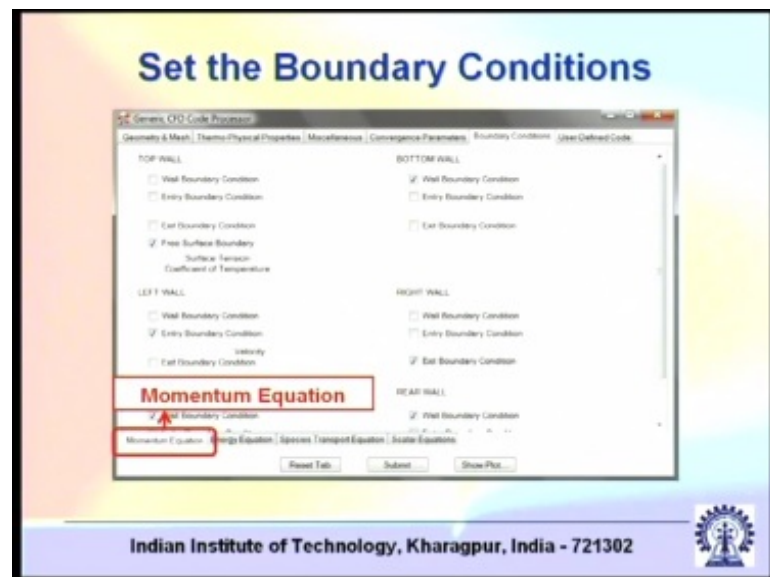
Then, you can set maximum iterations that you may allow for time step if it has not converged within those iterations. So, remember that within each time step, it is solving all the equations algebraically. So, it has gone through the iterations. So, if maximum number of iterations has been exceeded, then it may give you a message that convergence has not been achieved if it does not satisfy the convergence criteria. Then, certain additional things like if gravity is important for your problem as a body force, then you enable gravity and set the auto save frequency; that means, if you want to save results within interval of certain time steps. For example, it is a problem say where you have the time domain from 0 to 10 seconds and every 1 second you want to see the plots. So, every 1 second you want to store the result of your analysis. So, that is what we mean by auto-save frequency.

(Refer Slide Time: 08:42)



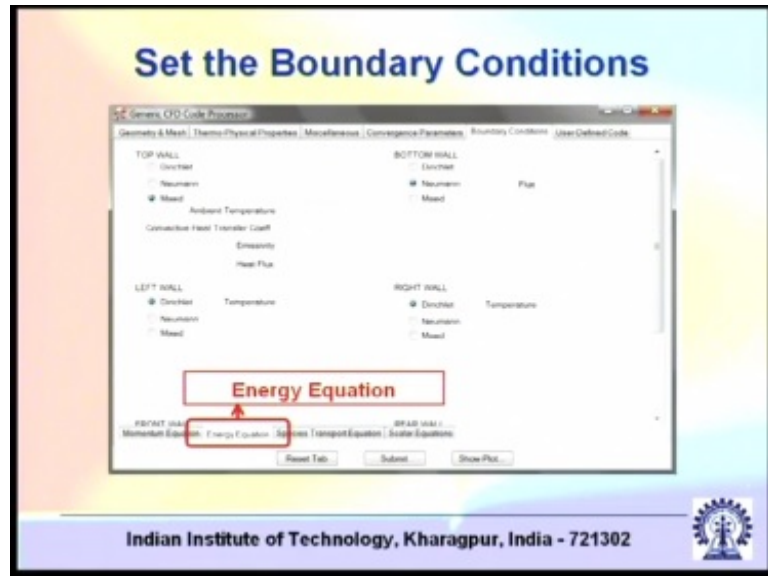
Then, set convergence parameters. So, convergence criteria and the relaxation factors; and, for the convergence criteria, you can use the relative error tolerance. So, we have already discussed that why we generally prefer relative error as criterion for convergence rather than absolute error.

(Refer Slide Time: 09:04)



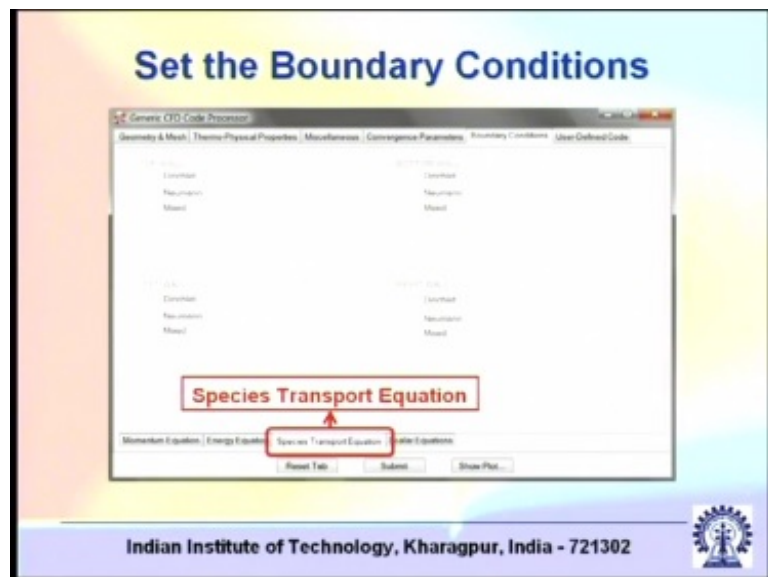
Then, set the boundary conditions. So, you can use different types of boundary conditions for example, momentum equation.

(Refer Slide Time: 09:16)



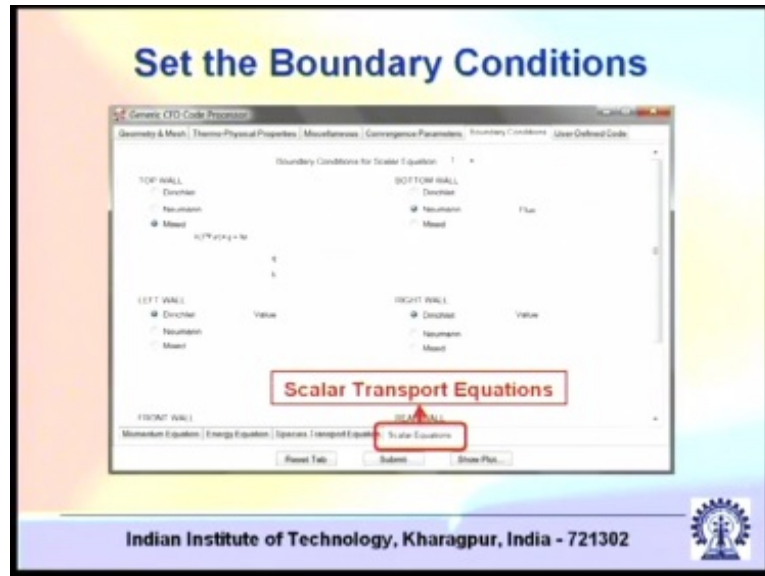
Then, energy equation – so, you can have boundary conditions for different equations.

(Refer Slide Time: 09:21)



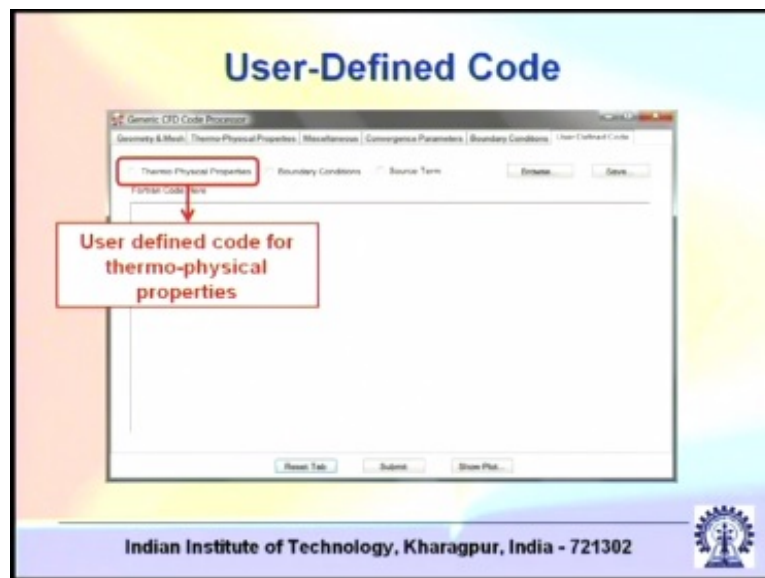
Species transport equation and for any other generic scalar equations.

(Refer Slide Time: 09:23)



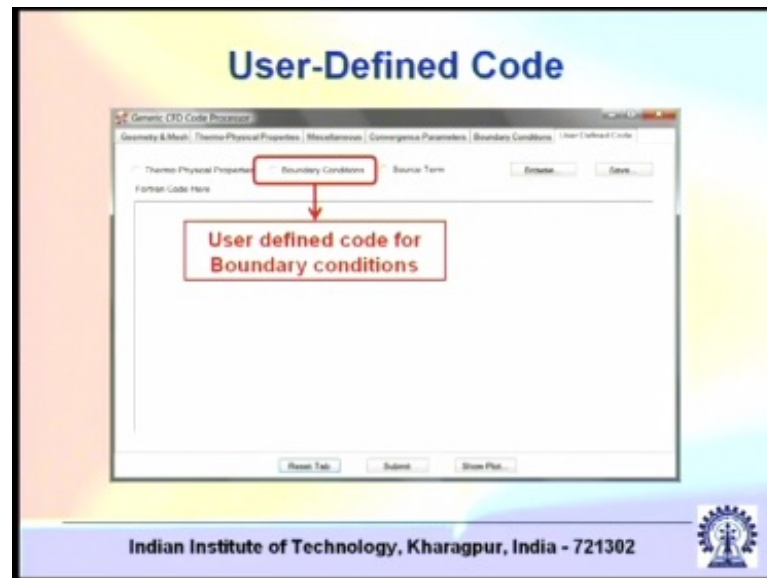
So, for each type of equations you can use different boundary conditions. We will go into the boundary conditions in more details through one or two examples that we will see later on.

(Refer Slide Time: 09:29)



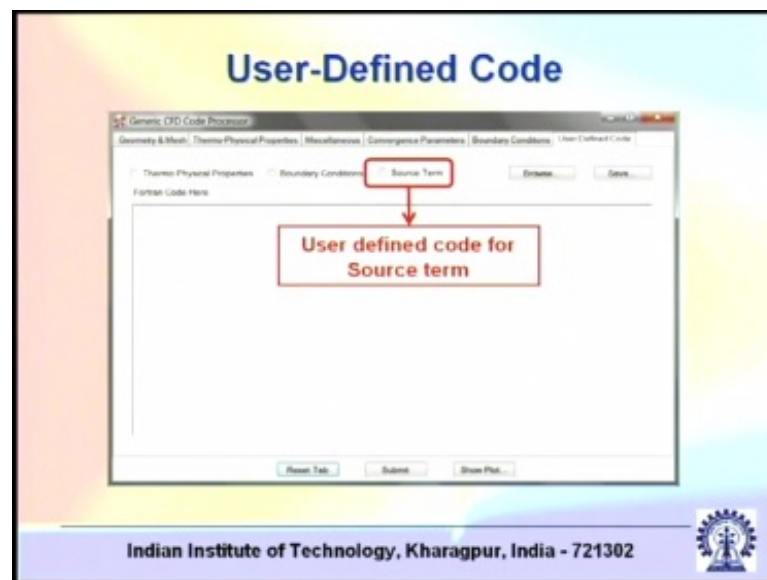
Then, let us come to the concept of user-defined codes. So, usually you use user-defined codes for accommodating something which is not there in the standard GUI or standard programmable interface. So, for example, if you have valuable thermo-physical properties and that you want to implement, that is one case.

(Refer Slide Time: 09:59)



Then, if you have user-defined code for boundary conditions; so, for example, if you are interested to have certain boundary conditions, which are not there in the standard GUI, then can you use the user defined codes.

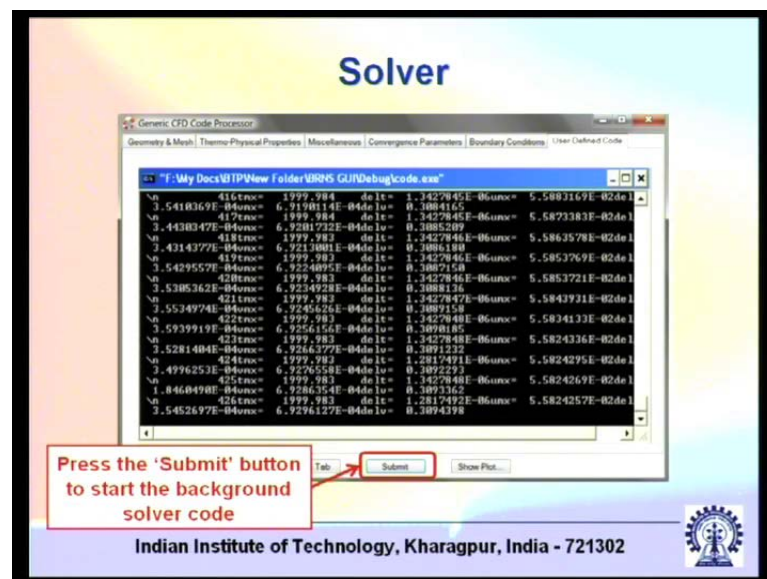
(Refer Slide Time: 10:15)



Or, maybe some special source terms. So, usually you can make the CFD code a fool by implementing some equations which are not the standard fluid flow equations. The equations may be... In whatever form, you may write it in a general conservative form. And then, whatever extra terms that have appeared, you dump that in the form of the

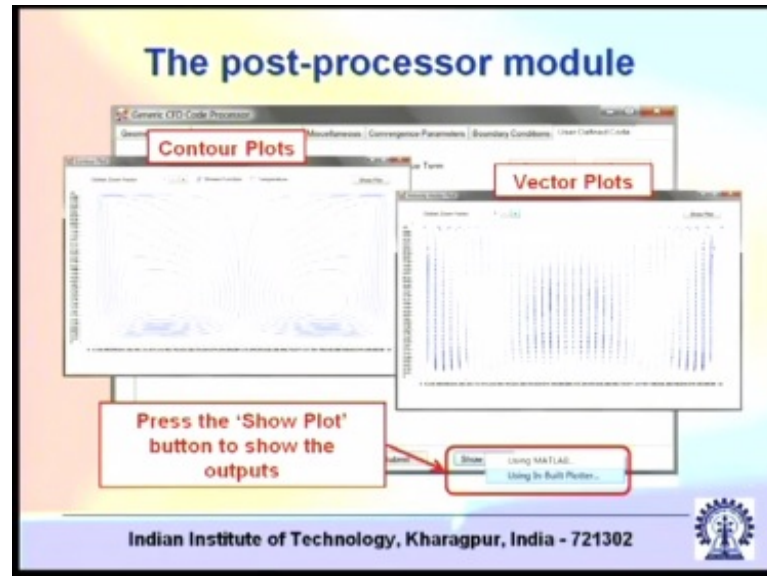
source term. So, we have already discussed that in CFD, when you write a governing equation, how do we make a CFD code understand that it is in the standard conservative form? We write it in the standard conservative form; it differs from one equation to the other in terms of a diffusion coefficient and a source term. So, if you specify the diffusion coefficient and a source term no matter how complicated it is, then you can implement any equation in that particular form to solve the problem. So, you may require user-defined code for implementing complicated source terms.

(Refer Slide Time: 11:21)



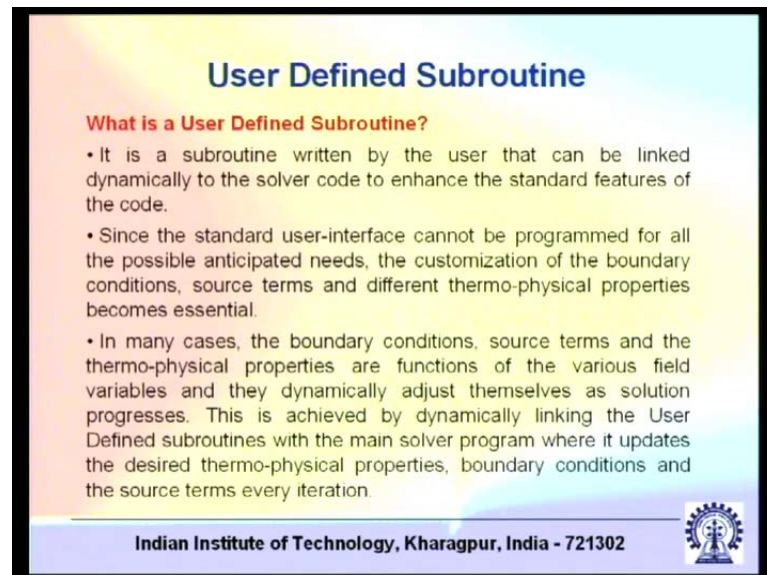
In that way, you can implement very complicated physical issues in a generic CFD solver. Then, you press the submit button to start the background solver code in this example and it will run and it will be showing you the level of iterations and how it is converging and all these.

(Refer Slide Time: 11:32)



Then, you can go to the post-processing parts. So, once the results are obtained, you can show contour plots. What are contour plots? These are isolines; that means, lines with constant values; different lines with different constant values. And, you can also show vector plots, where you have the velocity field for example.

(Refer Slide Time: 11:56)

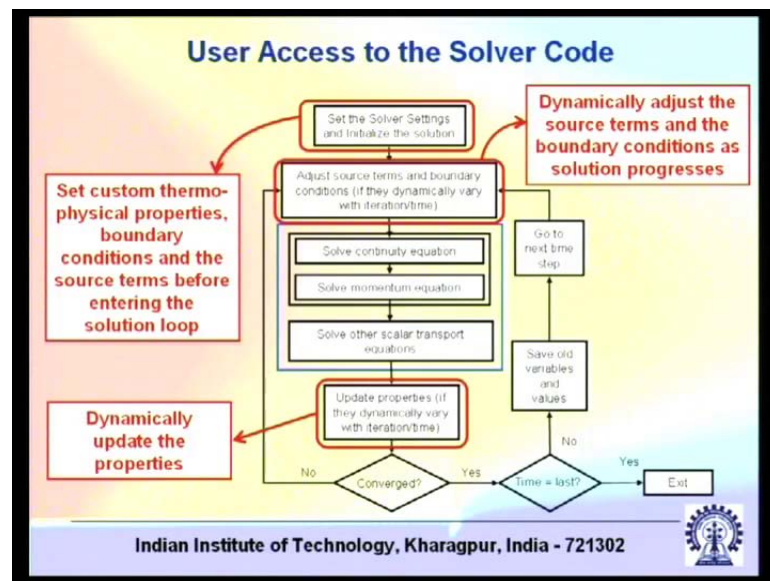


Now, let us go to the user-defined subroutine; or, in some cases, it is also called as user-defined function. Subroutine is a term borrowed from Fortran and function is a term borrowed from C, but in either way, it means the same. So, what is the user-defined sub

routine? It is a subroutine written by the user that can be linked dynamically to the solver code to enhance the standard features of the code. Since the standard interface cannot be programmed for all possible anticipated needs, see somebody who develops the CFD code develops the standard interface, either a graphical interface or a programmable interface. But, that cannot be customised to all possible needs, because different problems may have different intricate features with the governing equations, boundary conditions, etcetera.

And, in different cases, you require to customise all these, that is, the boundary conditions, source terms and thermo-physical properties. In many cases, these need to be dynamically adjusted during the solution procedure. This is achieved by dynamically linking the user-defined subroutines with the main solver program, where it updates the desired thermo-physical properties, boundary conditions and the source term with every iteration.

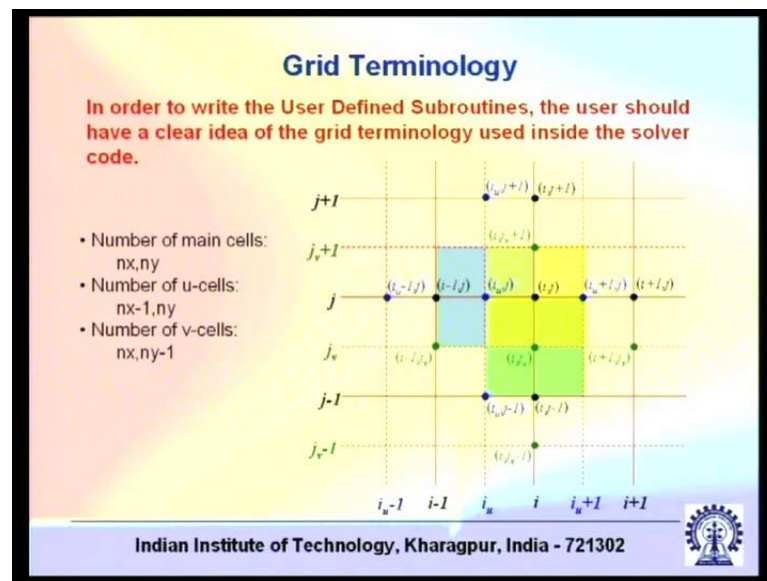
(Refer Slide Time: 13:11)



Let us quickly see that where do we require user access to the solver code. So, set the custom thermo-physical properties, boundary conditions and source terms before entering the solution loop. This you usually set as an initialization if you do not want this to be dynamically change; or, even if you want this to be dynamically change, you at least initialize those. So, that is one of the places. Then, you may require to dynamically adjust the source term and the boundary conditions as the solution progresses. That is

another place. And, you may require to dynamically update the properties. So, usually, the whole idea is that the solver of a good CFD code is very robust. So, you do not want to touch the solver; you do not want to get into the solver. Keep the solver as it is, but try to have stronger interaction or more intricate interaction with the solver by writing user-defined functions by facilitating the solver to address more complicated problems than the standard GUI will allow.

(Refer Slide Time: 14:18)



When you are writing the user-defined functions, you need to be careful about certain things like if you are using a grid terminal... if you are using staggered grid for example, what is the grid terminology? If you are using a collocated grid, then a terminology is relatively straightforward. But, if you are using a staggered grid, you do not have the same locations where you solve for the momentum equations and the other scalar equations. So, you basically have different control volumes. So, in this particular figure, wherever you have i with subscript u , that indicates that this is the location, this is the index where you are solving the x momentum, u momentum equation.

Similarly, j with index v will indicate that you are trying to solve for the y momentum equation. i index for x momentum and j index for y momentum. And, in this particular example, we are using certain nomenclature; it may be better to introduce that. So, number of main sales n_x comma n_y ; n_x along x , n_y along y ; number of u-cells is n_x minus 1 comma n_y . So, because it is staggered, number of sales for x momentum

equation solution will be one less than the total number of cells for the other scalar equations. Similarly, number of v cells is n_x comma n_y minus 1.

(Refer Slide Time: 15:50)

Using loops while writing the user defined subroutines:


Writing user-defined source terms:

Scalar Eqn.	u-momentum Eqn.	v-momentum Eqn.
do i=2,nx-1 do j=2,ny-1 ... end do end do	do i=3,nx-1 do j=2,ny-1 ... end do end do	do i=2,nx-1 do j=3,ny-1 ... end do end do

Writing user-defined boundary conditions:

Boundary	Scalar Eqn.	u-momentum Eqn.	v-momentum Eqn.
Bottom	i=1,nx, j=1	i=2,nx, j=1	i=1,nx, j=2
Top	i=1,nx, j=ny	i=2,nx, j=ny	i=1,nx, j=ny
Left	i=1, j=1,ny	i=2, j=1,ny	i=1, j=2,ny
Right	i=nx, j=1,ny	i=nx, j=1,ny	i=nx, j=2,ny

Indian Institute of Technology, Kharagpur, India - 721302



Now, you may use loops while writing the user-defined subroutines. So, this is an example where you are writing loops using fortran as a structure. There are certain user-defined functions which are neither fortran nor C; they have their own structures. So, this is just a generic example. Do not try to think that this is the user-defined function syntax for all. We are not trying to teach you a syntax. The whole intention is not to teach you how to run a particular code, but to illustrate that what is the basic principle in running different codes; what is the common principle.

Here for example, you can use this do loop and end do. So, this is just like a fall loop instead of a do loop if you use C. So, if you have a scalar equation source terms, see the source terms are defined for only interior grid points not the boundary points, because source terms are for control volumes; boundary terms do not belong to any control volumes. So, that is why you see that i starts from 2. So, in the scalar equation, i starts from 2; it ends with n_x minus 1. So, it ensures that it is only for the interior grid points. For the u-momentum equation, on the other hand, you see i starts with 3, because it is staggered; it is shifted by 1. So, i starts with 3.

On the other hand, for the v-momentum equation, j starts with 3. This is for the source term for the interior grid points. And for different boundaries, you have different values

of i and j . For example, if you have a square domain or a rectangular domain, a rectangular domain will have top, bottom, left, right like that. So, we are giving an illustration through a rectangular domain. So, i equal to 1 to n_x for the bottom. So, remember that horizontal line we are considering as x and vertical line as y . So, bottom i equal to 1 to n_x and j equal to 1. The top will be j equal to n_y ; left will be i equal to 1 and right will be i equal to n_x . So, this is for the scalar equation. For the u -momentum equation, i will be from 2 to n_x , not 1 to n_x . That is the only difference. And, for v -momentum equation, j will be from 2 to n_y . That is the difference.

(Refer Slide Time: 18:16)

User Defined Subroutine Some Representative Examples

User Defined Source: Gravity acting in x-direction

In order to write the source term, we have to calculate the temperature value at the staggered u-cells. In order to do that, we linearly interpolate the temperature value.

$$T_{u,j} = T_{i-1,j} + (T_{i,j} - T_{i-1,j}) \left(\frac{x(i_u) - x(i-1)}{x(i) - x(i-1)} \right)$$

$$= T_{i,j} \times f_x(i) + T_{i-1,j} \times (1 - f_x(i)) \quad \text{where, } f_x(i) = \left(\frac{x(i_u) - x(i-1)}{x(i) - x(i-1)} \right)$$

which is the weighted interpolation variable along x and is defined in the solver code**

```

if (equ == x_mom) then
  do i=3,nx-1
    do j=2,ny-1
      Tav=fx(i)*T(i,j)+(1-fx(i))*T(i-1,j)
      con(i,j)=con(i,j)+rhoinitial*g*beta*(Tav-tref)
    end do
  end do
endif

```

**** It is important to note that in order to efficiently write user-defined functions, the user should have a clear idea of the structure of the solver code and the important functions and variables used therein.**

Indian Institute of Technology, Kharagpur, India - 721302

Let us go through some representative examples for user-defined functions. Let us consider that you require a gravity acting in a x -direction. Let us say that the code has gravity acting along y -direction. Now, you have to implement gravity acting along x -direction; of course, you can swap x and y if you want. But, if you want to implement it by yourself; let us say the code has no provision of implementing a gravity source term and you want to implement that. So, where do you want to implement the gravity source term? You want to implement the gravity source term in the x -momentum equation; and, the gravity source term is let us say it is because of the natural convection. So, it is like $\rho g \beta$ into T minus t infinity or t reference. So, just recall from your basic understanding of natural convection that if you use a Boussinesq approximation, then you can write the gradient of the variation of density in terms of the variation of temperature. And, that is expressed in terms of volumetric expansion coefficient. So, it will come out

to be $\rho g \beta (T - T_{ref})$ or $T - T_{inf}$ depending on whether it is an enclosure or it is a problem with atmospheric boundary layer.

Whatever it is, now, you have to keep one thing in mind, if you are using a staggered control volume, what is the challenge here? See you have the temperature solved at the main grid points; whereas, you have to specify the x-momentum source term at the staggered location. So, you have to interpolate the temperature at the staggered control volume location as a function of the temperature at the main grid points. That is what we are doing in this example. So, for example, we can linearly interpolate the temperature. So, that is what we are schematically showing here; you have x_{i-1} , x_i ; and, we require to interpolate it at x_u . That is the staggered location. And, we require to find out what is $T_{i,u}$. So, for that, we are using this linear interpolation formula. And, in this linear interpolation formula, we are using term f_x , which is like $x_u - x_{i-1} / (x_i - x_{i-1})$. This is a particular variable, which in the code that is being used, that variable is already defined. So, if you know that there are certain variables, which are already defined in the code, you may make use of that to enhance the calculations.

Once you do that, let us go through the structures. See if equation logical equal to x mom; that means, if it is x momentum equation. How will the code know that for which equation you are interested to write the source term. So, for that, there must be one if. So, one if for letting the code know that for which equation; then, you go to the loop and write the corresponding average T. See $i = 3$ to $n_x - 1$, because it is staggered along x; $j = 2$ to $n_y - 1$, because it is not staggered along y; it is staggered along x only. So, then we are writing this con is like the constant source term like that $s_c + s_p \phi_p$; that s_c we are calling as con. So, that $con_{i,j} = con_{i,j} + \rho_{initial} (g \beta (T_{avg} - T_{ref}))$, that is, whatever was initialized plus some rho initial, that is, the reference density into g into beta into T average minus t reference. So, that is how you simply implement the natural convection.

(Refer Slide Time: 22:15)

User Defined Subroutine Some Representative Examples

User-Defined Boundary Condition: Thermo-Solutal Marangoni Convection

The boundary condition for the top surface looks like: $\mu \frac{\partial u}{\partial y} = \sigma_T \frac{\partial T}{\partial x} + \sigma_C \frac{\partial C}{\partial x}$

The discretized boundary condition looks like:

$$\mu_{i,s_y} \frac{(u_{i,s_y} - u_{i,s_y-1})}{y_{s_y} - y_{s_y-1}} = \sigma_T \frac{(T_{i,s_x} - T_{i-1,s_x})}{x_i - x_{i-1}} + \sigma_C \frac{(C_{i,s_x} - C_{i-1,s_x})}{x_i - x_{i-1}}$$

The user defined boundary condition looks like

```
do i=2,nx
  u(i,ny) = u(i,ny-1) + (y(ny)-y(ny-1)) * amu*(ny) *
    (SigmaT*(T(i,ny)-T(i-1,ny)) / (xi(i)-xi(i-1)) + SigmaC*(C(i,ny)-C(i-1,ny)) / (xi(i)-xi(i-1)))
end do
```

Indian Institute of Technology, Kharagpur, India - 721302

Then, another example; let us say that you have a thermo-solutal convection; what is thermo-solute convection. So, we know that there are cases in which you have free surface flows, that is, you have an interface between two fluids or a free surface where the interface is between a liquid and a gas. Now, you may have a temperature gradient on the surface of the liquid. And, for most liquids, the surface tension is a strong function of temperature. So, if you have a gradient of temperature, it induces a gradient in surface tension. And, because of the gradient of surface tension, it can induce an interfacial flow, which we call as Marangoni flow.

Now, you can have a temperature gradient; you may also have a concentration gradient if you have a multi-component system. In the multi-component system, you may have a concentration gradient. And, in the concentration gradient, the surface tension may also be a strong function of concentration. So, you can write the boundary condition in this way. So, the free surface should be free of shear; that means, the shear due to viscosity is balanced by the shear due to surface tension. So, what we are writing, this is like $\mu \frac{du}{dy}$ is for the viscous shear. And, shear due to surface tension, you can write for example, $\frac{\partial \sigma}{\partial x}$. So, $\frac{\partial \sigma}{\partial x}$ is $\frac{\partial \sigma}{\partial T} \frac{\partial T}{\partial x}$, where σ is the surface tension coefficient. So, $\frac{\partial \sigma}{\partial x}$ is a shear stress because of gradient of surface tension, where σ is a surface tension coefficient. So, y for example, is the direction normal to the interface and x is the direction tangential to the interface. That is what nomenclature we are considering.

So, you have $\frac{d\sigma}{dx}$ as $\frac{d\sigma}{dT}$, which we call as σ_T or temperature coefficient of surface tension; that means, how surface tension changes with temperature; that is, the rate of change of surface tension coefficient with temperature. So, this is $\frac{d\sigma}{dT}$ into $\frac{dT}{dx}$ or σ_T into $\frac{dT}{dx}$ plus $\frac{d\sigma}{dC}$ into $\frac{dC}{dx}$; $\frac{d\sigma}{dC}$ is σ_C . So, you have two coefficients of surface tension: one function of temperature, another temperature of concentration. And, this is what you are writing along the tangential direction to the interface; this is again a stress tangential to the interface; and, these two tangential stresses are balancing each other. But, this tangential stress is a function of the normal gradient of velocity; that is what Newton's law of viscosity tells you. This is true only if you have v equal to 0; otherwise, you have $\mu \frac{du}{dy}$ plus $\frac{dv}{dx}$. But, if you have v equal to 0, that is, if you have an interface where it is oriented along x , you have no penetration boundary condition across the interface. So, v equal to 0.

The discretized boundary condition will look like this. So, what you first do, you discretize the boundary condition. So, you write this in terms of corresponding differences. So, $\frac{du}{dy}$ is like $\frac{\Delta u}{\Delta y}$. So, like $u_{i,n} - u_{i,n-1}$; that is $\frac{\Delta u}{y_n - y_{n-1}}$; that is, Δy equal to σ_T into $\frac{dT}{dx}$ like $\frac{\Delta T}{\Delta x}$ plus σ_C into $\frac{\Delta C}{\Delta x}$. And then, you rearrange this equation algebraically. What is your objective while you rearrange? While you rearrange, you write $u_{i,n}$; y_n is the top most one. So, we are considering the top surface as a free surface as a function of $u_{i,n-1}$. So, as we have already discussed, you write the boundary as a function of the interior, not interior as a function of the boundary, because you are giving the boundary condition, not the interior condition. So, boundary should be expressed as a function of the interior. So, you can see that this do loop in this do loop, you write the boundary condition in this way, where you basically discretize this particular boundary condition.

(Refer Slide Time: 26:52)

Representative Case Studies

	Transport Equations
Lid Driven Cavity	Momentum Equation
Natural Convection inside a Cavity	Coupled Momentum & Energy Equation

Indian Institute of Technology, Kharagpur, India - 721302

Now, we will go through some representative case studies. So, two of the interesting case studies that we will go through will be, one is lid-driven cavity flow. So, there is a cavity in which the top lead is driven with a particular velocity and that induces a motion. So, this is a standard benchmark problem in CFD, which is known as lid-driven cavity problem. And then, a natural convection in a cavity, where you need to solve coupled momentum and energy equation.

(Refer Slide Time: 27:26)

Lid-Driven Cavity

Problem Definition

Thermo-physical Properties

Density, ρ (kg/m ³)	997
Viscosity, μ (Pa.s)	8.55×10^{-4}

Initialization Parameters

u_0	0
v_0	0

Time Stepping Method

$\Delta t = 10$
 $t_{total} = 500$

$u = 0.1 \text{ m/s}, v = 0$

$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$

$\frac{\partial \rho u}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) = \nabla \cdot (\mu \nabla \mathbf{u}) - \frac{\partial p}{\partial x}$

$\frac{\partial \rho v}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{v}) = \nabla \cdot (\mu \nabla \mathbf{v}) - \frac{\partial p}{\partial y}$

0.2 m
40 Grid Points

0.1 m
40 Grid Points

Autosave Frequency
- Per 50 time steps

Relative Error Tolerance

u mom	1×10^{-6}
v mom	1×10^{-5}

Relaxation Parameters

u momentum	0.5
v momentum	0.5
Pressure correction	0.5

Indian Institute of Technology, Kharagpur, India - 721302

Let us describe the problem of lid-driven cavity. You have a rectangular cavity; for simplicity, we are considering a rectangular geometry, but you can use different geometries while solving the problem. The lid-driven cavity is a standard problem, where you always use a rectangular geometry; or, in some cases, you may also use a cylindrical geometry. But, these are standard geometries. Now, in this example, we are considering the x as the horizontal direction and y as the vertical direction, and you are using different grid points along x and y . One important thing to mention is that the number of grid points that you are using, will it be uniformly distributed or will it be non-uniformly distributed? It depends on the problem. So, you can use uniformly distributed grids in general if you are solving a simple problem. But, just like, you may use non-uniformly distributed time steps.

Similarly, you can use non-uniformly distributed grids. What is the situation in which you want to use non-uniformly distributed grids? Say you have boundary layer flow. So, when you have a boundary layer flow, you require to capture the strong gradients of velocity and temperature or whatever depending on what type of boundary layer it is within the boundary layer. To do that, you require to have a large number of grid points within the boundary layer, at least a substantial number of grid points. So, if the gridding is such that you do not have any grid point within the boundary layer, say one is at the boundary and the next one is outside the boundary layer, then you are not able to capture the physics within the boundary layer. So, where you have strong gradients in variables, there you have to cluster fine grids; and, where you do not have strong gradients, there you do not unnecessarily keep fine grids; there you can keep coarse grids. But, the art of meshing is that you do not suddenly jump from fine grid to a coarse grid.

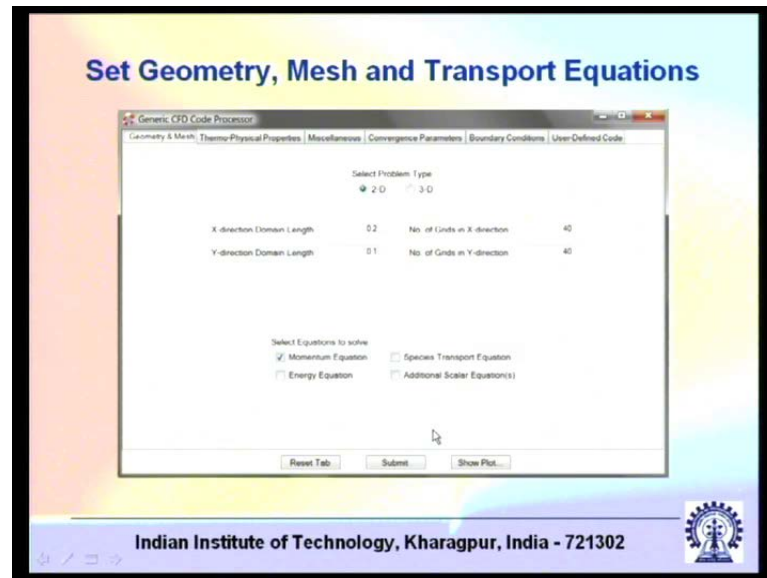
You go through several transitions, because a sudden jump from a fine grid to coarse grid can make the system of algebraic equations that are being solved unstable. So, there is a relationship with the nature of meshing and with the nature of algebraic equations that get generated out of it. So, if there is a variable meshing, one has to have a gradient. So, refined mesh at locations where you want to capture strong gradients; and coarse are meshed where you do not want to capture such strong gradients. Now, the question will come that then how will you ensure that your results are fine, because you may use different grid points and you may end up in getting different solutions. The answer is that at the end, you have to establish that your results are grid independent; that means, if you

use a refined grid and you get a solution, then you have to show that with further refinements of different levels your solution does not change. And then, that is called as a grid independent solution. So, grid independent solution is one of the bench marks of accuracy in a CFD problem implementation.

Now, let us look into this problem definition. So, you have the top boundary being pulled towards the right with a velocity 0.1 meter per second; the left and the right walls are subjected to 0 velocities. So, no slip, no penetration boundary condition; and, the bottom wall is also subjected to no slip, no penetration boundary condition. So, now, you have to model the problem. So, what is modelling? Modelling is basically idealization of a practical problem with certain assumptions. That is what is modelling. So, you model this problem by considering that it is an incompressible flow for example, or it is a special case of an incompressible flow, where you have the density as a constant, because you may also have variable density incompressible flow. But, this is special case, where you have density as a constant. And, viscosity is another property that you require. So, for solving any problem, you first have to ascertain that which properties you require. That depends on which equations you need to solve. So, you can look into these equations and see that you require the density and viscosity as the two properties. Then, what are the variables that you are solving? You are solving u , v and pressure. Pressure you are solving as a relative variable if you are using simple or simpler algorithm.

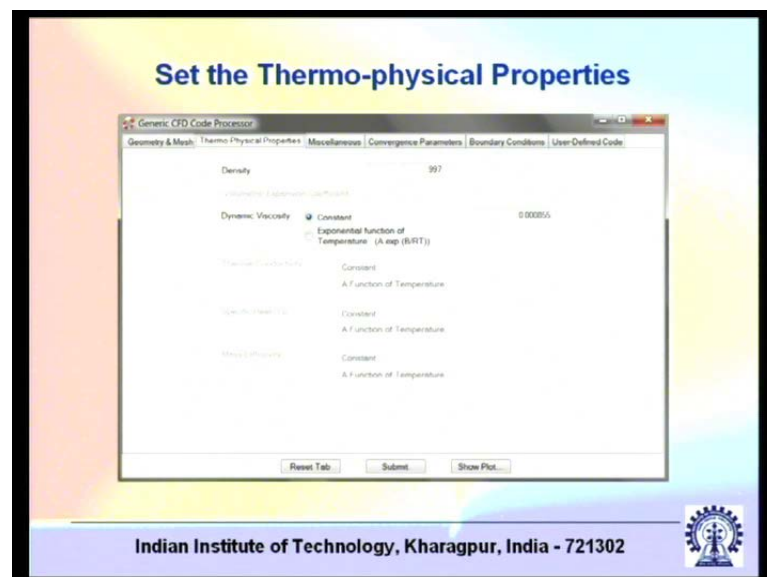
Now, initialization parameters, you may set the velocities as 0; you may use the time step, ΔT of 10 and final of 500. These are generic examples. So, do not think that these are for getting highly refined solutions. These are just for illustration to begin with. Then, autosave frequency – per 50 time steps you can have autosave frequency; that means, at the end, it will only store the result, because 50 time steps will make it 500. Relative error tolerance – x momentum and y momentum, that is, u and v momentum, you set as 10^{-5} . These are just some examples; you may make it more strict more relaxed depending on your requirement. Some relaxation parameters you can put. We have already discussed in the context of Navier-Stokes equation that what are the relaxation factors and why do you need that to control the convergence. So, these are some examples. And, in this particular example, you take 40 grid points along x and y .

(Refer Slide Time: 33:16)



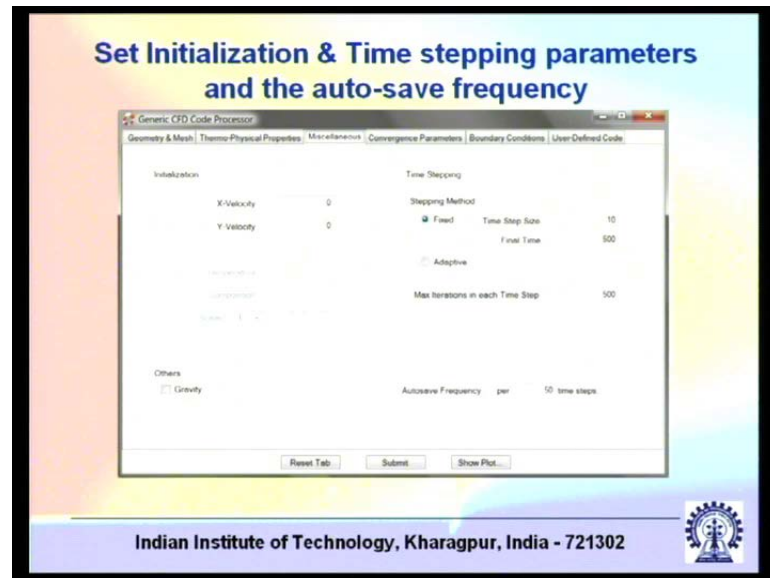
So, let us quickly go through the geometry. So, we set the problem as 2D; set the x-direction domain length, number of grids along x; y-direction domain length and number of grids along y; and, select that you require solve only momentum equation.

(Refer Slide Time: 33:31)



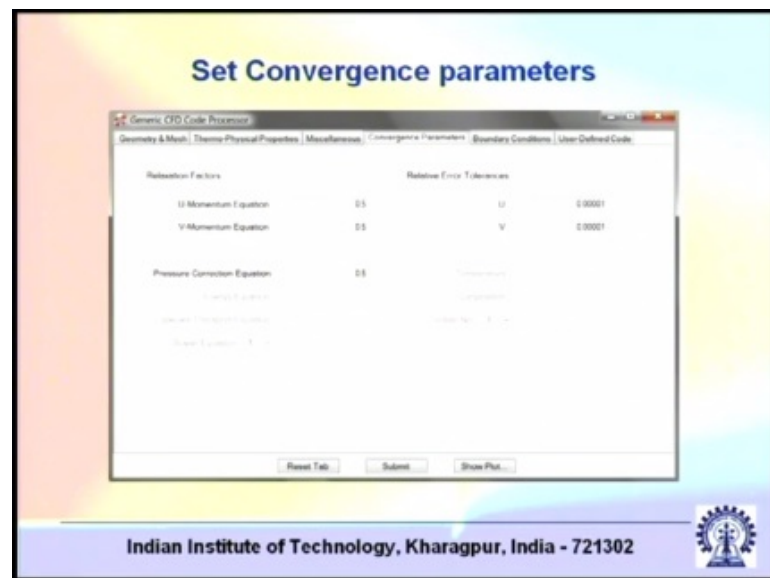
Once you select, that you require to solve through momentum equation automatically accommodates the continuity equation, because these are all coupled equations and algorithm takes care of that.

(Refer Slide Time: 33:42)



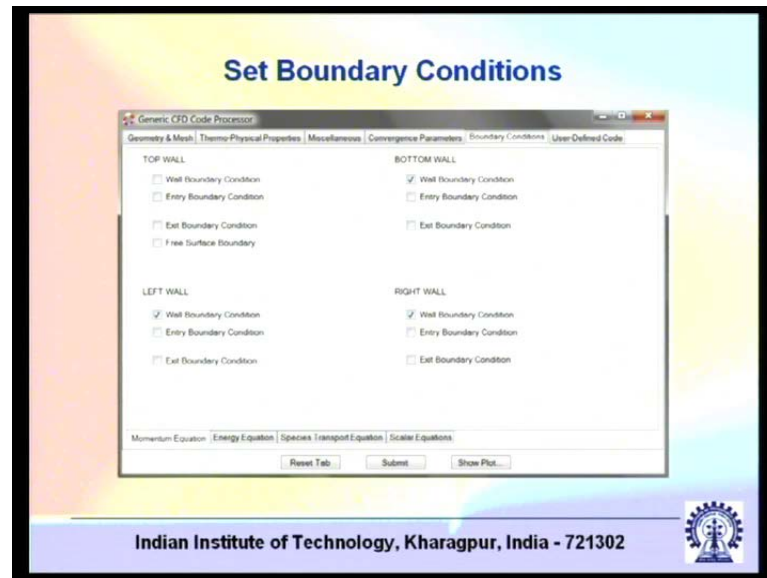
Then, you specify the density, viscosity, set initialization and time stepping parameters, so that time step and the final time, maximum iterations per time step.

(Refer Slide Time: 33:51)



Then, the relaxation parameters and relative error tolerances.

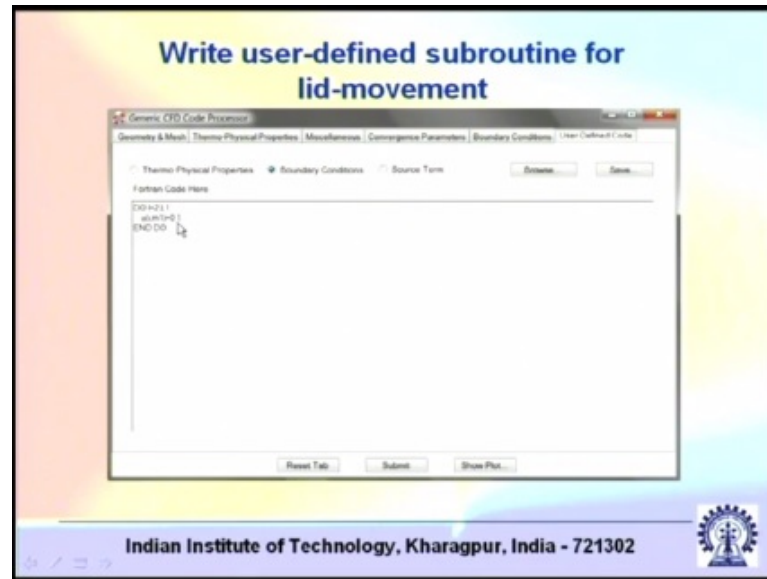
(Refer Slide Time: 33:55)



And then, set the boundary conditions – top wall, bottom wall, left wall and right wall. So, you can have wall boundary condition. There are certain common types of boundary conditions like entry boundary condition – in some codes, it is called as inflow boundary condition. Exit type of boundary condition – in some codes it is called as outflow boundary condition. And, wall type of boundary condition and other types of boundary conditions, which are not falling in these categories. Like for example, interfacial boundary conditions. Now, these boundary conditions may discuss in the context of fluid flow. So, the velocity boundary conditions. Entry boundary condition is typically where you specify the velocity profile, specify the velocity. The outflow boundary condition or the exit boundary condition – where you specify that along the stream, the gradient of the variable is equal to 0. So, for example, if it is a fully developed flow, the outflow boundary condition is along x ; there is no further change in u . So, $\frac{du}{dx}$ equal to 0.

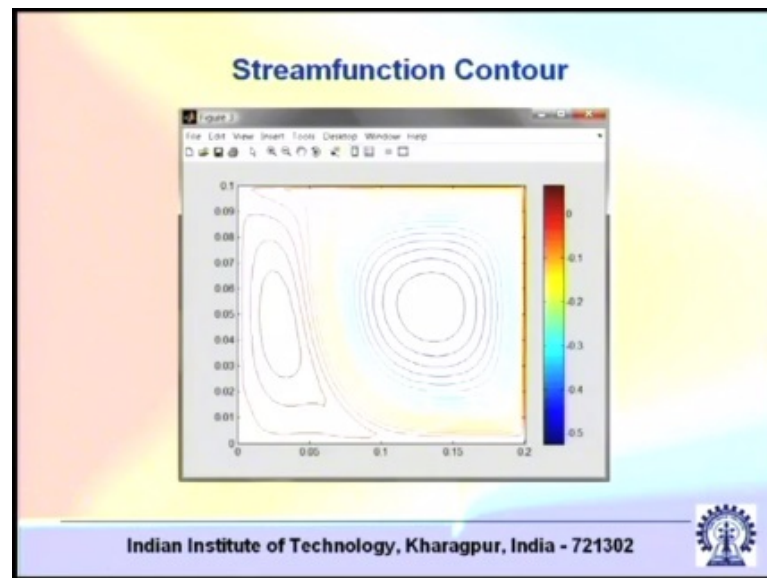
Here, you can see that the top wall is moving towards the right in the lid-driven cavity example. So, it is neither wall nor entry nor exit nor free surface. So, we have to write a user-defined function for it. But, for the bottom wall, left wall and right wall, it is a wall boundary condition. So, by wall, it implicitly assumes it as no slip, no penetration. So, if you want to implement a slip, then you have to write a user-defined function.

(Refer Slide Time: 35:26)



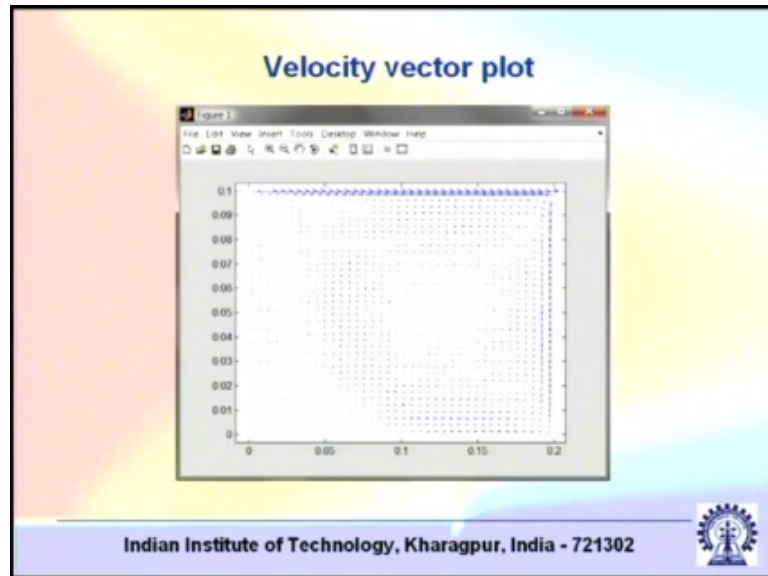
If you want to implement a penetration through a porous medium, you have to write a user-defined function. The user-defined function for the lid-movement – you can see that you are writing $u_i = 0$ is the number of grid points along y . So, this is like n_y equal to 0.1. So, you are specifying the u at the top; that is the meaning.

(Refer Slide Time: 35:47)



Usually, for fluid flow problems, you get results in terms of either streamlines or the contours of the stream function.

(Refer Slide Time: 35:57)



Or, the velocity vector plots.

(Refer Slide Time: 36:01)

Natural Convection Inside a Cavity

Problem Definition

Thermo-physical Properties

ρ (kg/m ³)	997
μ (Pa.s)	8.55×10^{-4}
β	0.0002761
k (W/mK)	0.62
c_p (J/Kg K)	4179

Initialization Parameters

u_0	0
v_0	0
t_0	23

Time Stepping Method

$\Delta t = 10$
 $t_{total} = 500$

$u = 0, v = 0$
 $\partial T / \partial y = 0$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

$$\frac{\partial \rho u}{\partial x} + \nabla (\rho \mathbf{u} \mathbf{u}) = \nabla (\mu \nabla u) - \frac{\partial p}{\partial x}$$

$$\frac{\partial \rho v}{\partial x} + \nabla (\rho \mathbf{u} \mathbf{v}) = \nabla (\mu \nabla v) - \frac{\partial p}{\partial y}$$

$$\frac{\partial \rho T}{\partial t} + \nabla (\rho \mathbf{u} T) = \nabla \left(\frac{k}{c_p} \nabla T \right)$$

$u = 0, v = 0$
 $\partial T / \partial y = 0$

0.22 m

Autosave Frequency
- Per 50 time steps

Relative Error Tolerance

u mom	1×10^{-5}
v mom	1×10^{-5}
energy	1×10^{-5}

Relaxation Parameters

u momentum	0.5
v momentum	0.5
Pressure correction	0.5
energy	0.5

Indian Institute of Technology, Kharagpur, India - 721302

Now, let us go through the second example, where heat transfer is also involved. So, it is a natural convection inside rectangular cavity. So, here the situation is that you have the top and bottom walls as insulated; whereas, the heat flux is supplied at the left wall and at the right wall with the magnitude of 500. So, if you consider Si units for example, it is 500 watt per meter square. Here the energy equation and the momentum equation are coupled. So, when you have natural convection in a cavity... Here in this particular slide

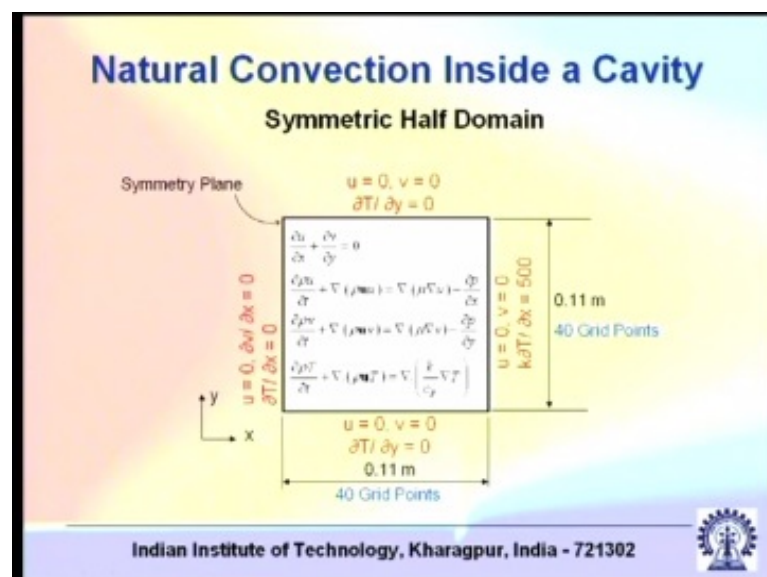
we have not purposefully kept one particular term in which there is a source term. Can you tell in which equation there will be a source term?

Y-momentum (())

In a y-momentum equation. So, in a y-momentum equation, there will be an additional source term, which you have to generate yourself. See the code does not know it. The code just knows that these are the generic equations that you need to solve. So, for the y-momentum equation, it will be $\rho g \beta (T - T_{ref})$. That source term will be there assuming that the Boussinesq approximation is valid. In this generic form, what it is written here, these are not necessarily the full forms of the equation. The full form of the equation will depend on what extra source terms that you need to use. And, if you need to use that, if it is already accommodated by the user interface, then you can just click that button; otherwise, you have to write a user-defined function.

What are the properties that you will require here? The properties for the momentum equation, that is, density and viscosity; and, the properties for the source term that will require, that is, beta; and, the properties for the energy equation k and c_p . So, these properties you need to specify. The initialization parameters u , v and T – these you initialize. Then, the time step again you use a time step of say 10 for example, and you set the relative error tolerances and relaxation parameters. Now, you also have the energy equation.

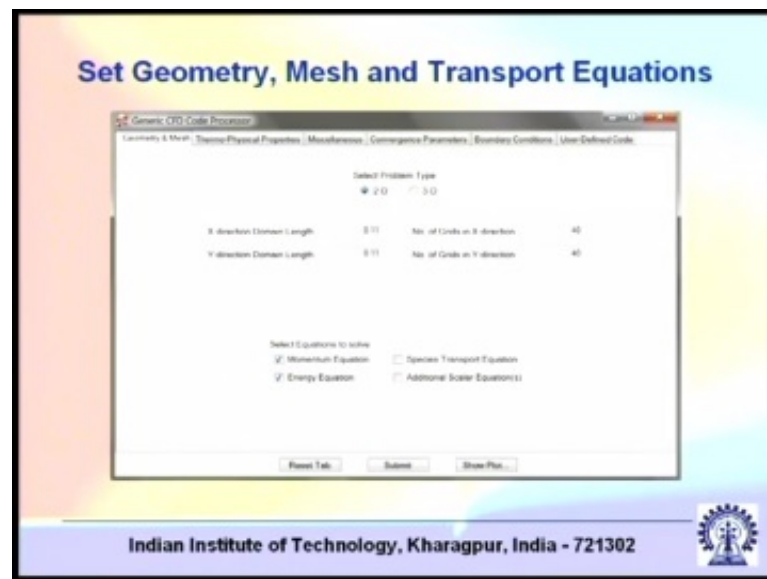
(Refer Slide Time: 38:16)



Now, let us try to solve the problem by considering the symmetry of the problem. See the problem is symmetric with respect to the centre line. So, if you pass an axis through the centre line, then the left-hand side and the right-hand side is exactly symmetry. So, you can solve half of the domain to save your computational efforts. So, what you do, you consider a symmetric half domain with... So, this is the centre line; this is the symmetric plane on the centre line of the rectangular cavity. So, you have to specify the boundary condition now at the centre line.

What is the boundary condition on u ? u equal to 0. Why u equal to 0? Let us come to the previous slide, if you consider the centre line. Now, if you consider some flow to right of the centre line, because of symmetry, the same flow will go to the left of the centre line also, so that the net will be 0. So, at the centre line, u should be equal to 0. And, because of symmetry, you have $\frac{\partial v}{\partial x}$ also equal to 0. And, $\frac{\partial T}{\partial x}$ equal to 0. So, it is very important to ascertain what will be the types of boundary conditions. And then, let us try to implement this problem.

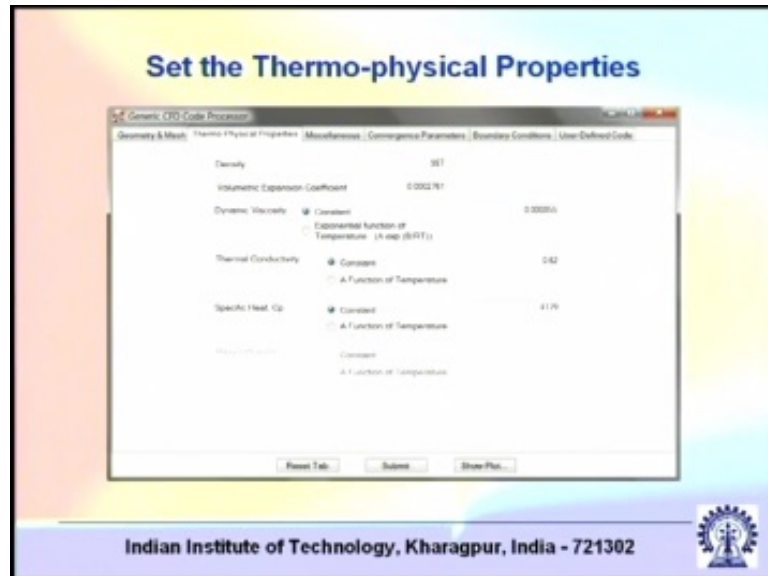
(Refer Slide Time: 39:42)



Set geometry, mesh and transport equation first is a 2D problem. So, we have selected 2D x-direction domain length, y-direction domain length. See always use consistent units. You do not expect that any unit conversion will be done in internal to the code. So, whatever unit you are using, you will be just be giving numbers. So, you have to make

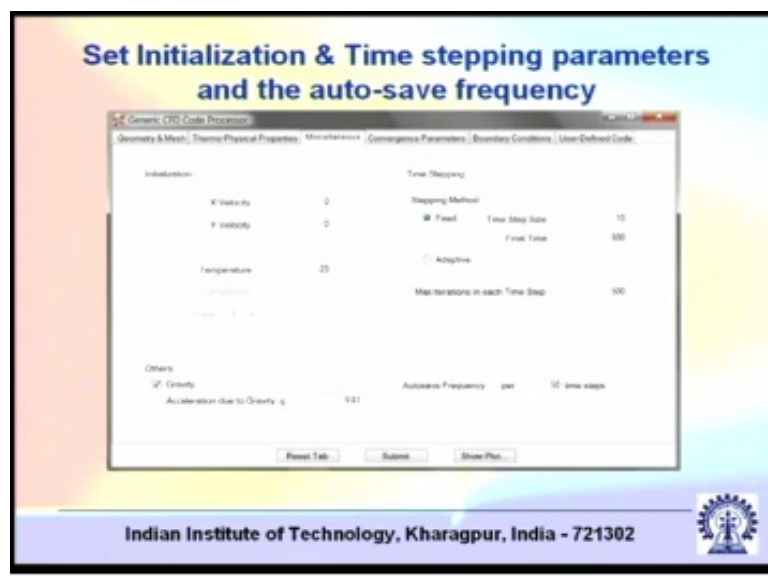
sure that you are using consistent units. Then, specify number of grids along x-direction and number of grids along y-direction.

(Refer Slide Time: 40:14)



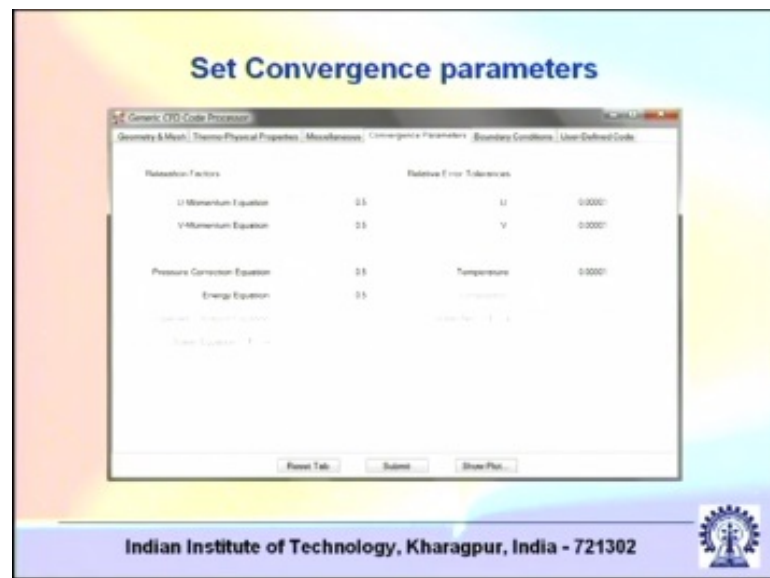
Then, set the thermo-physical properties. So, dynamic viscosity, thermal conductivity, specific heat – all these are constants. So, if these were functions of temperature, then we have already glanced through an example, where you can specify those as functions of temperatures also.

(Refer Slide Time: 40:33)



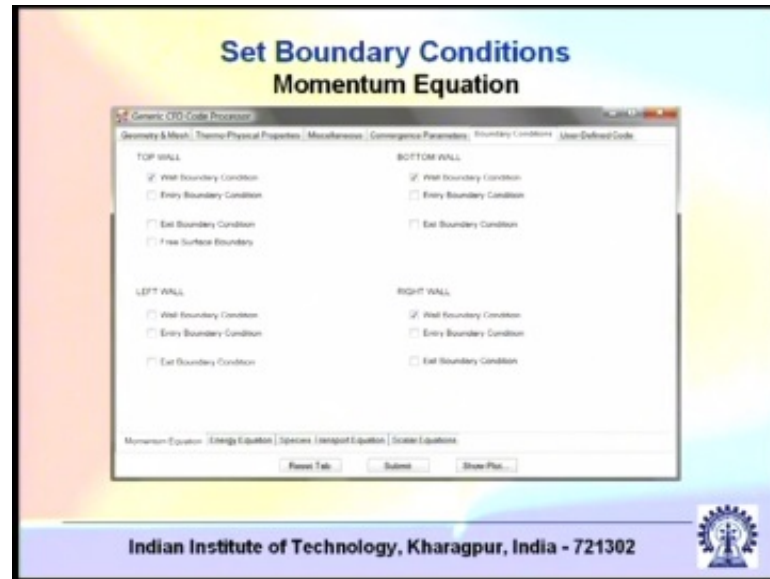
Then, the initializations; so, initialize the velocity and temperature. You may use a fixed time step for this problem; this is a relatively simple problem. So, one need not bother so much about variable time steps. Then, others – you can see that you do not have to write a u d f for gravity along y, because it is already accommodated within the GUI. So, you have a gravity option; you just click the gravity option here and give the value of acceleration due to gravity. So, remember, y is positive upwards. So, to make sure that you are implementing the gravity along negative y, you give the g as minus 9.81. But, it is not always necessary. You have to understand what is there inside the code. Maybe inside the code, gravity is already implemented along negative y. That is why do not try to use a code without trying to at least understand superficially what is there inside it; otherwise, you can make mistakes by considering some values which are not consistent with what has been put inside the code.

(Refer Slide Time: 41:40)



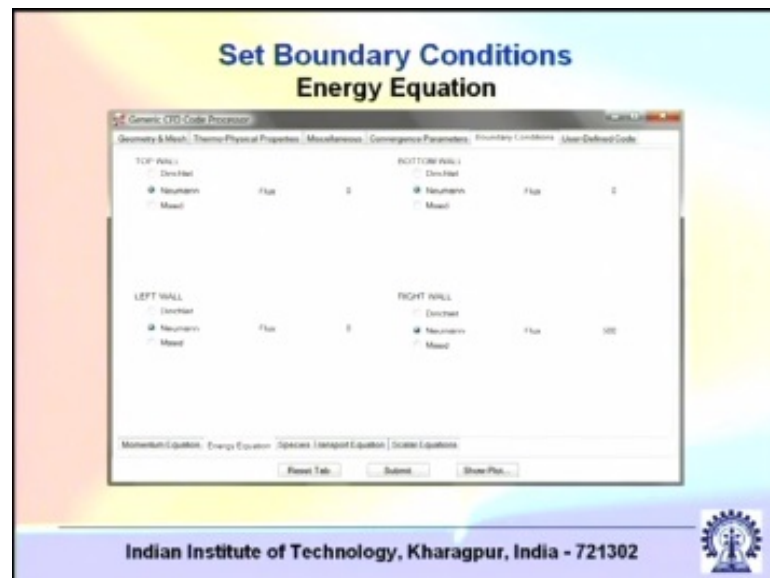
Then, set convergence parameters – u-momentum, v-momentum equation; you set the relative error tolerances. Also, you do the same thing for the pressure correction and the energy equation.

(Refer Slide Time: 41:51)



Set the boundary conditions – the top wall is the wall boundary condition; bottom wall is the wall boundary condition; left wall is neither wall nor entry nor exit; and, the right wall is the wall boundary condition.

(Refer Slide Time: 42:11)



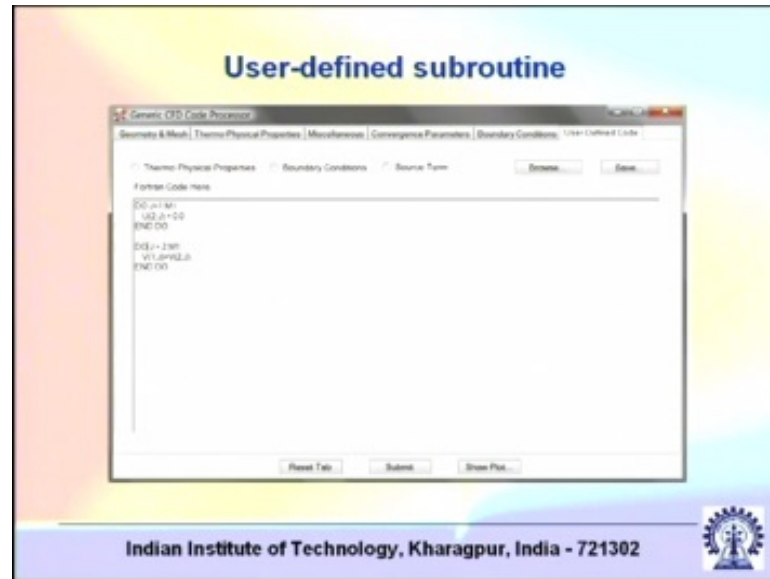
The previous slide was for the momentum equation. Then, you come to energy equation. For a generic equation, we can have the boundary conditions. What are the different types? The Dirichlet boundary condition, where you specify the value; the Neumann boundary condition, where you specify the gradient; and, the mixed type of boundary

condition, where you specify the value as a function of the gradient; of course, you can also have a periodic type of boundary condition. So, here in this particular example code, we are having possibilities of Dirichlet, Neumann and mixed. So, at the top wall, it is a Neumann boundary condition; it is insulated; that means, it is a heat flux boundary condition, where the flux is equal to 0. So, you have flux equal to 0. Bottom wall also, flux equal to 0. Left wall is Neumann boundary condition, because of what? Because of symmetry. It is not actually a wall. Left wall has come out because of the standard GUI; it is written as wall, but it is actually the axis of symmetry for this particular problem. So, at the axis of symmetry, across the axis of symmetry, you have flux equal to 0. And, the right wall – you have Neumann boundary condition, where the flux is specified. So, you can see that here at all the boundaries, Neumann boundary conditions are specified till you will... So, we have come across some examples where we had fallen in troubles, where in all boundaries, Neumann boundary conditions are specified. In this particular problem, we will not fall in trouble. Why?

Initialization

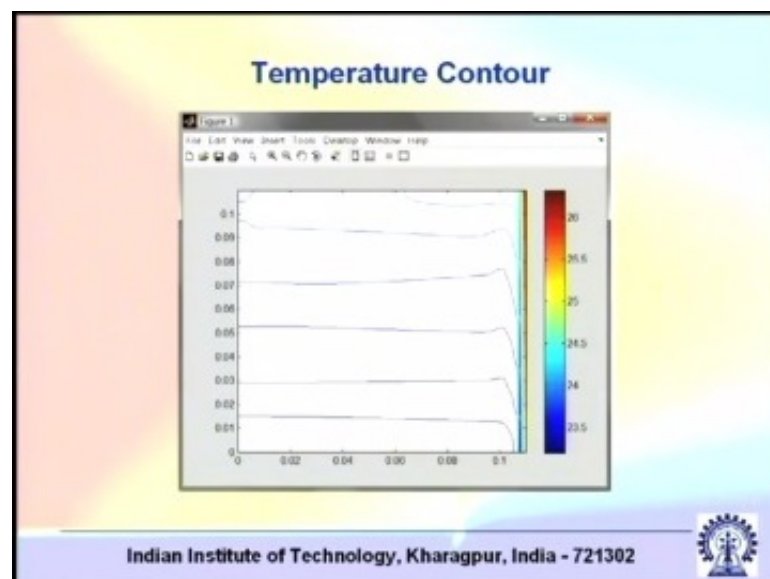
So, here the problem is a time dependent problem and unsteady problem. An unsteady problem can evolve from an initial condition. So, at time equal to 0, you have a particular value of temperature and that is a unique value. It is not an artificial iteration-driven value. So, if it is an iterative solution, then even if it is a steady state problem, you could start with initial guess. But, that is a guess, an artificial initialization. Here you have a real initialization, where at time equal to 0, you have a particular value of temperature. So, based on that level, it will subsequently evolve and it will not create any problem for having a Neumann boundary condition at all the boundaries. So, do not confuse those examples with this particular example that we are considering.

(Refer Slide Time: 44:37)



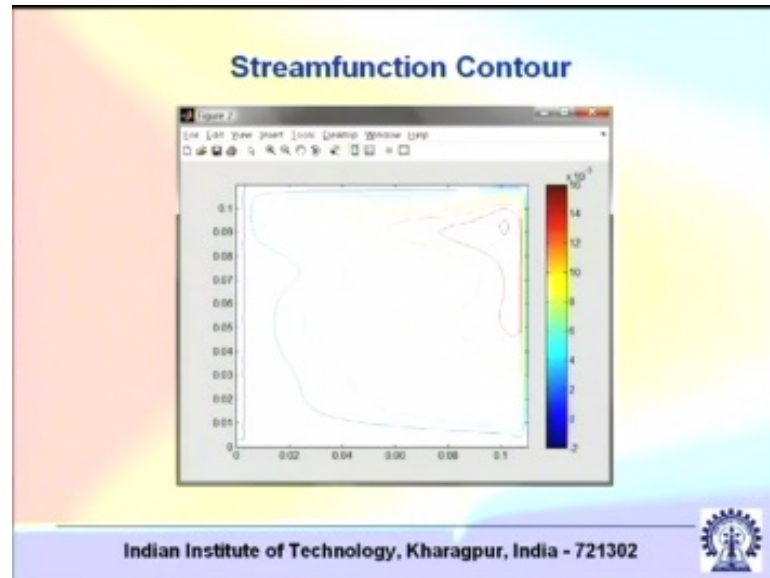
Now, user-defined subroutine that is there; what you do, you consider the left boundary. See at the left boundary, the velocity was neither inflow nor outflow nor wall. So, you have to write a user-defined function for... First is u; u equal to 0. So, you write u equal to 0 there. And, $\frac{\partial v}{\partial x}$ equal to 0; that means, v_1 equal to v_2 . Remember, for v, you are starting index with 1, because it is not staggered along... For y-momentum, it is staggered along y. So, v is not staggered along x. So, that is why the indices for v are 1 and 2. These are subtle things, but I am repeating these over and again, because these help you to write the good user-defined functions.

(Refer Slide Time: 45:26)



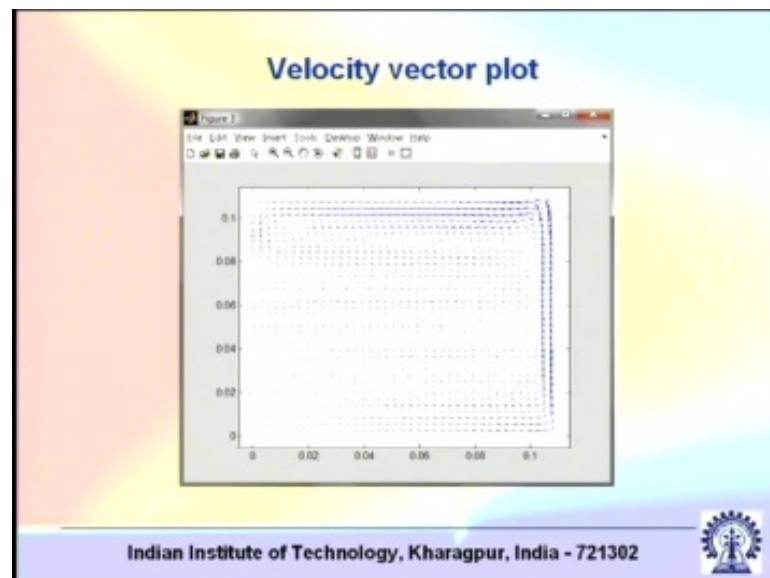
Now, you can see that this is a contour in the half domain. So, because of the heat transfer to the system, there is a natural convection and you can see that there are strong temperature gradients along the wall; and, you have different stratified layers of fluid.

(Refer Slide Time: 45:47)



And, this is a stream function contour or a velocity representation.

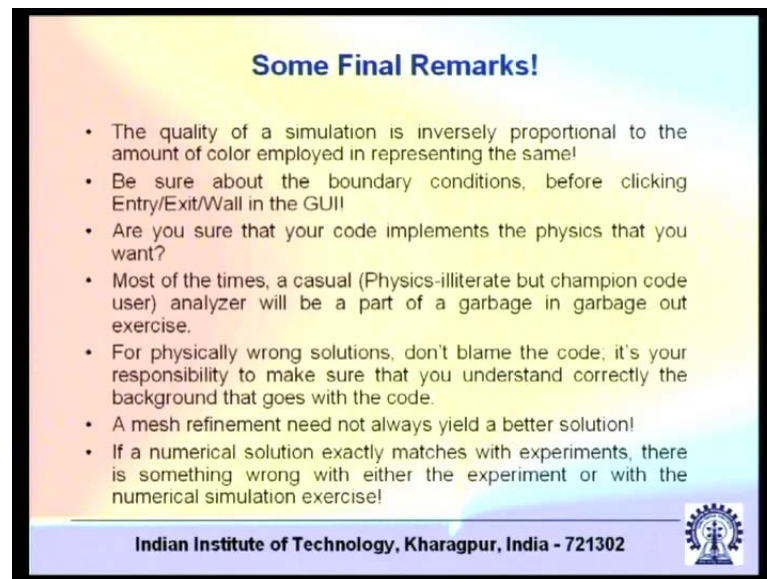
(Refer Slide Time: 45:56)



For representation of velocity, you either you streamlines or you use velocity vectors. So, how do you derive these from your calculations? The velocities are directly derived. You may develop your own GUI or you may use some standard GUI or standard graphical

softwares. So, there are some commercially available softwares also; and, there are some common academic user type of software also; that you can use. So, for example, this is a plot shown using matlab as a GUI interface. So, what you are doing here is you are using a matlab inbuilt function called as quiver. So, if you have u and v as two velocity components, quiver u comma v generates the velocity vector plots in matlab. So, this is just a generic example, but you can use different types of graphical user interface. So, for contour plots in matlab, you can use a similar function called as contour.

(Refer Slide Time: 47:11)



Some Final Remarks!

- The quality of a simulation is inversely proportional to the amount of color employed in representing the same!
- Be sure about the boundary conditions, before clicking Entry/Exit/Wall in the GUI!
- Are you sure that your code implements the physics that you want?
- Most of the times, a casual (Physics-illiterate but champion code user) analyzer will be a part of a garbage in garbage out exercise.
- For physically wrong solutions, don't blame the code, it's your responsibility to make sure that you understand correctly the background that goes with the code.
- A mesh refinement need not always yield a better solution!
- If a numerical solution exactly matches with experiments, there is something wrong with either the experiment or with the numerical simulation exercise!

Indian Institute of Technology, Kharagpur, India - 721302

Now, before winding up, let us come to some final remarks. Now, these remarks are important; I would expect that you would take some of these remarks in a light mood, but some of the remarks are not exactly in a light mood. When I was a student, I was studying in a class and this is a particular statement that one of the professors made to us – the quality of a simulation in CFD is inversely proportional to the amount of the colour in representing the same. Now, the reason is that these days we have many available softwares, where you can give say different types of inputs, just have a button click on certain things and you will generate excellent nice plots. Once you generate these excellent nice plots, then what you get out of these? You do not know whether you are getting the right solution or wrong solution or what, but plots are very colourful. Because these plots are very colourful, they are very much presentable; and, because they are very much presentable, you get tempted to present them in whatever presentations that you are going for. And, that is why CFD to many communities is known as colourful fluid

dynamics; not computational fluid dynamics, because the results are filled up with coloured plots of contours and whatever.

Sometimes it is also told as colour for directors, because in some companies, the directors may become very much fascinated; they may not look into the details of the CFD, because they do not have the time. So, they will look into the results. And, if you can impress your director, you know that you can get a lot of mileage in your company. So, it is also called as colour for directors. But, jokes apart, if you take it very seriously, many times we are not very much careful about the authenticity of the results once we get nice plots. And, this is something which is very deterring. One should be very careful about what? One should be very much careful about what is the physics that has gone into the CFD code.

Now, once you considered the physics, one of the important considerations that comes is the boundary condition. Be sure about the boundary conditions before clicking entry/exit/wall in the GUI or inflow/outflow wall. This is very tempting, because what we see very commonly that you click for example, outflow boundary condition where it is not exactly outflow. Let us give an example. Let us say you have a thermally fully developed flow. So, if you have thermally fully developed flow along x , then it is not $\frac{\partial T}{\partial x} = 0$ at the outlet, but $\frac{\partial \theta}{\partial x} = 0$; where θ is $T - T_{\text{wall}}$ by $T_{\text{bulk mean}} - T_{\text{wall}}$. So, if you are solving for temperature, you will be most commonly tempted to click the outflow boundary condition, because it is available with the GUI. But, that will be totally wrong if you are implementing a thermally fully developed flow. You have to somehow implement $\frac{\partial \theta}{\partial x} = 0$. So, if you are solving for θ as a variable and then implementing the outflow boundary condition, fine; but, not if T as a variable. So, you have to be very careful whether the boundary condition that you are implementing is representing the right physics.

Not only the boundary condition, in general, are you sure that your code implements the physics that you want? This is very important, because all problems do not have the same physics. Say you are interested to solve for a verified gas dynamics problem. So, if you are interested to solve a verified gas dynamics problem, then in that case, continuum equations may not be valid. And, in that case, maybe you may require molecular simulation or microscopic simulation; similar other types of simulations are there. Now, there you are forcefully trying to use the standard Navier-Stokes equation without any

alteration in the boundary condition; and, you are expecting that your code will give you the correct result. That is not what you are going to look for. So, you must make sure that your code implements the physics that you want. That is very important. If you are unsure about that, first make sure that what are the governing equations; is it possible to cast those governing equations in a conservative form? If it is not possible to cast those governing equations in a conservative form, then this CFD code is not for that particular problem.

Everything is not for everything. So, it is not that you have a CFD code, which you can use for solving any problem; it has to be a... So, CFD code fundamentally is what? A CFD code fundamentally is a PDE solver. So, it is a coupled PDE solver so to say. So, if the PDEs are possible to cast in a general conservative form, then only the CFD code is possible to be implemented for the particular problem that you are implementing. If you are having equations which do not fall into that category, simply do not use that code. Try to use the different code or develop a code or whatever, but do not use it.

Most of the times, a casual – physics illiterate, but champion code user – so, these days there are many such people whom we see like even in research labs or in the companies, several places. There are people who can use the code very nicely, but he is not at all a person who is good in physics in solving such a problems. Most of the times, such an analyser will be part of garbage-in, garbage-out exercise, because such an analyser will not be careful about the physics; such an analyser will be careful only about how to just click the mouse button for giving some inputs and getting some outputs without knowing that what the code is doing. So, it is very important not always to write a code by yourself, but at least to know very thoroughly what the code does. So, what are the strong and weak points?

So, if such a user gets physically wrong solutions, do not blame the code; it is your responsibility to make sure that you understand correctly the background that goes with the code. So, most of the times, when something has gone wrong, we blame the code. So, the code is a dumb individual; it is inanimate. So, it is very easy way to blame a code. You cannot blame an individual, but you can blame a code. So, for physically wrong solutions, do not blame the code; it is your responsibility to make sure that you understand correctly the background that goes with the code.

Then, there are certain issues about meshing. See theoretically, we always learnt that a mesh refinement will always give a better solution. But, practically, a mesh refinement to an unlimited extent may not always yield a better solution. The reason is that when you go for mesh refinement, you are sometimes dealing with small numbers, because the grids sizes are small. And, those small numbers can interfere with some of the calculations. And so, those may interfere with round of errors and so on. So, a mesh refinement need not always yield a better solution; in theory, yes; in practice, may be / may not be.

Now, the other thing is that when you have a solution, how do you validate it? CFD problems have benchmarks. So, there are some standard benchmark problems in the literature; you can validate your solution with the benchmark problems. Sometimes you validate your numerical solutions with experimental results. But, if a numerical solution exactly matches with experiments, there is something wrong with either the experiment or with the numerical solution. The reason is you cannot perfectly match an experimental condition, which is a reality with an idealized condition, where you do not know actually the boundary conditions; you say something isothermal that is not really isothermal. Then, you specify certain properties; you really do not know how those properties vary with temperature and other conditions. So, your reality experimental condition is not mimicked by the idealized modelling conditions. So, all in all, these are some of the practical remarks that we should keep in mind. It is very important to use a CFD code, but it is also important to keep in mind that what are the cautions, what are the important points that you should remember while implementing a CFD code. We stop here today with that.

Thank you very much.