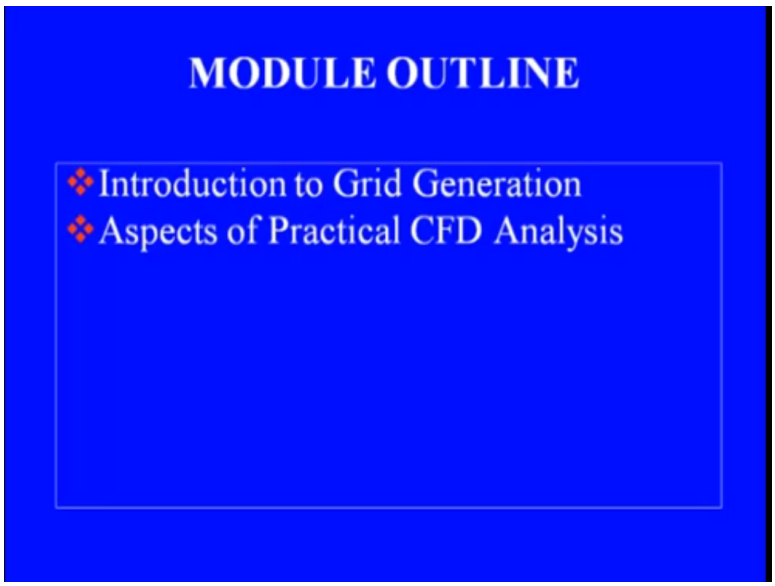
Computational Fluid Dynamics
Dr. Krishna M. Singh
Department of Mechanical and Industrial Engineering
Indian Institute of Technology - Roorkee

Lecture - 42
Introduction to Grid Generation

Welcome to module 10 on grid generation and aspects of real life CFD analysis. This module would be a last one in this introductory course and we would focus very briefly on 2 topics. The first one is introduction to grid generation, which is an essential part of numerical simulation procedure and the next part would be the aspects of practical CFD analysis.

(Refer Slide Time: 01:04)

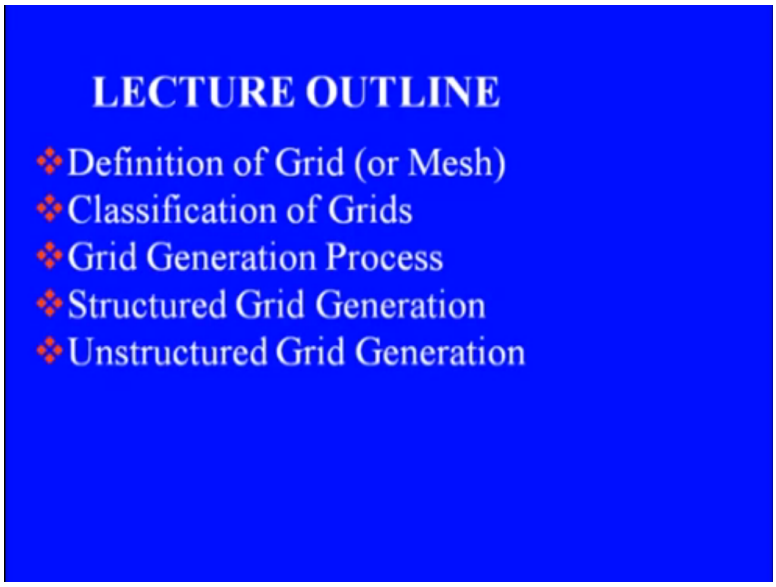


What are the things which we must keep in mind while carrying out a real life CFD analysis for a complex geometry problems? So these are 2 aspects which we will deal with. I will briefly touch upon in this concluding module of our course on computational fluid dynamics. So we have broken this module into 2 lectures.

The first one would cover grid generation and the second one we will take up the aspects of validation and verifications, which are essential for any industrial or research CFD analysis. So let us have a look at first lecture in this module, introduction to grid generation. We will basically cover briefly the types of grids, which are there which are normally used in CFD analysis and generation process in brief.

And we would briefly discuss about advanced material, which you can refer to, to develop expertise in grid generation. This being an introductory course, we have no intention of introducing the actual numerical techniques which are used in CFD analysis. We would cover only the basic theoretical aspects and thus where we would leave this particular lecture. So now let us have a look at the lecture outline.

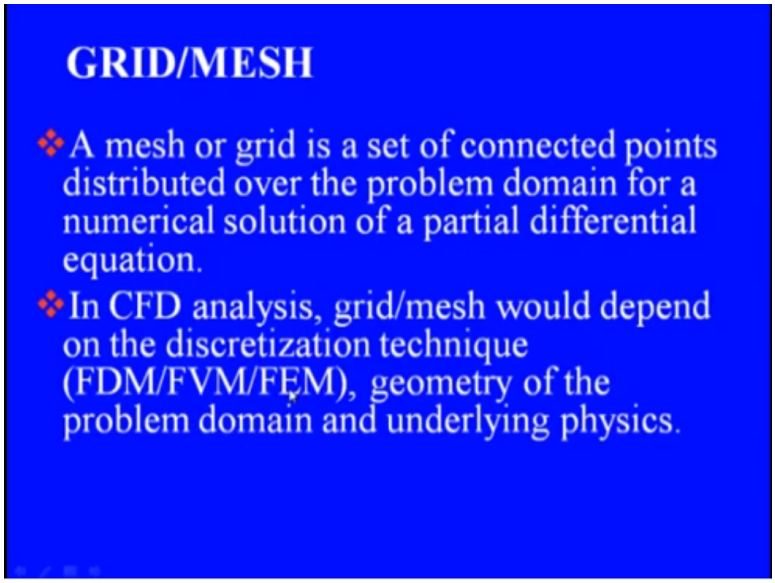
(Refer Slide Time: 02:37)



We will first formally look at what we call a grid that is definition of grid or mesh. Then we would briefly touch upon the classification of grids. Then overall grid generation process, what are the steps involved in grid generation? And then we will look at 2 types of grid generation processes, a structured grid generation and unstructured grid generation. Now let us see how do we define a grid or a mesh?

Both of these terms are interchangeably used in CFD literature. Grid is more common in numerical techniques dealing with finite difference or finite volume applications and mesh is the term, which is most popular in finite element based CFD analysis. So how do you define? What is a grid or a mesh?

(Refer Slide Time: 03:38)

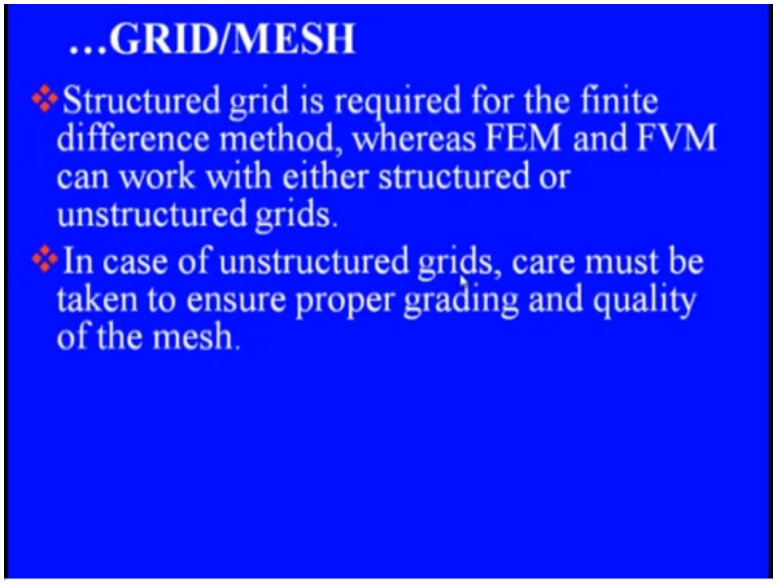


When we are solving continuum problem, we would solve partial differential equation, which is defined over a finite continuum domain. Now that continuum domain contains infinite number of points. Grid is essentially the discretization of this continuum domain for numerical simulation. Say about discretize that is to say we would approximately represent our continuum domain by a set of discrete points.

Now in mesh based methods for instance finite difference, finite volume or finite element techniques, these discrete points are not arbitrary, they are connected in a certain way. So that is what we call a mesh. So mesh or grid is a set of connected points distributed over the problem domain for a numerical solution of a partial differential equation and in CFD analysis which type of grid or mesh we would use that would depend on the discretizing technique which we employ.

That is whether we use finite difference method or finite volume method or finite element method so that is one aspect, which will control the choice of the grid. The second would be the geometry of the problem domain and specifically the grid gradation that would also be influenced by the underlying physics of the problem. The gradients of the flow variables would be one of the essential features, which will influence our grid design.

(Refer Slide Time: 05:17)



Now we would use a structured grid for finite difference method, most often we go for a Cartesian structured grid in finite difference analysis. Other option for finite differences could be that we could go for body-fitted grid in which case our computational domain where finite difference method is applied that is still a Cartesian mesh, but that is mapped to our complicated problem geometry.

And that mapping gives us what we call a body-fitted structured grid. So that is why we would use structured grid for finite difference method. Finite element and finite volume methods they can work with either structured or unstructured grids. We will have a look at the formal definition about these grid types little later in the lecture. Now in finite volume or finite element analysis, when we using unstructured grids we have to take few precautions.

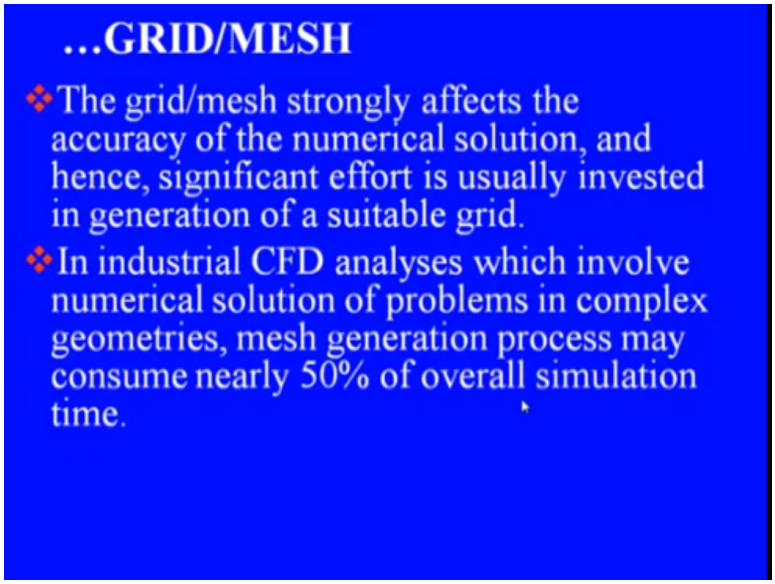
So k must be taken to ensure proper grading of the grid by grading we mean the size distribution of the mesh elements in different regions of the problem domain. We would have to ensure that we have finer meshes close to the problem boundaries or solid boundaries where we expect very large gradients of the flow variables and coarser grids away from the boundaries.

Similarly, we have also got to make sure that what we call finite elements or finite volumes they have got proper geometric structure and that leads to a term, which is we call quality of the mesh. The quality of the mesh is defined in terms of the angles, which are contained in the elements and the size gradation in the mesh. So both of these features will affect our overall quality of the mesh.

And unless we have got a good quality mesh, the numerical errors in the discretization process would spoil our numerical solution. In fact, even convergence maybe badly affected if we do not have a properly graded and good quality mesh. In addition, specifically in 3-dimensional situations we have got to make sure that all parts of the interior of the problem domain are contained in an element.

That is to say the creation of elements has not left what we call (()) (07:57) in our continuum domain because then that would lead to severe problems in numerical simulation. It might invalidate our entire numerical simulation itself.

(Refer Slide Time: 08:10)

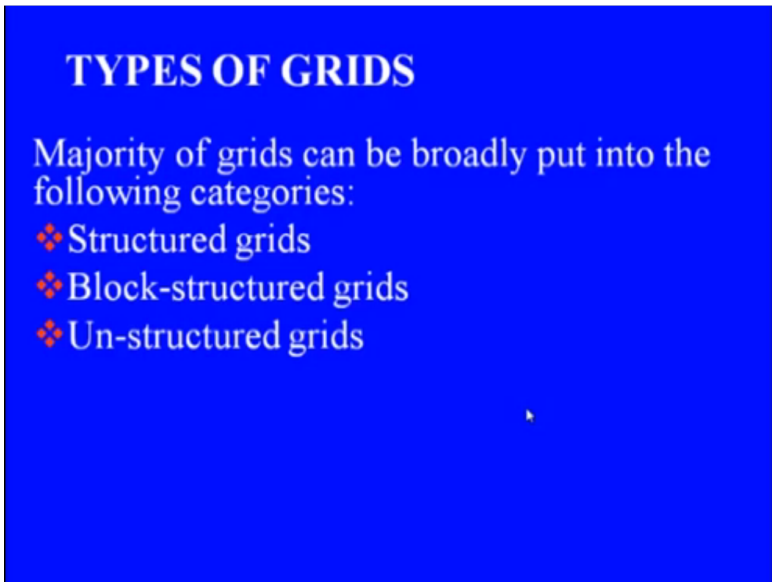


So this grid or mesh, which we design that strongly affects the accuracy of the numerical solution and hence a significant effort is usually invested in generation of a suitable grid. So much so that if you are dealing with industrial CFD problems wherein we would encounter complex geometries. Since such cases mesh generation process may consume nearly 50% of the overall simulation time.

Sometimes it may so happen that more than 50% time may be taken by the geometry modeling and a suitable good quality mesh generation process. So there is nothing surprising because that is the starting point that is the foundation of the CFD analysis. We have to generate a proper computer representation of the complex geometry of the problem and we have to come up with a good quality mesh.

Only then the numerical solution, which we would get, can with any confidence represent an approximation of the physics of the problem. Now let us have a look at the basic classification of the grid. We would have a look at only broad classifications. There are many other classifications or sub classifications for each type so we are not going to those details in this lecture.

(Refer Slide Time: 09:52)



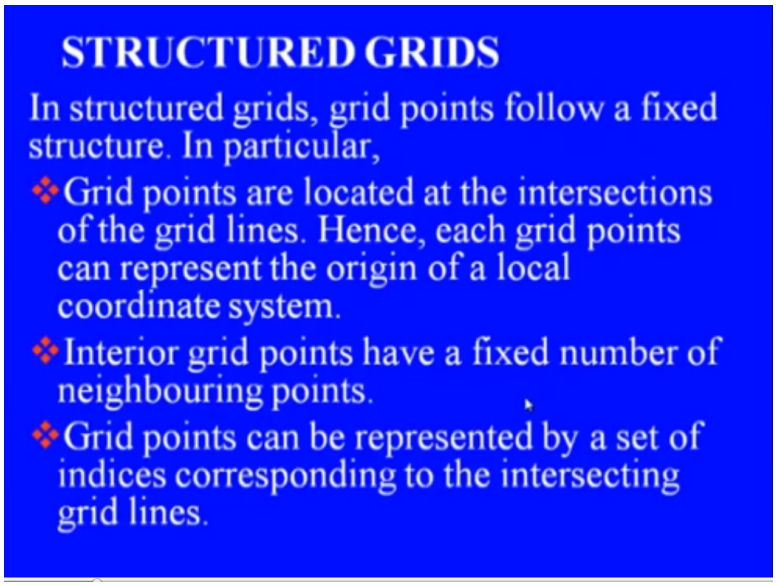
We would have a look at what we call the broader or top level classification of the grids. So majority of the grids can be broadly put into the following 3 categories, the first one is what we call structured grid where there is a specific structure, which assume post by the grid lines whose intersection represents our grid nodes and the next category is what we call block-structured grids.

Wherein we do not have a single structured grid yet over our problem domain, but we might have broken up a problem into a set of blocks, but over each block we have got a structured grid. Now the nature of the structured grid is gradation that might differ from one block to another and the last one is what we call unstructured grid wherein there is no fixed connectivity between the grid points.

There might be some interior grids, which might have say 4 neighbors in a 2-D problem or they might have more than 4 neighbors. So number of neighboring nodes, their orientation this would all be unstructured or unfixed that will not be precisely defined in the case of unstructured grids. Let us have a look at each of these type in bit more detail. So what do you mean by a structured grid?

In a structured grid, grid points follow a fixed structure. In particular, the grid points are located at the intersection of what we call grid lines. Hence each grid point can represent the origin of a local coordinate system. We will have a graphical representation of these concepts very soon okay and the interior grid points in a structured grid there are fixed number of neighboring points unlike unstructured grid where each interior grid point might have a differing number of neighboring points.

(Refer Slide Time: 11:53)

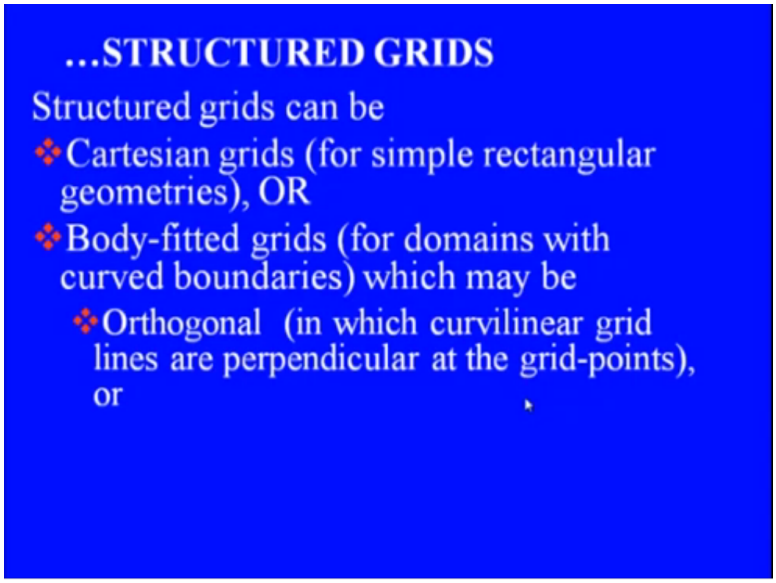


And the grid points can be represented by set of indices corresponding to intersecting grid lines. In fact, this aspect we have already discussed very briefly when we were dealing with finite difference discretization. We will revive these concepts once again and if you look at the geometrical nature of the structured grids, they contain of elements which can be mapped to a rectangle in 2 dimensions or a parallelepiped in 3 dimensions.

Now here we are talking about a generic structured grid, which could be Cartesian which could be body-fitted or which could be curvilinear. So irrespective of the type of structured grid or the nature of the structured grid, which we have got any grid element in 2-D can be mapped to the rectangle and a parallelepiped in 3 dimensions. This all grid lines are oriented regularly in either 2 or 3 directions.

So that coordinates transformation of curvilinear lines results in rectangle in 2-D and parallelepiped in 3-D.

(Refer Slide Time: 13:00)

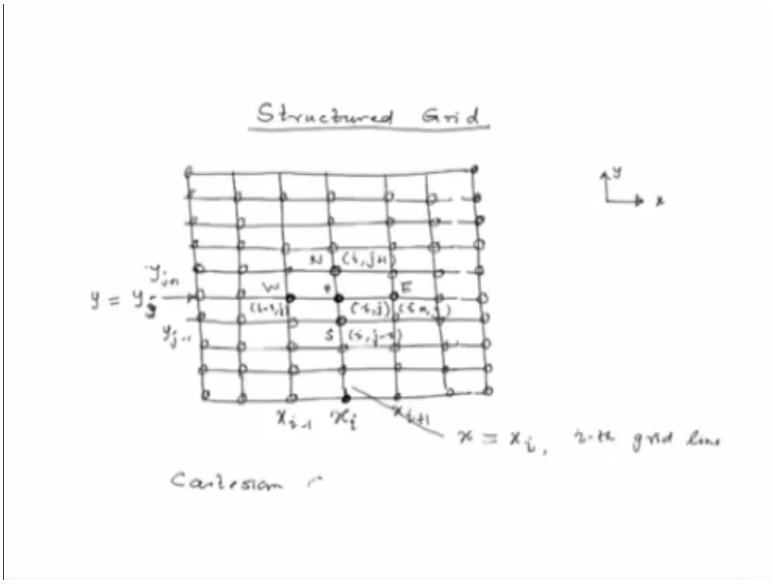


Now these grids could be Cartesian that is the simplest possible grid. Cartesian grids are preferred for simple rectangular geometries, but recently Cartesian grids are also used for arbitrary complex geometries in conjunction with immersed boundary techniques. We will have a brief look at the use of Cartesian grids for complex geometry problems in the next lecture.

Other one is what we call body-fitted grid for domains with curved boundaries. Now again body-fitted grids might have 2 types of grids, the first one is what we call orthogonal grids in which curvilinear grid points or grid lines are perpendicular at grid points that is to say they intersect each other orthogonally that is why this is called an orthogonal grid or other possibility is what we call non-orthogonal grid in which curvilinear grid lines intersect obliquely.

The angle between the grid lines is not 90 degree at the grid points. Now let us have a look at the geometric details of the structured grids in our board.

(Refer Slide Time: 14:08)



We will first illustrate on the concepts we just discussed using 2-D Cartesian grids. So in 2-D Cartesian grids let us say this is our x, y axis. Our grid lines would be parallel to these axis lines so let us draw a set of grid lines. These are what we call y grid lines, which are parallel to x axis. Similarly it is an intersecting set of grid lines, which are perpendicular to these y grid lines or what we call x grid lines.

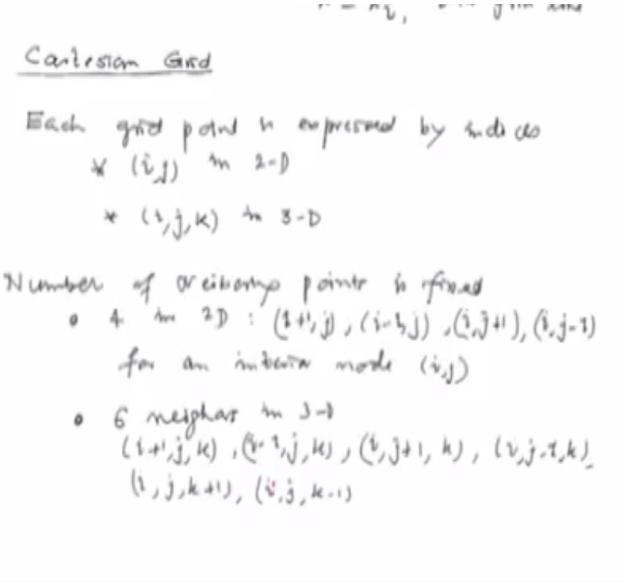
Now if we had a rectangular problem domain of course that is where the structured grid is most natural to use. Now each of these grid lines can be identified by an integer index, which will correspond to our x locations. For instance let us say a generic point here, this vertical grid line which corresponds to x=xi. So this particular grid line is what we call x=xi or ith grid line.

Similarly suppose this line has got the equation y=yj so we would use 2 different indices i for x direction and j for y direction. So then intersection points of these 2 grid lines that can be identified by 2 unique indices i, j in 2 dimensions. If we had a 3-dimensional problem we will have yet another grid line let us say z=zk grid line, which intersect at this point and the index of the point would be i, j, k.

Now if you look at this grid point it has got a fixed number of neighbors, 2 neighbors in the top and bottom and similarly 2 neighbors along x direction what we call east or west. So east neighbor whose grid indices are i+1, j and this is w this index is i-1, j and northern neighbor with an index ij+1 and southern neighbor ij-1. So whichever grid point we take it does not matter any of these internal grid points will have the same number of neighbors.

So these are all our grid points and you pick up any grid point, the number of neighbors is fixed. In 2-D, we will have 4 number of neighbors for each interior grid point. We have used the word interior specifically because if we are at the boundary point suppose we are at any of these boundary nodes, the number of neighbors might be less than 4. So depending on the location on the boundary we might have different number of neighbors for the boundary nodes.

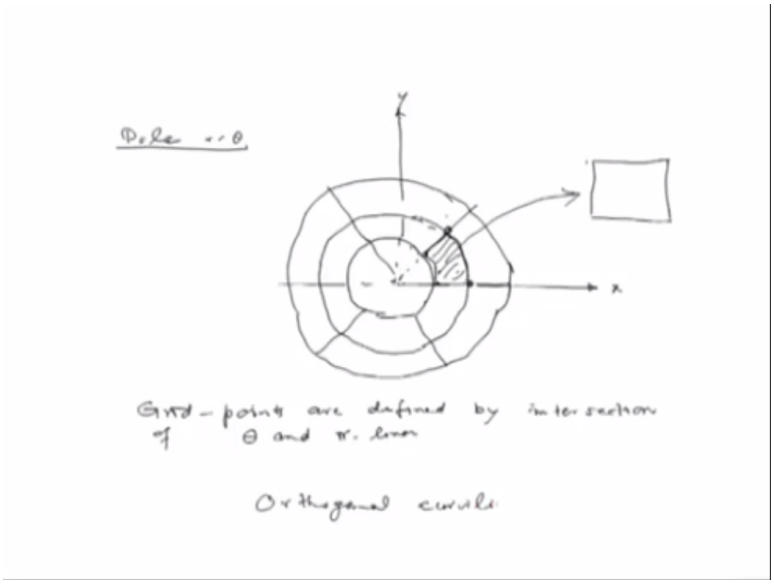
(Refer Slide Time: 18:50)



So that is why we talk only in terms of the interior grid points. So this is our typical Cartesian grid. So each grid point is expressed by indices i, j in 2-D or i, j, k in 3-D. Similarly, we have also seen a number of neighboring points is fixed it is 4 in 2-D, that is we have got i+1, j i-1, j ij+1 and ij-1. These are indices of the neighboring points for an interior node. Similarly, we can see the same thing happen similar set of indices or number of neighbors would be 6 in 3-D.

It is i+1j, k i-1j, k ij+1k ij-1k ijk+1 and ijk-1. Similarly, what we can see you pick up any interior node now that interior node can itself be thought of as if it is origin of a local coordinate system x prime y prime. So that is yet another fundamental feature of all the structured grids that any grid point can be taken as local origin of a coordinate system. Now the Cartesian grids they are by nature orthogonal.

(Refer Slide Time: 21:58)

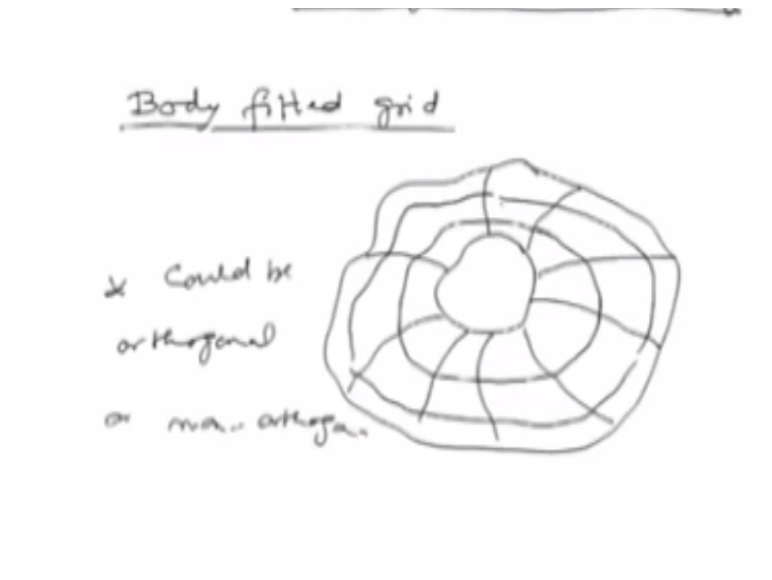


We can also have orthogonal grids in curvilinear plane as well. For instance if we have an annular domain, suppose this is our annular domain defined by 2 concentric circles. Now in this case, it does not make sense for us to define or use rectangular Cartesian grids. We might better use r theta coordinates x y instead we will use the polar coordinates and we can draw different theta lines.

So here our grid points would be defined as the intersections of the theta lines defined by intersection of theta and r lines. So r lines would be basically concentric circles and theta lines would be very real lines and each of these curvilinear elements, which we get here, suppose we take this curvilinear element it can be mapped to a rectangle corresponding to this r and theta values.

So once again here if you look at same sort of properties apply that this again an orthogonal mesh r and theta line, they intersects orthogonally. So this is one example of orthogonal curvilinear grid.

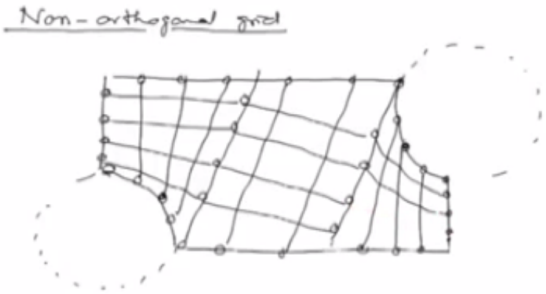
(Refer Slide Time: 24:26)



Now let us have a look at bit more complicated one what we call a generic body-fitted grid. Suppose we had fairly complicated region with a hole inside. Now what we can do is again we can identify our curvilinear coordinates and draw those curvilinear grid lines. Now these curvilinear grid lines they might follow our surfaces fairly closely that is why they are called body-fitted grids.

And depending on our mapping, we might have an orthogonal or non-orthogonal grid. So this (()) (25:29) we might have more number of curvilinear lines. So these grids could be orthogonal or non-orthogonal.

(Refer Slide Time: 25:56)



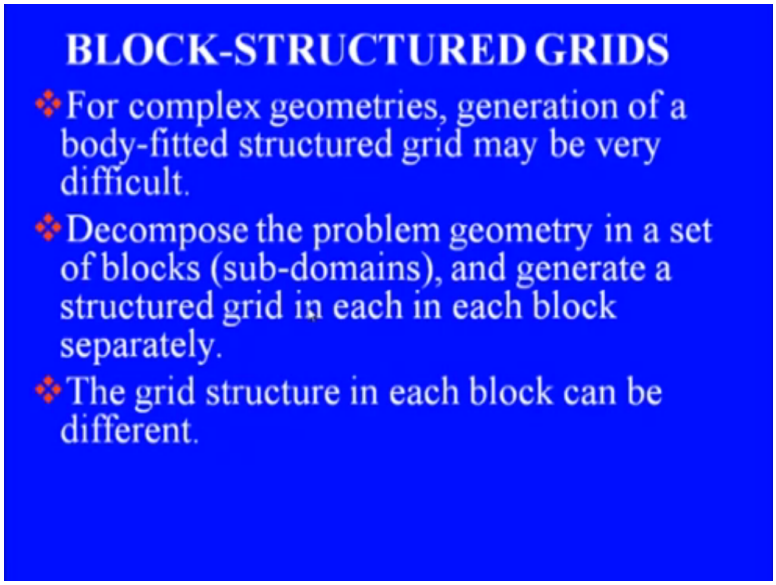
Let us have a look at one more non-orthogonal grid where non-orthogonality would be evident. Now this grid we would generate let us say in flow area between a set of tubes in

shell and tube heat exchanger. So this is quarter of one of the tubes and then other side we have got quarter of another tube. So this is how this area would look like so flow domain. Here what we have got is our one tube here and that tube here and because of symmetries lots of tubes present in the flow domain.

We would just concentrate in our flow analysis on this solid region. Now how do we generate a grid in this particular domain? We might divide it into 3 different sub regions and then we start off with our structured grid generation process as choose to. So this is our one set of grid lines. In structured grid on the corresponding faces, we must have equal number of grid points.

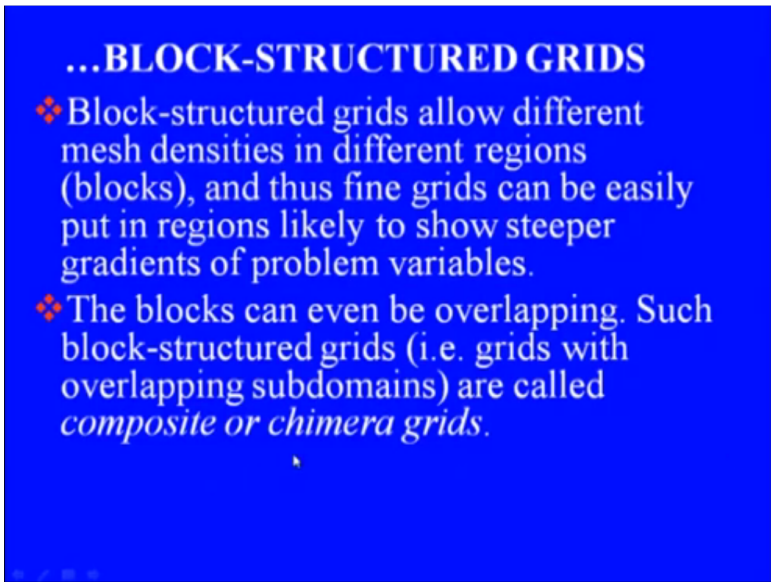
So you can clearly see here that the grid lines do not intersect orthogonally. Now let us have a look at the next category of what we call block-structured grids. Block-structured grids are used in complex geometries wherein generation of a body-fitted structured grid maybe very difficult over the entire problem domain. So for the sake of simplicity what do we do? That we decompose our problem domain into smaller blocks and over each block we can generate a structured grid.

(Refer Slide Time: 29:44)



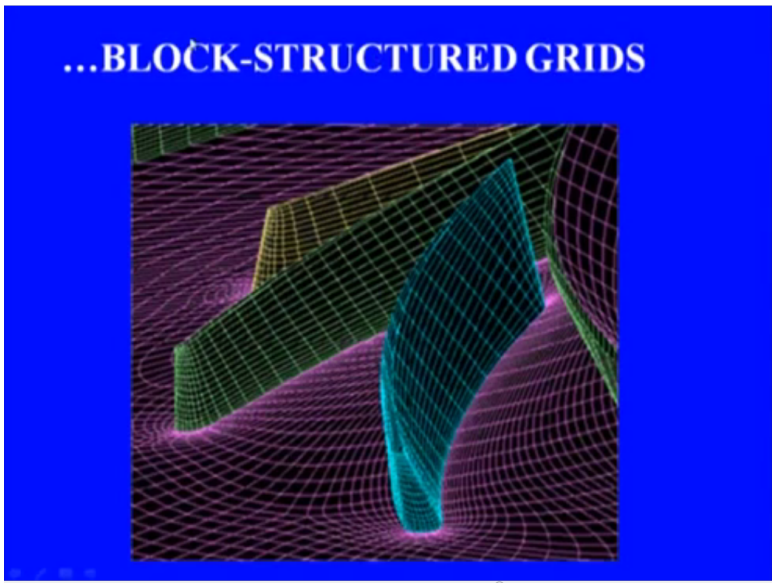
So decompose the problem geometry in a set of blocks which we call subdomains and generate a structured grid in each block separately and the grid structure in each block can be different. We might have a Cartesian grid in one block, curvilinear grid in another block. So there is no restriction whatsoever in the block-structured grids and we can have different mesh densities in different regions of space.

(Refer Slide Time: 30:12)



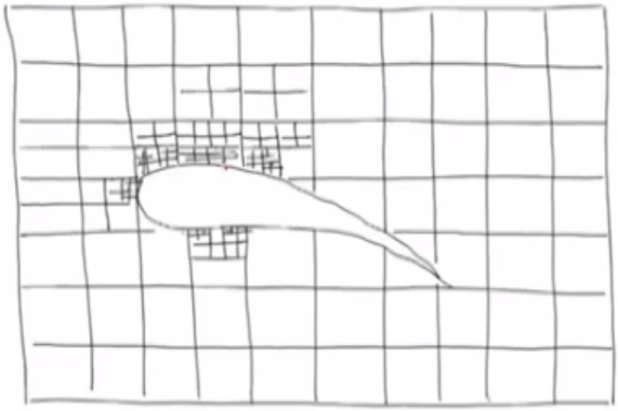
And thus fine grids can be easily put in regions likely to show steeper gradients of the problem variables and not just that the blocks can even be overlapping are such block-structured grids wherein we have got overlapping problem when they are called composite of chimera grids.

(Refer Slide Time: 30:35)



So now there is one typical example taken from rotating flow machinery. So there are different blades and each blade here has got a mesh of different density and the structured grid on each one was generated separately and similarly in the region which is filled here there is a different set of structured grids. You can see the layout of the grid also is different in different regions of the domain.

(Refer Slide Time: 31:20)



To take yet another example of these blocked structured grids let us have a look at how we would use Cartesian grids for flow over an airfoil. Suppose this is our airfoil and this is our computational domain or flow domain over which we would solve for the flow variables okay. We can choose a fairly coarse mesh to begin with suppose this is our coarsest set of grid lines. Let us define our vertical grid lines as well.

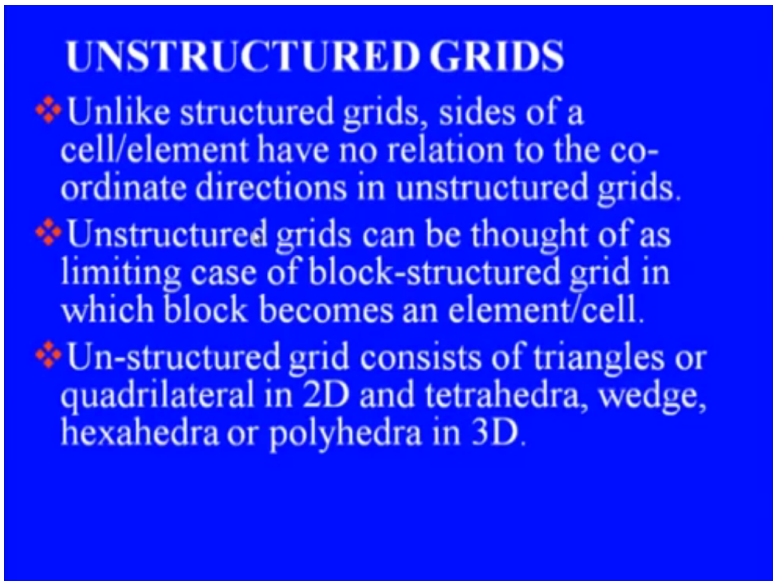
Now far away from the airfoil this coarse grid might be okay, but not near the airfoil so what do we do? We can use finer blocks close to the body of the airfoil. So we can define very fine grids here. So for instance, this is one block now this could be divided into separate sub blocks and even in each sub block we might have the grids of different densities. In fact, we can go for a hierarchy of such grid levels close to the surface of the airfoil whereas bit away from the surface we can do away with coarser grid.

This one typical wherein which we can map all around the surface we can have smaller blocks. We break them into even smaller blocks and have even finer grids close to the surface. So this will help in 2 ways, if we are dealing with we can still keep our self to Cartesian grids, but by using finer and finer Cartesian grids close to the surface of the airfoil, we can get a better representation of geometry that is one thing.

Similarly, these finer grids would be able to resolve the higher gradients, which are present in the flow field close to the surface of the airfoil. Similarly, where we expect the formation of the shockwaves we can again have very fine grid structures by defining local blocks with

very fine Cartesian grids therein. So there are umpteen number of possibilities. I have just illustrated part of the block-structured mesh generation around airfoil.

(Refer Slide Time: 35:09)

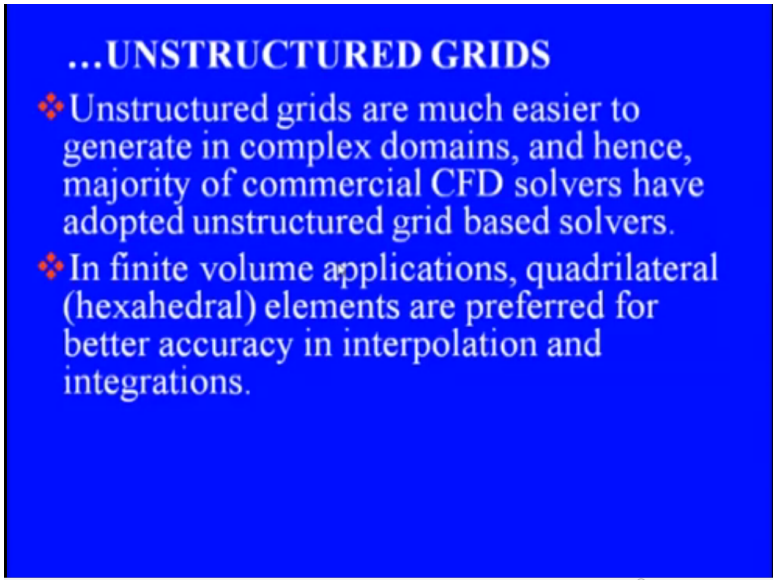


We can fill in same procedure we can follow. Now let us move on to the next category, which we call unstructured grids. Now these unstructured grids they can be thought of as a limiting case of these structured grid or rather our block-structured grid in which each block becomes an element or a cell and unlike structured grids sides of cellular element have no relation to the coordinate directions in unstructured grid.

And usually in 2 dimensions, unstructured grids would consist of triangles or quadrilaterals in 2-D. Triangles are the easiest ones to generate, quadrilaterals specifically in finite volume analysis. They result in better interpolation and integrations. Integration and interpolation can be done more accurately with quadrilateral finite volumes in 2 dimensions. Similarly, in 3 dimensions tetrahedra, tetrahedral elements are easier to generate.

But hexahedral elements give us more control over the accuracy of interpolation and integrations. So in general in 3-D, we can have a collection of tetrahedra, wedge elements, hexahedra or even polyhedral elements. Now polyhedral elements have recently picked up in finite volume analysis and some of the commercial softwares like StarCD they prefer to use these polyhedral elements in 3 dimensions.

(Refer Slide Time: 36:39)

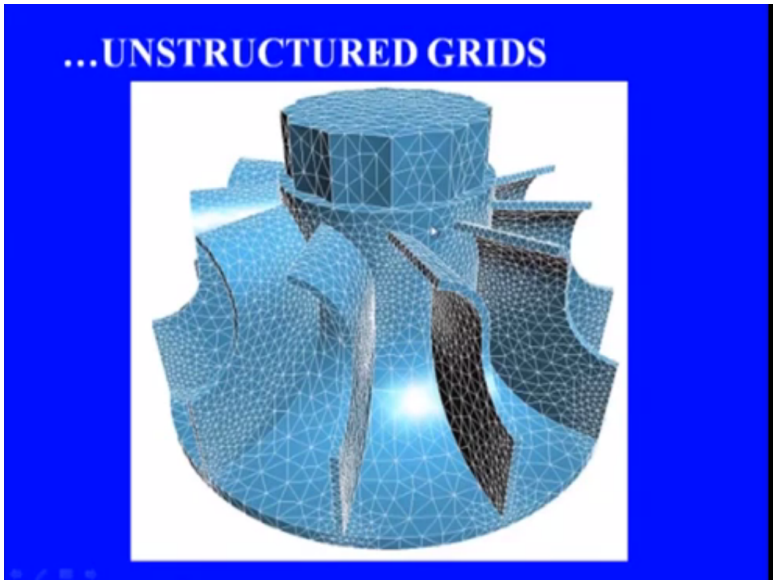


And the unstructured grids are much easier to generate in complex domains and hence majority of commercial CFD solvers have adopted unstructured grid based solvers. In fact, these unstructured grids generators they have been adopted as such from commercial finite element solvers, which have traditionally used unstructured finite element meshes.

Now in finite volume applications as I just mentioned quadrilateral elements in 2-D or hexahedral elements in 3-D are preferred for better accuracy in interpolation and integrations. Now this is one aspect, which you must keep in mind while designing your grid for finite volume analysis of a problem. So now this is one typical example of unstructured grid generated around a turbine root.

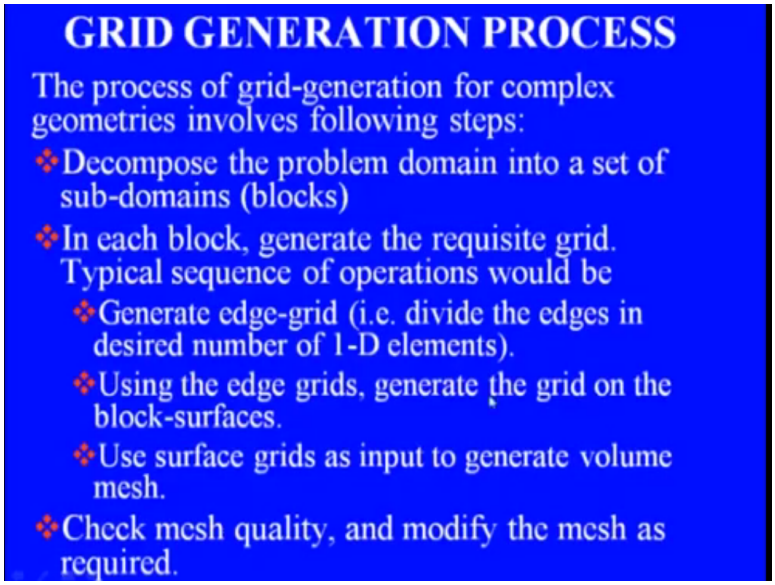
So you can see different grid densities or different densities of what we call cells or meshes and different parts of the problem domain. We have just shown the surface mesh and of course the volume mesh is developed started from the surface mesh if you are dealing with the flow analysis problem and if you are dealing with the stress analysis problem this is typical surface mesh for the structural part of a router.

(Refer Slide Time: 38:09)



So its fluid structure interacts in problem, will have 2 sets of meshes but this starting surface mesh would be what we have seen here. Surface mesh looks a collection of triangles and of course based on these triangles we can generate the tetrahedral 3-dimensional elements. Now next let us have a look at the overall grid generation process, which we will have to adopt for a real life problem.

(Refer Slide Time: 38:40)



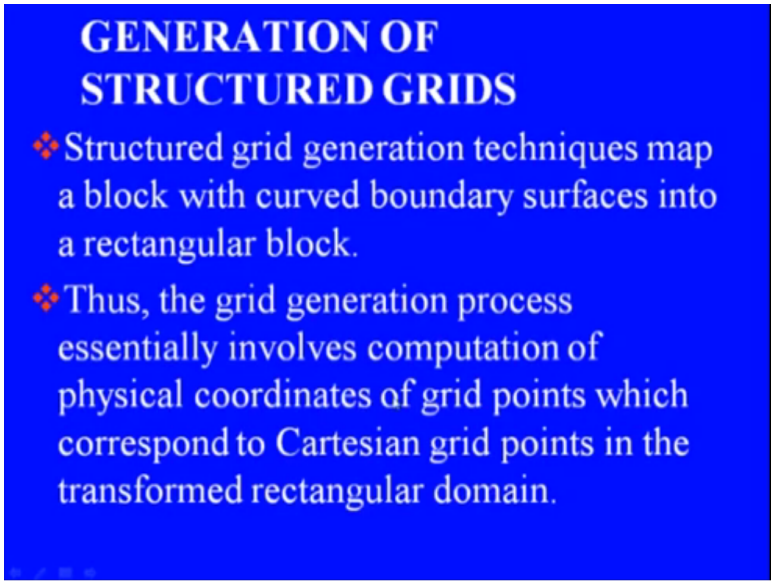
So we will typically use these set of steps for complex geometries. What do we need to first do? Decompose the problem domain into a set of subdomains or blocks, it is immaterial whether we want to generate block-structured grids or unstructured grids. This steps would always be adopted okay and in each block generate the requisite grid. The requisite grid could be a structured one or unstructured one.

For generating this grid in each block, the typical sequence of operations could be generate as grid that is to say divide the edges in desired number of 1-dimensional elements. Then use these S grids to generate the grid and block surfaces. So if you are dealing with 2-D thus the things would be finished, but for 3-D we will proceed further that use surface grids as input to generate the volume mesh.

So if you want to generate a good quality mesh, this is a typical sequence of operations which we must follow. So other commercial grid generated programs will give you an option of generate a volume mesh as such just click on a mouse button that gives you a volume grid but then you will have very little control over the gradation or the quality of such a grid whereas if you follow this 3 step process that is to say it control the division of edges that how many number of nodes you want along each S.

Then generate a surface grid over all the surface of the block we can have much, much better control over the gradation as well as the quality of the volume mesh, which is generated. So once you have generated the mesh in each of the blocks then we have to check for the mesh quality and then modify the mesh wherever required. So that is the typical grid generation process, which is adopted in real life CFD analysis.

(Refer Slide Time: 40:46)

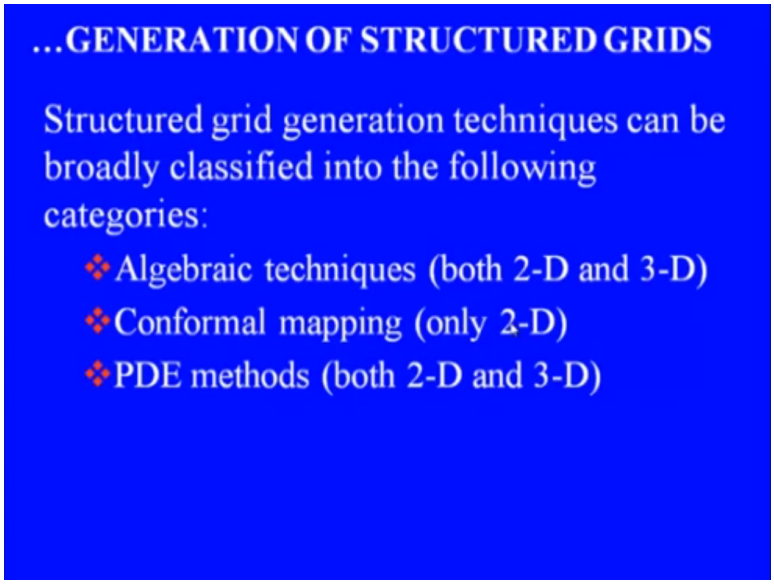


Now let us have a brief look at the generation of the grids. First have a look at how structured grids are generated. Now structured grid generation techniques they map a block with a curve boundary surface into a rectangular block. How do we achieve this mapping? This is

basically achieved by computing physical coordinates of grid points, which correspond to Cartesian grid points in transformed rectangular domain.

Sometimes the word which we use is for this Cartesian grid spaces what we call computational domain and physical domain, which represents our actual physical geometry, come up with the mapping. This mapping could be based on finite element shape functions. It could be based on transfinite interpolation or it could be based on solving system equations, partial differential equations.

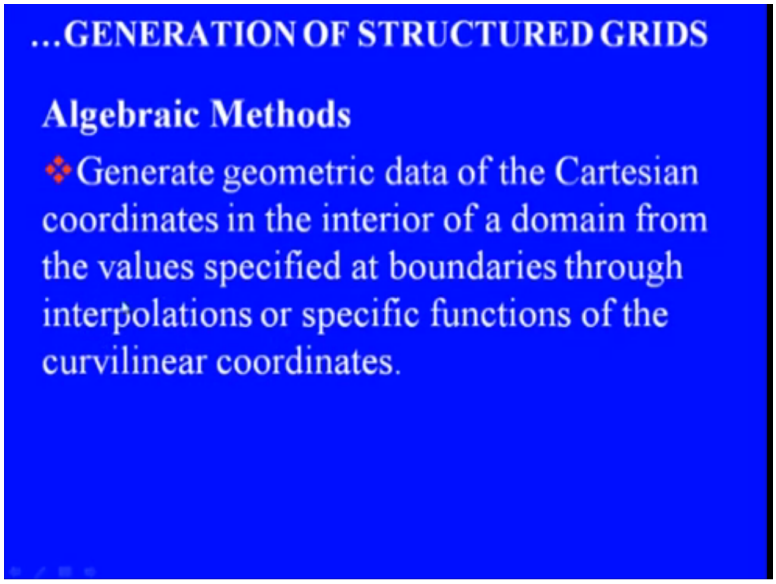
(Refer Slide Time: 41:41)



It does not matter which way we use, but that is what we have to do. We have to generate a mapping and we can classify these structured grid generation techniques into following categories, algebraic techniques which use simple mapping, schemes or interpolation functions these can be used for both 2-D and 3-D, conformal mapping which we have learnt in complex analysis class, but this can be used only in 2 dimensions.

And then partial differential equations based method, we can have an elliptic generation systems or hyperbolic systems, which can be used in both 2-D and 3-D space.

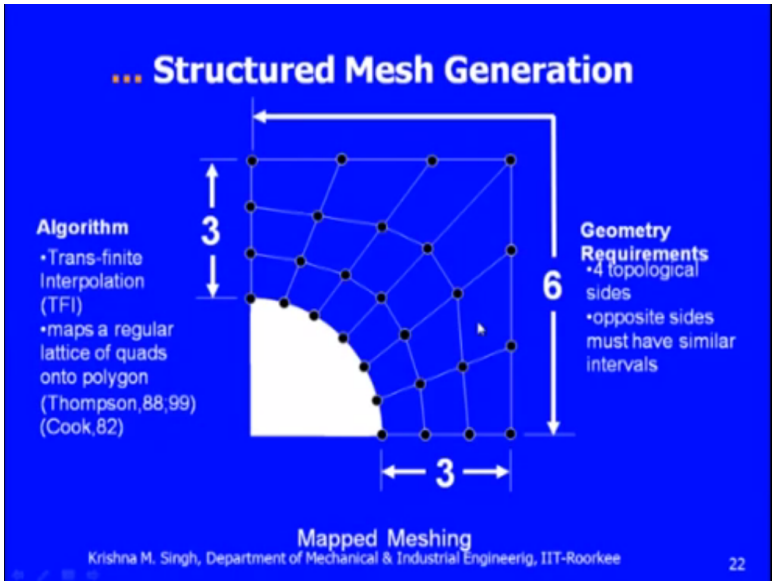
(Refer Slide Time: 42:17)



Now these algebraic methods they generate geometric data of the Cartesian coordinates in the interior of a domain from the values specified the boundary through interpolations or specific functions of curvilinear coordinates. These interpolations could be what we call transfinite interpolations or more complex finite element shape functions and this is one of most popular techniques.

This is very simple to program that is why it is very popular technique. So this is one typical example here.

(Refer Slide Time: 42:35)

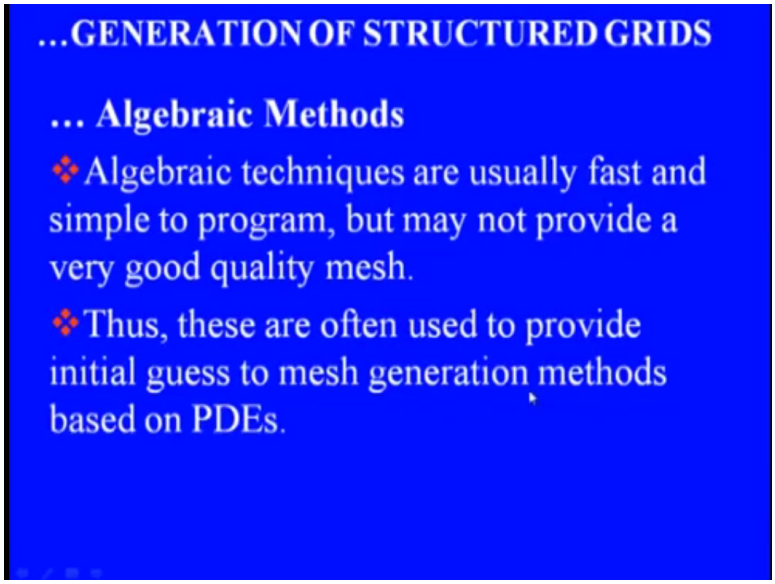


We had a problem domain, which is outside maybe this is one of the solid tubes and this is our fluid domain. We want to generate a structured mesh. So on each side we now got to fix the number of points, which we want to have and these numbers would be same as the

number of points in the opposite side okay. This is 3, this is 3 here. Here we have got 1, 2, 3, 4, 5, and 6 divisions. So these 6 divisions would be mapped to the 6 divisions here using transfinite interpolation and this is what our mesh generated due to lines of the structured mesh. It is clearly non-orthogonal structured mesh based on transfinite interpolation.

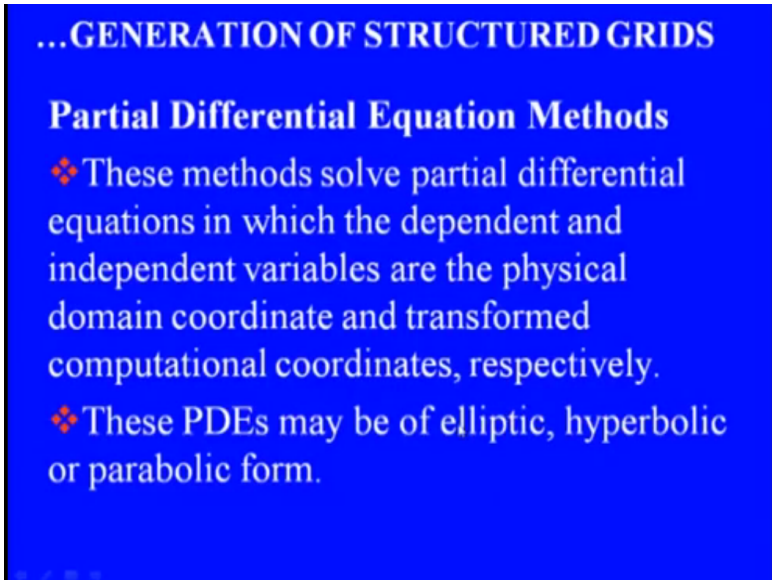
Now these algebraic techniques are usually very fast and simple to program, but they may not provide a very good quality mesh specifically if you are dealing with fairly complex geometries.

(Refer Slide Time: 43:43)



So these are often used to provide initial guess to mesh generation methods based on partial differential equations, which would require us to provide with an initial guess of the grid points.

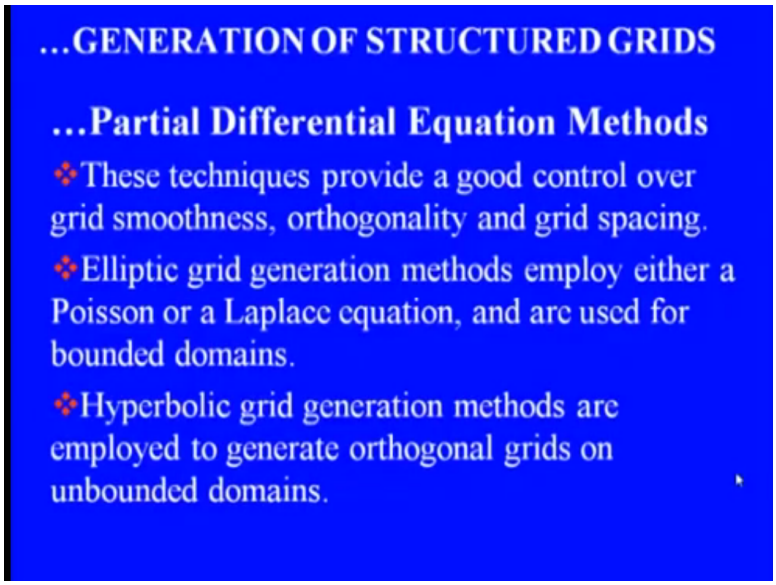
(Refer Slide Time: 43:54)



Now this PDE based methods they require solution of a partial differential equations in which dependent and independent variables are the physical and transform coordinates. Independent variables are the transform coordinates in the computational domain, which will be a rectangular in 2-D and a parallelepiped in 3-D and the dependent variable would be x, y and z coordinates of the physical problem domain.

Now the PDE which we might solve that could be elliptic, hyperbolic or parabolic. Elliptic PDEs are normally used for closed problem domains and hyperbolic would be used for infinite domains.

(Refer Slide Time: 44:33)

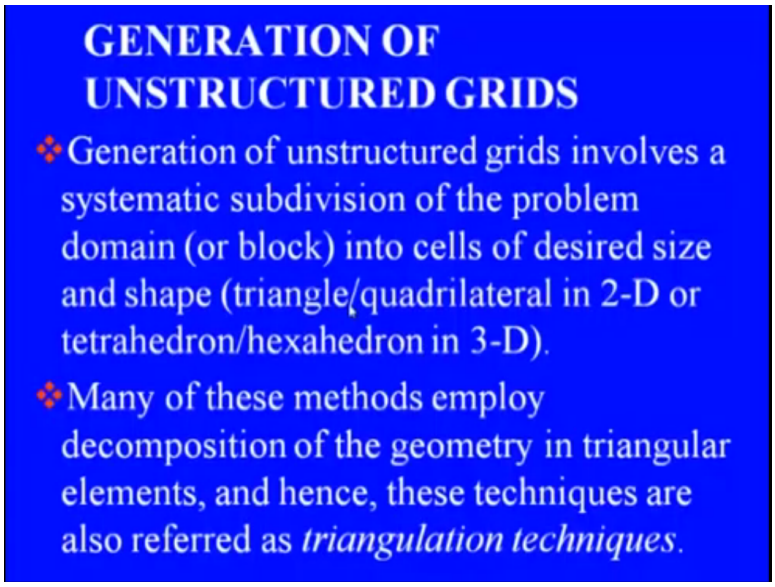


And these techniques provide a good control over grid smoothness, orthogonality and grid spacing. We can have control functions specified as a part of the formulation and this elliptic

grid generation method they can employ either a Poisson or Laplace equation are used for bounded domains, hyperbolic methods would be used for generating orthogonal grids on unbounded domains.

For further discussions, we will refer you a few books towards the end of the lecture. Please have a look at the equations, which are involved and how do we solve these equations to generate these structured grids.

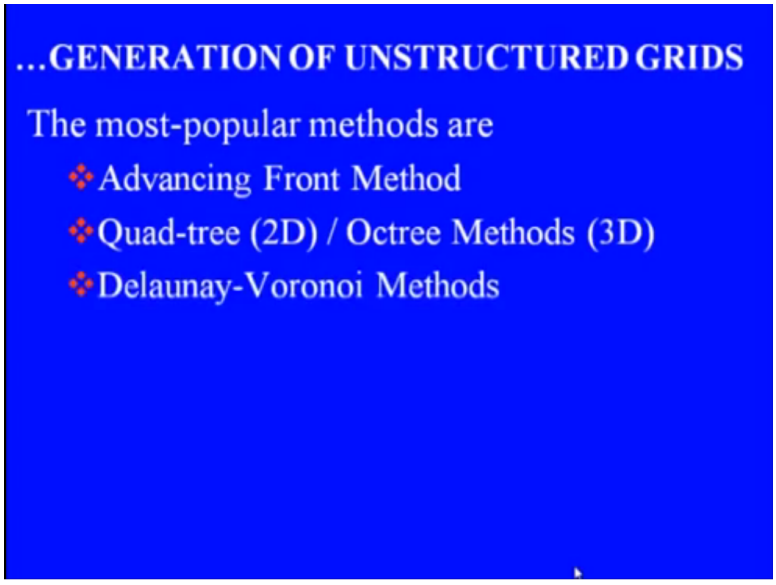
(Refer Slide Time: 45:11)



Next is a very fleeting look at unstructured grid generation process. Now unstructured grid generation process involve a systematic subdivision of problem domain into or what we call sub blocks into cells of desired size and shape, for instance triangles or quadrilaterals in 2-D or tetrahedron or hexahedron in 3-D and many of these methods employ decomposition of the geometry in triangular elements.

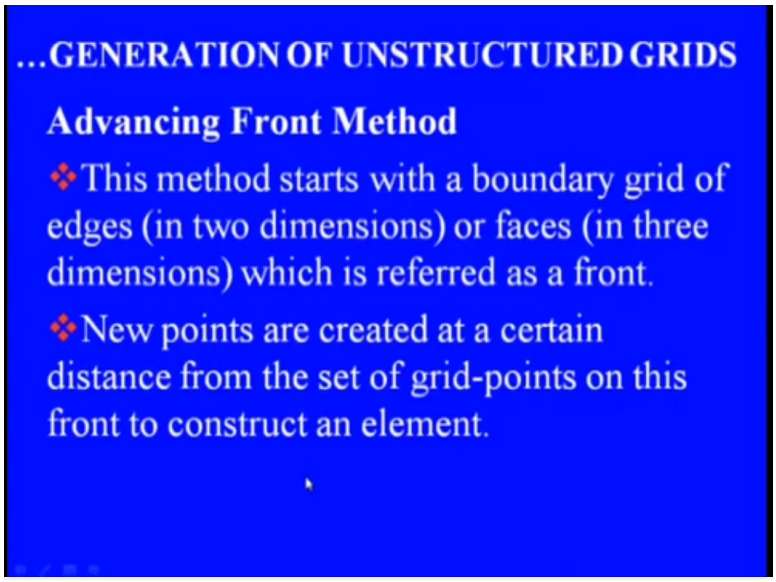
Hence, collectively these unstructured grid generation techniques are also referred to as triangulation techniques though you might generating a hexahedral mesh, but still the generic name for techniques is what we call triangulation techniques because this is how the things started off with that generation of triangular elements in 2-D.

(Refer Slide Time: 45:54)



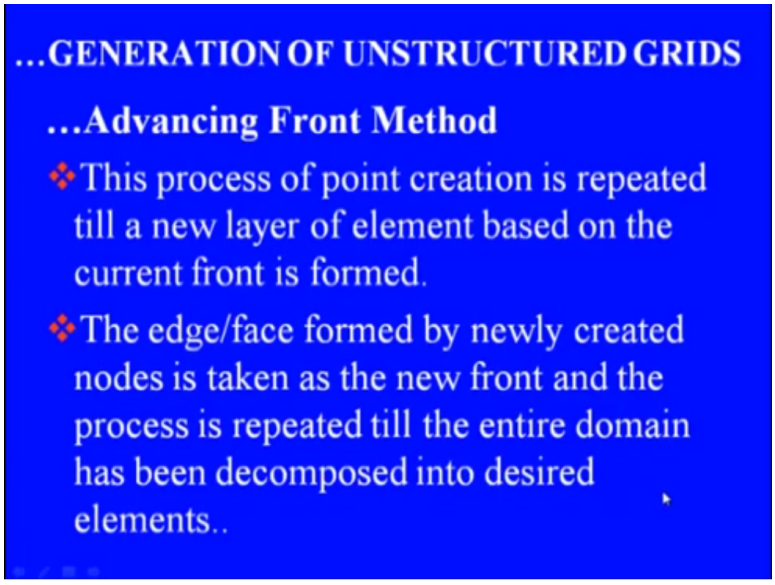
Now let us have a look at some of the popular methods advancing front method, Quad-tree or Octree based methods and Delaunay-Voronoi based methods.

(Refer Slide Time: 46:15)



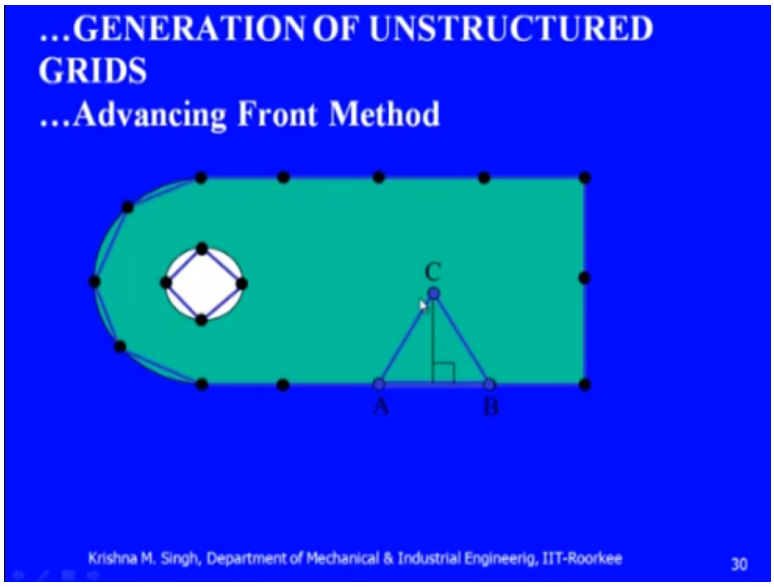
In advancing front methods, we will start up with the boundary grid S in 2 dimensions or boundary face, which would be discretized in 3 dimensions and that would be used as our starting base what we call as front. Starting from that front we would generate yet another layer of nodes, which we call new points at certain distance from the set of grid points on this front to construct an element.

(Refer Slide Time: 46:31)



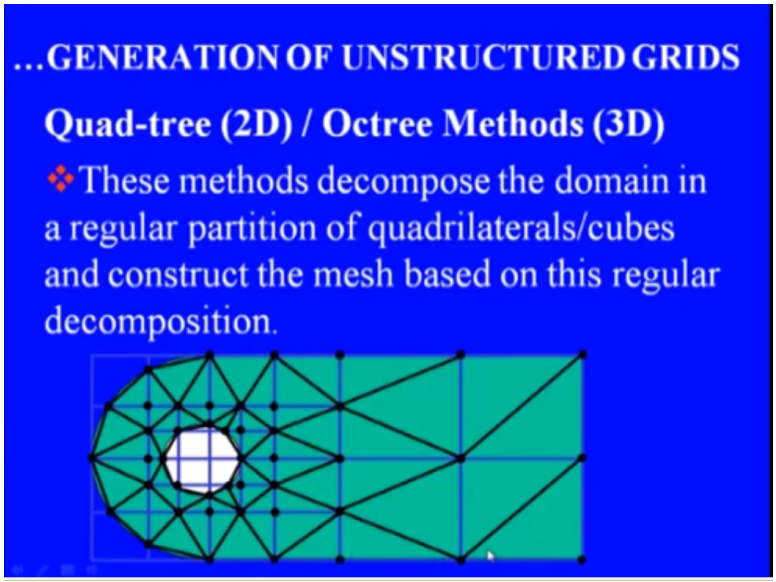
Once that has been done the process of point creation is repeated till a new layer of element based on the current front is formed that becomes our new front. So edge/face form by newly created nodes is taken as new front and the process is repeated until the entire domain has been decomposed into desired elements like this could be a typical sequence here.

(Refer Slide Time: 46:53)



We have got our boundary mesh, let us take these 2 nodes and choose a certain height at which we want to generate the new layer of node C. So generate these ones, this will satisfy certain properties, generate a new layer and then we have now filled up the entire problem domain with elements of our choice. Now this Quad-tree or Octree based methods, they employ the Cartesian decomposition.

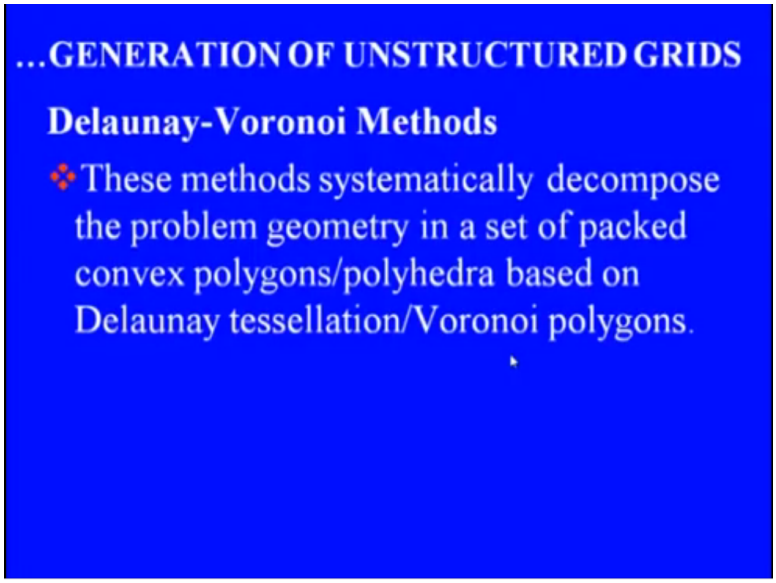
(Refer Slide Time: 47:23)



They decompose a domain into regular partition of quadrilaterals rather squares and cubes and construct the mesh based on this regular decomposition. Suppose this was our outline will give us a view rectangular decomposition or say bounding rectangle keep on decomposing it into smallest rectangles until we have got the nodes at these boundaries and thereby we have obtained a decomposition of domain into elements of required size and shape.

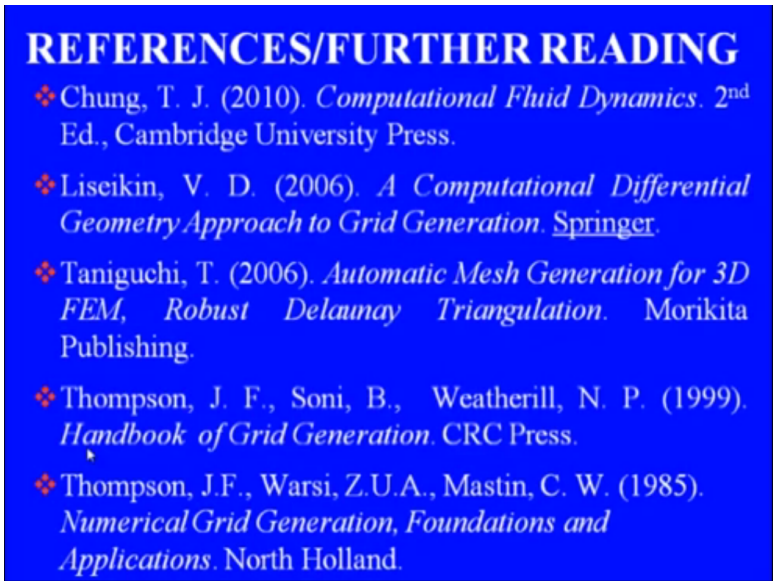
So depending on where we are we will stop the decomposition process for instance here we are happy with fairly large Cartesian decomposition, just add these 2 nodes to get triangular mesh. If you want a rectangular mesh, we want to add these 2 nodes we will just have a simple rectangular element here. The same thing we can do here, we can create a partition here to get 2 quadrilateral elements and so on.

(Refer Slide Time: 48:30)



Last category is one of the most fascinating one and it also one of the most difficult to program they are called Delaunay-Voronoi methods. They generate a tessellation or what we call triangulation based on Voronoi polygons. So these methods systematically decompose the problem geometry in a set of packed convex polygons or polyhedral based on Delaunay tessellations or they also called Voronoi polygons.

(Refer Slide Time: 49:00)



And further details of these methods please have a look at these references. You can find some quite elaborate detail in the computational fluid dynamics book by Chung. Recent book which is dedicated to grid generation is by Liseikin published in 2006 is called computational differences geometry approach to grid generation. Yet another book for 3-D based expressively on Delaunay triangulation is that by Taniguchi published in 2006.

This gives you how to write in automatic mesh generation scheme and if you want to have compendium both structured as well as unstructured grid generation methods, the handbook of Thompson, Soni and Weatherill is a very good resource. You can find a collection of different chapters dealing with different types of grid generation procedures and earlier version textbook by Thompson, Warsi and Mastin.

This is a simpler one so maybe if you just want to have a start off we can start off with Thompson's book and then you can refer the handbook or go to Taniguchi's book or Liseikin's book for further learning on aspects of grid generation.