Computational Fluid Dynamics

Prof. Sreenivas Jayanti
Department of Chemical Engineering
Indian Institute of Technology, Madras

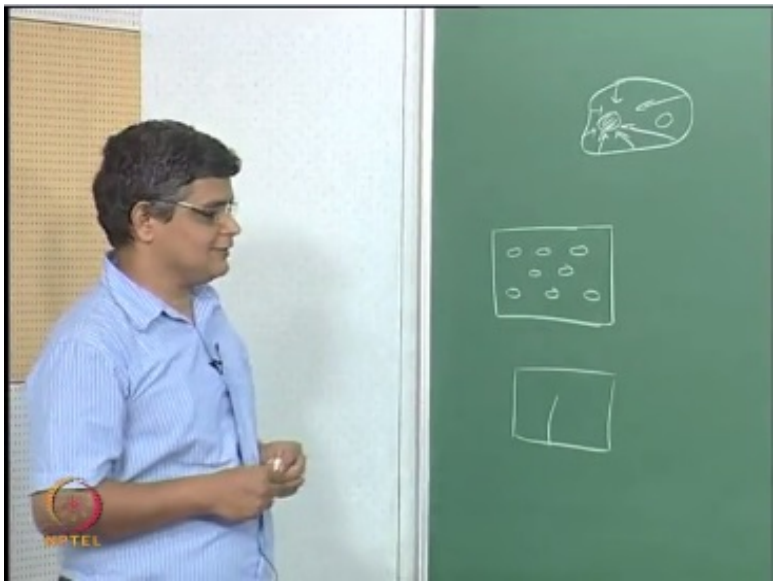
Module No. # 07
Dealing with Complexity of Geometry of the Flow Domain
Lecture No. # 45
Topics
Unstructured Grid Generation
Domain Nodalization
Advancing Front Method for Triangulation

We have seen that for a given geometry, which may be such that it cannot be described in simple coordinate systems, we can come up with a body-fitted grid one in which that coordinate lines follow the shape of the body in terms of especially the walls that enclose the overall control volume. We have seen that the generation of this grid is fairly complicated and you have to solve partial differential equations especially if you want to have some control over the way the grid is structured. And, not only that, if you want to do for example, impose orthogonality condition especially at the wall, we have to do some post-processing, we have to do further work in order to do this. But, generally speaking, it is possible to get a body-fitted grid for a three-dimensional flow geometry using the partial differential equation approach.

The other alternative to this kind of structured grid is the unstructured grid. And, we have remarked earlier in one of the lectures that unstructured grid gives us great flexibility in dealing with complicated geometry. And also, it gives us great flexibility in terms of local refinement of the mesh. So, the question is what kind of method or algorithms we can use to generate an unstructured mesh. It may look trivial, but the actual problem is not so easy. And, one has to make sure that the grid that we get is such that there are no overlapping parts. And therefore, the internal meshing and joining of several points together to make up control volumes and to make up the tiles is not a trivial thing; one has to do a lot of book keeping.

There are many methods used by many researchers for the unstructured mesh generation. In our course, we will look at only the case of two-dimensional computation domain and then we can see what issues are involved in this. We would not be dealing with every possible algorithm, but we look at a possible algorithm by which we can generate an unstructured mesh for a given domain. If it is a simple domain, it is not so difficult. But, when we have what is known as multiply connected domain, that is, a domain with holes inside – for example, you have a duct of rectangular cross section and inside that you put a tube; and then, the flow can take place only in the region, which is not occupied by the tube. So, that is an example of a multiply connected domain.

(Refer Slide Time: 03:25)

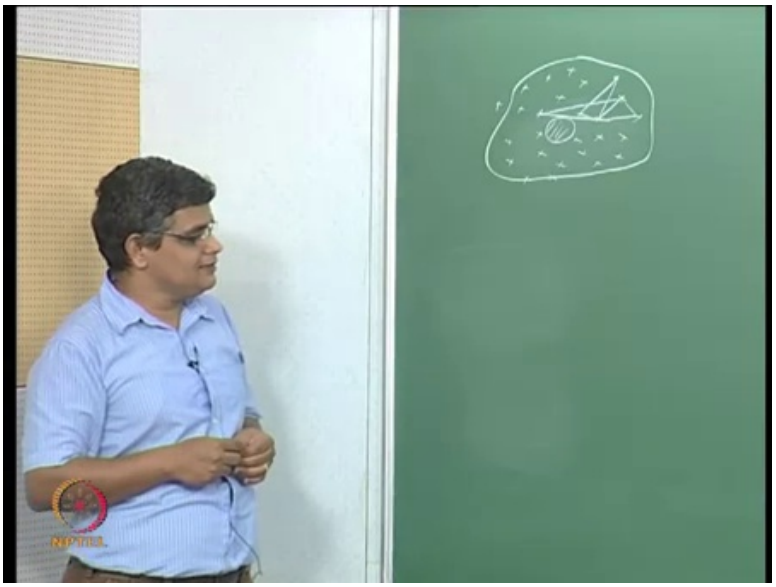


Essentially what we mean by multiply connected domain is that if we had a flow domain like this and if we were to shrink this whole thing, write down to a point and if you are left with nothing there, then it becomes a simply connected domain. But, if you have a flow domain here, which is not part of this – so, this is not part of the flow (Refer Slide Time: 03:45) domain, no matter how much you shrink here, there is this part, which is remaining. So, in that sense, we call this a doubly connected domain. And, if we have two such things, then it becomes multiply connected domain and so on like this. So, these multiply connected domains are quite possible if you are looking at for example, flow through a heat exchanger either this type with so many tubes in staggered arrangement; or if you are looking at flow with some baffle plates for flow distribution. These are the kinds of things that are quite often occur in practical industrial situations,

where you want to put some flow optimization devices within your duct, so that the overall pressure drop is minimized or that the flow maintains uniformity.

Or, if you want to reduce noise and so many other features for which you would like to intervene into the duct domain, then you want to make some changes to that. So, in normal practice, we do not have the luxury of simply connected domain we have the practicality of multiple connected domains and we have to generate a mesh in spite of the complexity that is there. So, we are going to look at the generation of an unstructured grid for a two-dimensional geometry essential to three-dimensional on a case by cases. And, there are special algorithms developed for that.

(Refer Slide Time: 05:29)

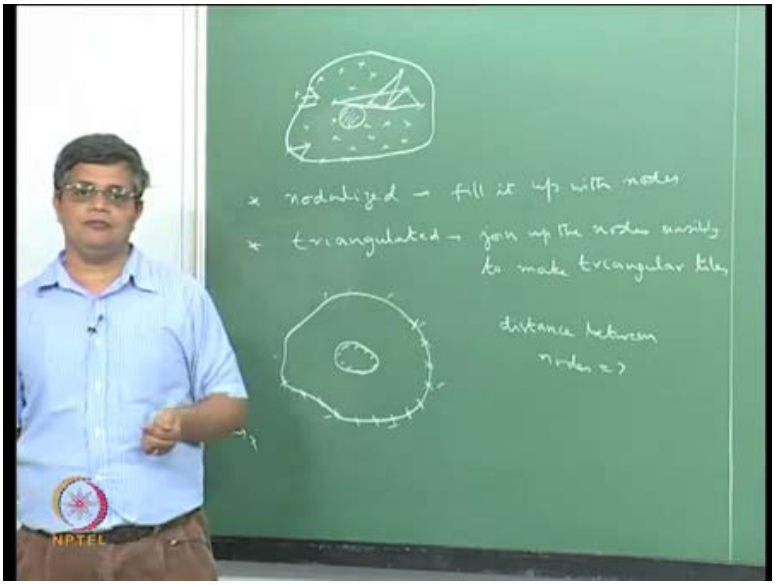


Now, let us say that we have this as the flow domain. The objective of the grid generation is to discretize this domain into small tiles, such that when you put all the tiles together, you get the shape. And so, we need to have points on the boundary and points in the interior also. And, as mentioned several times that in a CFD solution, we would like to have these points at which we want to evaluate the solution variable. The number of points should be very many and they should be spread throughout the domain, so that at every part of the domain, we can make reasonably accurate approximations for the derivatives that appear in the governing equations. So, we would like to have some way of spreading points throughout the domain not leaving out any part and we also would like to have a method by which we take only the points, which are inside the domain and

not outside. So, in the process of doing this, we cannot take a point here, because that is outside the domain and that is outside our zone of competence of our interest. So, we need to have a very distinguishing algorithm – one which distinguishes between what is outside and what is inside and then only deals with what is there that is inside.

And, if there is a body, which is inside this, which is not part of the domain, then that method should also make sure that this is a no go area and within that there should not be any points. So, putting the points itself is one challenge in the general case. The second challenge is that we should be able to join this together in such a way that we get tiles. For example, one can say, visually, I will join these three points to make this triangle (Refer Slide Time: 07:42). How will I know that I should join only these three and not for example, this, this and this, and this, this and this. So, I can see some areas, which are overlapping here. Firstly, one should have an algorithm by which we can put points throughout the domain including the boundary and only on the boundary and including the domain not outside the domain. And, one should also have an algorithm by which we can join them in a sensible way, so that there is no overlapping, so that there are no nodes that are left out, so that there is no area of the tiles, which is left out. So, these two stages: one of putting points throughout the domain in as uniform way as possible taking account of the boundaries inside and outside of the flow domain; and secondly, join them in a sensible way so as to get an unstructured tile. These are the two stages of an unstructured mesh generation.

(Refer Slide Time: 08:56)



When we talk about unstructured, by definition it is unstructured, so that the tile that we have can be a polygon of n number of sides. The simplest possibly is a triangular tile, so that any complicated shape here can be put through in the form of a triangle with one side always coinciding with the boundary like this. So, triangulation of the flow domain is very useful way of generating an unstructured mesh. Although one sees the structure in the sense that every part of this domain is a triangular, it is still unstructured in the technical sense that the points that we are looking at here are not associated with any coordinate lines. So, when we talk about a structure, it is not about what elements we have made use of to discretize to cut this up into tiles. But, whether or not the nodes that are associated with these tiles are at the intersection of coordinate lines. So, in that sense, even if we use the same element throughout, it is still an unstructured mesh as long as we make the point that these points at which we are joining together to make the tile are not associated with constant values of coordinates either x, y, z or psi, eta, zeta that type of thing.

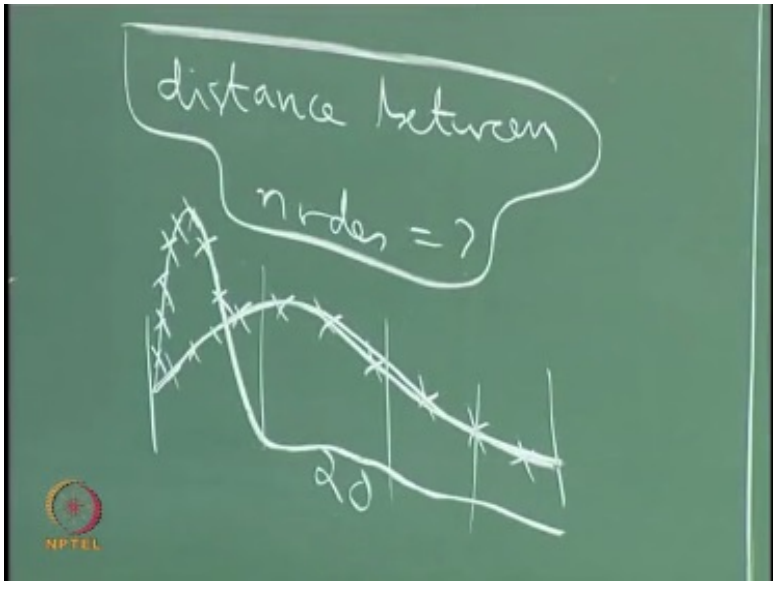
We are going to look at something like this kind of computational domain. Firstly, nodalized and then triangulated. So, when we say nodalized, we would like to cover this entire area and only that area, which is available for flow and fill it up with nodes. Spread throughout with some notional distance between the nodes. We will see that we cannot exactly satisfy that the distance between any nodes is the same for all the points, but we would like to have some notion of what is the distance that we would like to have between nodes and we would like to honor that particular thing.

And, triangulation here – we would like to join up the nodes sensibly to make triangular tiles in such a way that when we put all the triangular tiles together, then we get the entire flow domain and only the flow domain nothing beyond, nothing less. So, how do we do this? We would like to describe an algorithm for each, which appears to work on the face of it and which has been successfully used by our students to do nodalization and triangulation. So, what we are looking at is, first part is to take a domain like (Refer Slide Time: 12:55) this and we want to take for example, a domain in which this is the flow area and all this is wall, so that it makes it some way a doubly connected domain. So, this is not a simply connected domain. And, we would like to put points throughout this. So, how do we do this? We are looking at an xy plane. And, to start with, we have a complete description of this bounding curve and this bounding curve, so that we know

the x, y points at each of this. So, we can start with at some point here and then go along this curve and then put boundary points at some distance, which is roughly the distance that we would like to maintain between nodes.

Now, what should be the distance between (Refer Slide Time: 14:02) nodes? There is no single answer to that. The distance between nodes depends on what we can afford, because if we make the distance very small, then that means that will have very large number of points. And, large number of points means that large number of storage of the variable information and also large matrices in terms of a phi equal to b type of solutions. That means large amount of computational time. So, we cannot put very small distance between the nodes. But, at the same time, unless the distance between the nodes is very small, we will not get accuracy in terms of representation of the first and second derivatives, which appear in governing equations using first order or second order accurate formula, because we have to keep in mind that we do not have the luxury of going to higher order of approximations especially in the unstructured meshes. And therefore, we have to restrict ourselves first or second order accuracy, which means that the distance between nodes should be small. So, we should have a compromise between the amount of computational storage requirement and time requirement for the solution of a phi equal to b (( )) equations and the accuracy requirement that is needed.

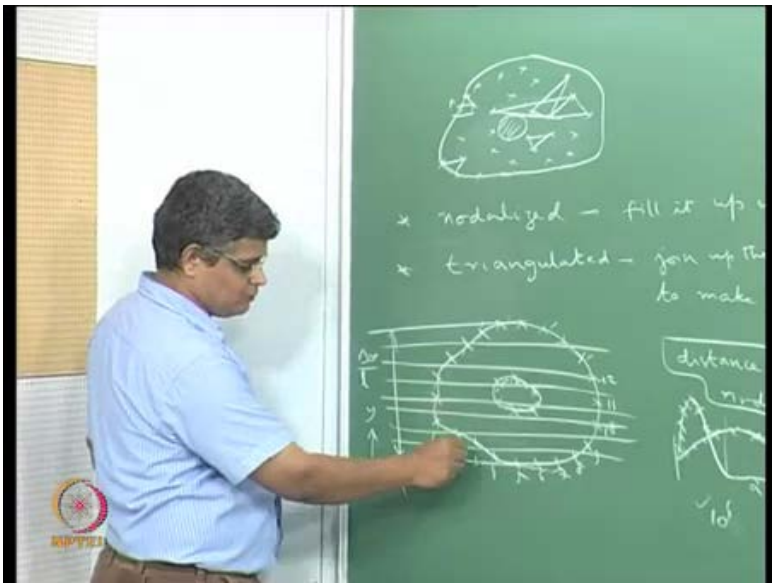
(Refer Slide Time: 15:59)



For a given problem, it may not be something that is known straight away, but at least we should have some idea of what kind of flow profiles we may expect. And, based on that, we may want to have enough number of points. If this is the domain length and if you are getting a velocity profile like this, then we should have enough number of points to make like this, so that we can join them by smooth line to get a velocity profile. So, if you want to capture a velocity profile like this, then if you have only four points at which the velocity is evaluated, then we would not be able to get a faithful representation of this curve. Therefore, one would say that for a curve like this, we would like to have at least 20 points to be able to represent. That is a useful number. But, at the same time, if the same number of 20 points will not be useful, if you had a velocity profile like (Refer Slide Time: 16:53) this, because this is a steep variation and probably in order to capture this, we need more number of points here. So, we need to have grid points like this. So, from that point of view, the distance between nodes is related to the kind of flow solution that we are expecting and that we want from the problem.

And, it may not be settled right at the beginning itself. One may have to start with some grid and some distance and then come up with a grid, have a computation. At the end of that, we have to see whether the grid that we got is good enough. If not, we can go back and remesh it with more spacing or less spacing like that. So, in that sense, it is someway dynamic process that we have to adapt. But, we must have some notional idea of what this distance between the nodes (Refer Slide Time: 17:50) is and also in terms of the accuracy and in terms of how many grid points we can afford. Based on our computer time and based on the past practice, we may say that let us have 100 points in one side or we can say we can have a total number of 10,000 points for a two dimensional problem; or, we can got up to million points. If we are doing a time dependent problem with very fine time step, then we want to go through large number of computational time. We may not want to be doing with ten million grid points we may want to come down to 10 to the power 4 grid points. If we are done with 10 to the power of 4 grid points and then found that the resulting… There are lots of fine details that you want to resolve. Then, you may have to go to a higher number grid points. So, in that sense, we have to suck it and see in a way if we have absolutely no idea of what we have to do, what the grid spacing the node should be.

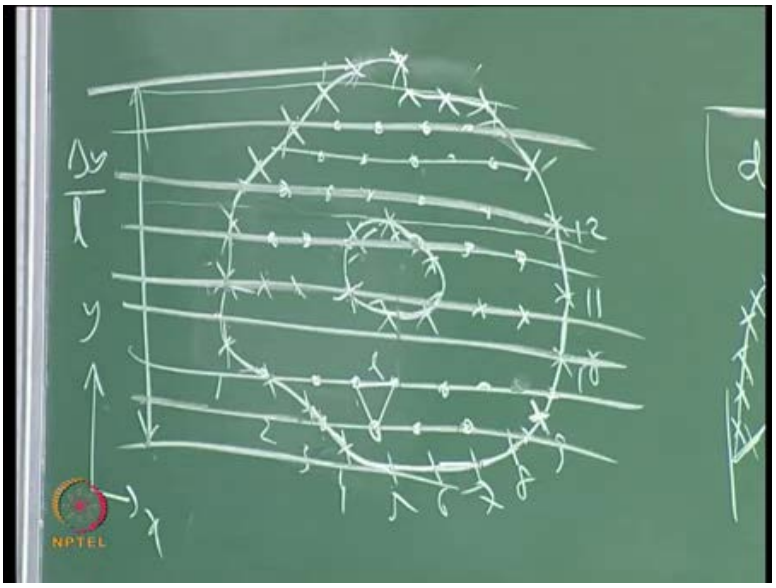
(Refer Slide Time: 19:00)



But, when we start the grid generation, we come up with some idea of the distance between nodes and we can use that as a measure to start the discretization of the boundary nodes like this and also the interior boundary; whatever is all the boundary of the computational domain, we have to do it. And, at this stage, we would like to number these boundary nodes like 1, 2, 3, 4, 5, 6, 7, 8, 9 and so on. And, it is usually to… If we measure doing like this in an anti-clockwise way, the interior domains are numbered in the clockwise way. So, that kind of thing is there. But, at this stage, what we are interested in is to fill this domain with lots of interior points, such that they are only interior not exterior and such that there is some defined spacing between the grid points, so that ultimately when we joint them, we have a size of the tile, which is not very far from what we wanted to be. So, how do we do that?

We know that overall maximum height (Refer Slide Time: 20:24). So, this is the variation of the minimum height and the maximum height. And, this is where the flow domain is constrained. And, we can divide this maximum or the delta y by that l, which is roughly the distance between the nodes that we want to have. And, based on this, we can draw parallel lines across this, which are separated by the distance of l here.

(Refer Slide Time: 21:17)



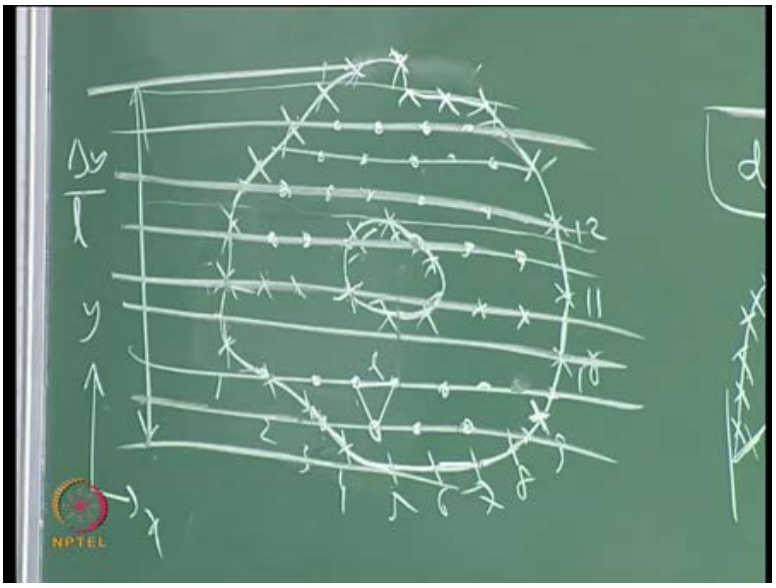
And, we see that these l, these horizontal lines will cut this domain at a several points, which we can find out. We know this curve and we know the equation for this; we can find out; otherwise, we can look at now… What we are saying is that this boundary here is now represented by so many edges. And, that edge is a linear element between two successive nodes like this. So, this is the line and this is also a line. So, from that point of view, we can check which edge is being cut by which line here. So, we can evaluate the points of intersection of each horizontal line with the boundary and the boundary which is exterior boundary as well as the interior boundary. So, now, we take that line, which has even number of intersections with the boundary.

For example, this (Refer Slide Time: 22:17) line here has intersected the total boundary at two points: one point and one point here. And, this one has intersected at four points between this point, this point, this point and this point here. And, if you were to draw a line here for example, this might have intersected here and here and here. So, there is an even number of things; that is an odd number of things, in which case we would like to move this line slightly, so that we have even number of intersections. And, odd number of intersections are possible when the boundary line for example, is also horizontal. For example, if I make something like this and I am drawing one of the horizontal lines to coincide with this, then I may have an odd number of things like that.

Essentially, what we are trying to do is that we want to take those line segments between two successive boundaries. For example, I take this line (Refer Slide Time: 23:22) here; this has cut the boundary at four different points. And, I take this line segment here and then I put number of nodes such that they are separated by a distance of l – 1, 2 and 3. So, I am putting internal points along the horizontal line segments, which are bounded by two boundary nodes corresponding to this intersection here. So, similarly, this point of intersection, this point of intersection constitute one line segment, which break up into integral number of small segments roughly of length l each. So, I put a boundary point here (Refer Slide Time: 24:06); I put this. Again, I take this length here and I put this segment here; I take this segment; then, I put 1, 2, 3, 4; that is a bit smaller, but it is OK.

Again, I take this here (Refer Slide Time: 24:26) and then I can 1, 2, 3, 4, 5, 6. So, I can locate internal points along the horizontal lines that I have drawn in such a way that the two successful horizontal lines are distance l apart, where l is roughly the target distance between the nodes, so that the individual points that I have here are of the order of l distance here. And, they are also vertically of the order of l distance. When I cut this segment also into roughly length l, one can see that I have a point here, I have a point here and also I have a point here. If I were to join these, I would have a triangular element of roughly of length l as a side. So, that is objective of this kind of nodalization. So, the whole objective is to put points only within the domain and not outside the domain. So, that we are ensuring by making sure that we are taking a line segment, which is bounded on either side by the boundary.

(Refer Slide Time: 21:17)

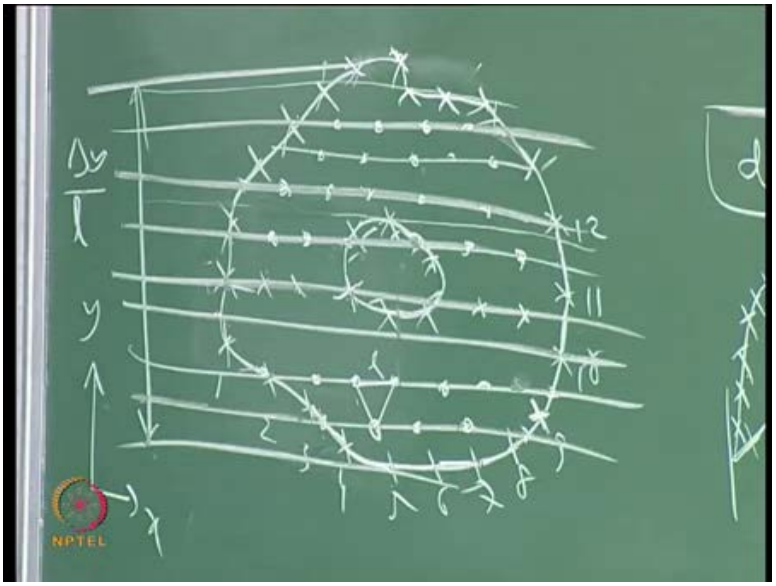


For example, if I have even number of intersections like two points of intersection, then I know that this whole line here from the first intersection point to the second intersection point lies within the flow domain. And, I can put internal nodes at each of these anywhere on this line. And, I put them in such a way that they are at length l apart – here, here, here, here. If I come to this line, this horizontal line (Refer Slide Time: 26:17) has intersected the overall boundary at four points. And, I can evaluate these points of intersection and this point of intersection, this point of intersection. And, I can see that this horizontal line now consists of three line segments: the first one, which is within the flow domain; the second part, which is outside the flow domain; and then, the third part, which is inside the flow domain again. So, I take each line segment, which is inside the flow domain and then I can put points, which will be eventually the internal points, internal nodes like here. So, in this way, I can fill up the entire flow domain and only flow domain with nodes, which are roughly at distance of l apart.

And, if there is a point, which is lying too close to the boundary, then I can remove those points at the end of this. For example, let us say that I have… These are the overall points here and I find one point here, which is too close to this point, which is one of the boundary points. Then, I have a choice of removing this completely; or, I can move it back along this; I can readjust the lines here a bit and do that kind of post processing. So, the algorithm for nodalization, that is, filling up the domain with nodes, which are separated by a target distance l consists of breaking up the boundary, which is a known

curve into line segments, which are at distance l each; that is, the segmentation of the boundary line both the inside one and the outside one. And, having done that, you look at the overall spread of the flow domain in the y direction. We can also do it in the x direction, but we take it in the y direction and overall spread is this much. And, we draw horizontal lines (Refer Slide Time: 28:47) within the spread, which are separated by distance l starting from the lowest point here. And, these horizontal lines intersect the boundary edges at several points depending on what kind of internal domains we have; they may intersect at two points or they may intersect at three points or four points or six points, eight points; we have many such things; then, they can be like this.

Refer Slide Time: 21:17)

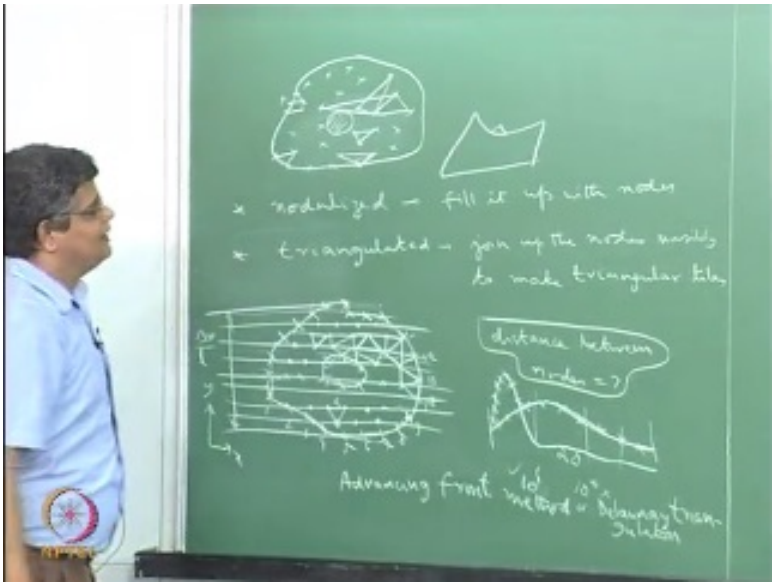


And, they may be intersecting at a boundary node or they may be intersecting between two boundary nodes like this way (Refer Slide Time: 29: 27). Now, once we find out all the points of intersection, at this point, we break up the entire horizontal line into line segments such that each line segment is bounded by the boundary points. So, if you have if you have even number of intersections, then one can say between first intersection and second intersection is part of the flow domain; between second and third is not part of the domain; between third and fourth is again part of the flow domain; and then, the next one will be not part of the flow domain like that. So, we can identify those line segments, which are part of the flow domain and we break up each line segment again into small parts by placing nodes, which are at distance of l apart. So, we do that consistently. If there is any horizontal line, which is intersecting at odd number of points, then we move

the horizontal line slightly up or down, so that we have even number of intersections. So, in that way, we can break up the whole domain into horizontal line segments, along which we put nodes, which are spaced at distance l apart.

At the end of all these (Refer Slide Time: 30:59) things, we will have many points, which are roughly at distance of l apart in the horizontal direction, also in the vertical direction; not exactly l, but roughly a distance of l. And then, we see whether there are any points, which are lying too close to the existing boundary nodes. If they are very close, then we can delete those things, because if they are very close here, then if we join them by a triangle, then we have an edge, which is very small and that is not desirable. So, as part of the overall discretization triangular element, we would like to have an element, which is more or less equilateral.

(Refer Slide Time: 31: 50)



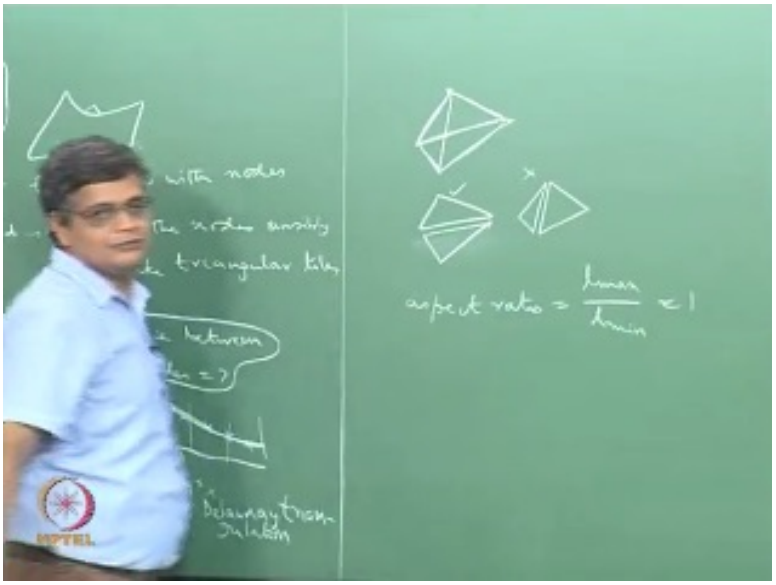
We do not want to have a node, a triangulation, which is like this with a very small node on one side and then large lengths on the other side. So, this is possible if we have a boundary node and if we have an interior node, which is very close by. So, in such a case, we just remove this; and eventually, when we do the triangulation, this node will not be there; we can make a triangulation like this. That is better than having a triangulation which is very small. So, in this way, we can do nodalization of a given two dimensional domain, which may be either simply connected or multiply connected.

The next task is triangulation here. Triangulation here means that all the nodes that we have put on the boundary end in the interior will have to be joined together in order to make triangles. For example, I can join these three, I can join these (Refer Slide Time: 32: 57). So, I can go on doing this. But, here I am making use of my eyesight and hand-eye coordination to do these things. But, we would like to have a computer to do this automatically and in such a way that it does not do overlapping. So, we need to come up with a good way of doing this.

And, there are two algorithms, which are widely used: one is what is known as an advancing front (Refer Slide Time: 33: 38) method; and, the other is known as Delaunay triangulation. There are advantages and disadvantages to either of this, which is what we normally find in any scientific domain. If there are two things, then both of them must have some advantages and disadvantages; otherwise, the one which has only disadvantages, will not survive except as a text book example for example.

As part of a learning process, we may sight something, which would not work, so that we can identify such non ideas right in the beginning itself. But, these (Refer Slide Time: 34:49) both the methods are used and they offer certain advantages and also certain disadvantages. And, the primary advantage of an advancing front method is that it can also deal with concave domains. When we say concave domains, something that is a flow domain like (Refer Slide Time: 35:12) this and specifically the advancing front method will make sure that no nodalization of this kind takes place; that is, it would not consider any point, which has part of the domain, which is outside this. And, whereas, Delaunay triangulation may not be able to take account of this concaveness and it may come up with some tiles which have a part of this outside surface also included as part of the domain.

(Refer Slide Time: 36:02)



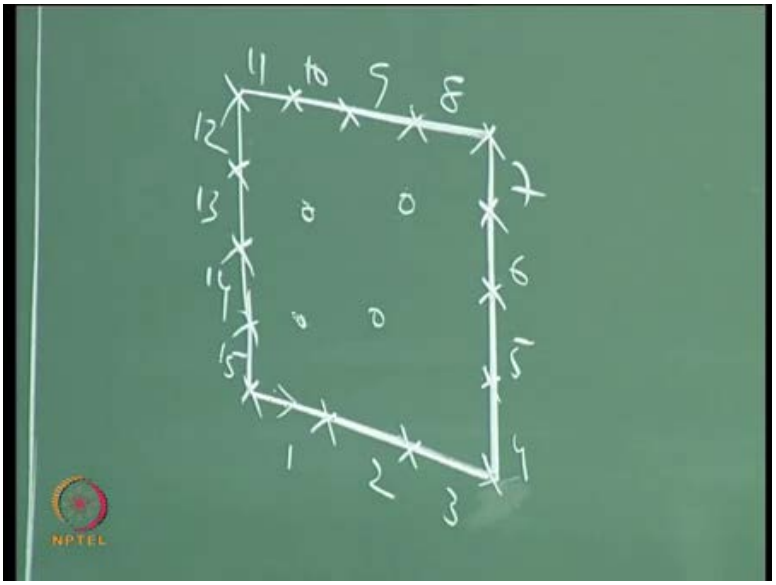
The advantage with the Delaunay triangulation is that it has an inbuilt feature, whereby we can triangulate for example, four points in the best way that is possible. If you have four points here and if you want to this a quadrilateral, then one can immediately see that we can make it into two triangles we can make it into triangles in two ways either like this or like this; into two triangles like this or in this way. So, which of them is better?

One set of triangles like (Refer Slide Time: 36:50) this and the other set of triangles is… may be some exaggeration, but these are the two possibilities. And, between the two, which are covering the same overall quadrilateral area, one would prefer this, because in this particular combination, both the triangles are more equilateral than in this particular case. In this case, we can say that is almost equilateral, but this has got one side, which is too small. So, this has a high aspect ratio; that is, the ratio of the largest line segment to the smallest line segment within the triangular thing can be taken as the aspect ratio. And, we would like to have an aspect ratio, which we can say is l max by l min. We would like to have an aspect ratio of one, so that we have an equilateral triangle.

But, for a given set of four points, the Delaunay triangulation can try to choose that way of decomposition into triangles, which gives us the least lower aspect ratio. So, it will automatically choose this particular way (Refer Slide Time: 38:17) of triangulation and not this; whereas, the advancing front method – it will take either this or this as it comes; it does not discriminate between the two possibilities. So, the overall triangulation that

we get with the Delaunay method has better aspect ratio, but it is not guaranteed to deal satisfactorily with concave surfaces. So, if there is a concave part to the overall floor domain, then we have to be careful with the Delaunay triangulation; whereas if we have severe concavity, then advancing front method is preferable in that way. So, we will look at the advancing front method and we will also look at Delaunay triangulation to get some idea how the triangulation of a nodalized domain can be done.

(Refer Slide Time: 39:37)



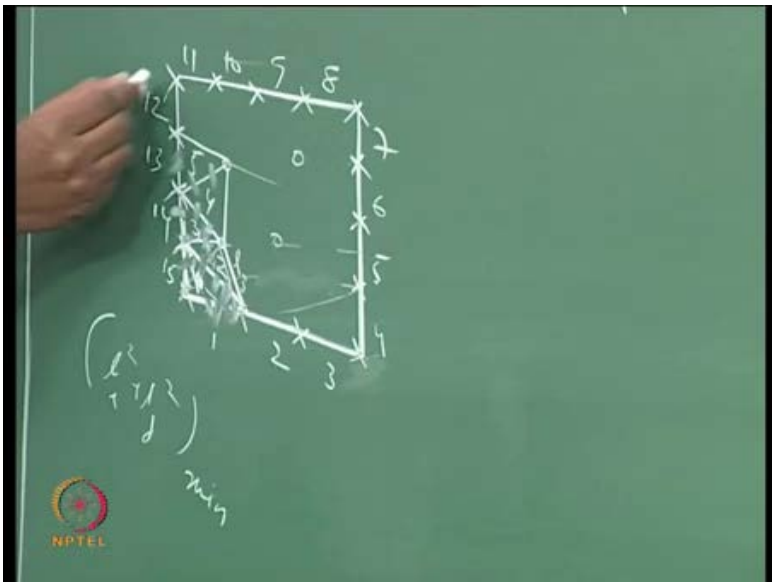
Now, we will take a much simpler case than this. Let us say that we are looking at a domain like this, a simple domain and which we have broken up into nodes; and, in the process of the nodalization, we not only have the boundary nodes, but we also have some interior nodes like this. Now, the idea is to join these things to make them into triangular elements, so that when we join all the triangular elements, we can get them together as one element. And, one can immediately see that if one were to do it randomly, this and this and this – these three can be joined together; or, this and this can be joined together; and, this and this can be joined together, which will play havoc with overall tiling. So, we have to come up with a rational way of doing this, so that that kind of overlapping is not permitted.

In the advancing front method, we start with the boundary that is there and the boundary consists of several edges. And, this is where we can start with 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 – 15 edges we have in this. And, these (Refer Slide Time: 41:03) are the

outer edges, are usually put in this counter clockwise way. And, we can start with for example, the first edge. The first edge could have been here, but we have started with this. So, this is the first edge. And, we take any point that is close. For example, we take this point here and then we say that this point should be joined with which edges in order to make a triangle. For example, should I join this and this – these three or these three like that? Which edges should it be joined? Of course, we cannot join with these three points because this point can be joined with only an edge, which has already two nodes. For example, I can join this with these two points or I can join with these two points or I can join with these two or these two like this.

Now, the idea is that which edge do I join with this? So, I start with this (Refer Slide Time: 42:07) edge here and I would like to choose that node, which is best joint with these two nodes in order to make a triangle. So, how do I do that? I start with this and then I am looking… I already know the counter clockwise direction. And, I look at all the possible nodes, which are to the left of me. So, that is, if I consider this; and, that all the possible nodes includes the interior nodes and the boundary nodes.

(Refer slide Time: 42:40)



I look at this point is on the… This is on the same edge. So, this is on the same thing. So, on the same line, which means that I cannot join these three to make a triangle, but I can join with these; I can make a triangle by joining these two. So, this is one possible triangle. So, this is one possible node to which I can join this. Again, this is another

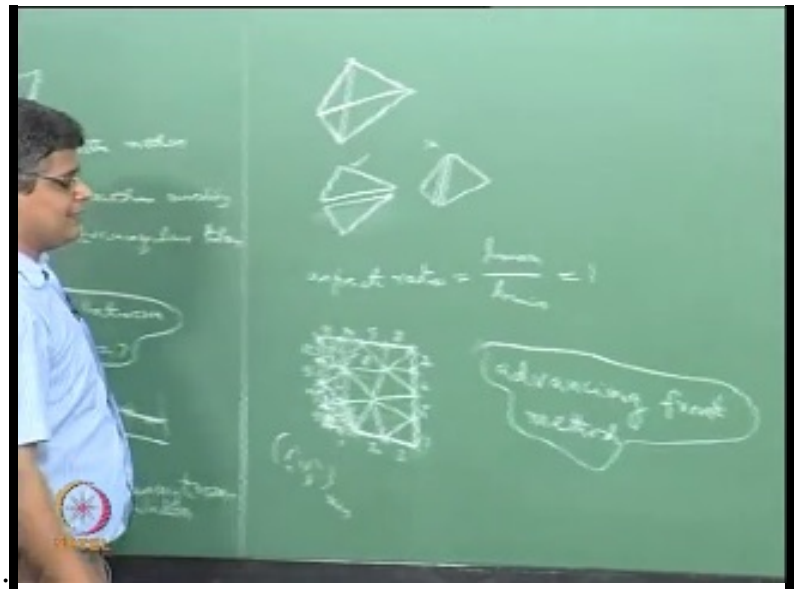
possible thing this. All these points here and these interior points are the possible nodes with which I can make a triangle. Now, out of all these possible nodes, I would like to take that node, which is nearest, which for example, if I join with this, then I have l 1 and l 2. And, if I join with this, I have l 3 and l 4. So, I take l i square plus l j square corresponding to each node and I see which combination will give the minimum value; so, the minimum length here. So, that node is the node with which I should like to join them, so that I can join with the closest node. So, from that point of view, if I take this length plus this length square, is obviously much greater than this length and this length. So, from that point of view, all these nodes and all these nodes will be out of contention and one can only seriously consider this node and this node. So, for these two nodes, I evaluate this length square plus this length square and this length square plus this length square. And, I can see that this node wins over and then I say that this is the node with which I should join these two points of the edge in order to make a triangle. So, I make a triangle like this. So, this is the triangle 1.

And, at this point, I say that the boundary for this domain consists of this (Refer Slide Time: 44:52) whole thing and this is not part of the boundary. So, at the end of making a triangle, I redefine the boundary in such a way that I have the overall continuity here. So, at this point I have made one triangle and I have deleted two edges here, which are no longer in contention. Now, a new front, which is advancing towards the best possible node to capture it and form a triangle. So, now, again I start with this and I look at all the points, which are to the left of me and then see with which I can join. And, I can see that almost all these points are to the left of me. And, of all these points, joining with this will give me the smallest triangle. So, I will join this. So, I made a second triangle here and again I take this out; this is now the front here.

I can start with this (Refer Slide Time: 45:53) and then look at all the possible nodes, which are to the left of me. So, if I take this, then all these points are to the left and these points are to the right. So, I do not consider this. And, I take all these points here and I choose which of them will give me the smallest triangle. So, obviously, this one. So, I join this; I make one more triangle and then take out these things. And then, I have my front, which is like this. Again, I take, look at all the possible things. And here now, I will join these two to make a triangle, the fourth triangle; I take it out; my advancing front is this. Start with this; and obviously, this should be joined to make the fifth

triangle; my front is like this. So, I can keep on going in each way. At each point, I look at all the possible nodes including the boundary nodes and the interior nodes, which are to the left of the edge that I am considering. And, all those qualified nodes are potential mates to be formed into a triangle. And, I take that point, that node, which gives me the least square of the lengths. So, that is l i square plus l j square. And then, use that to form the triangle.

(Refer Slide Time: 47:17)



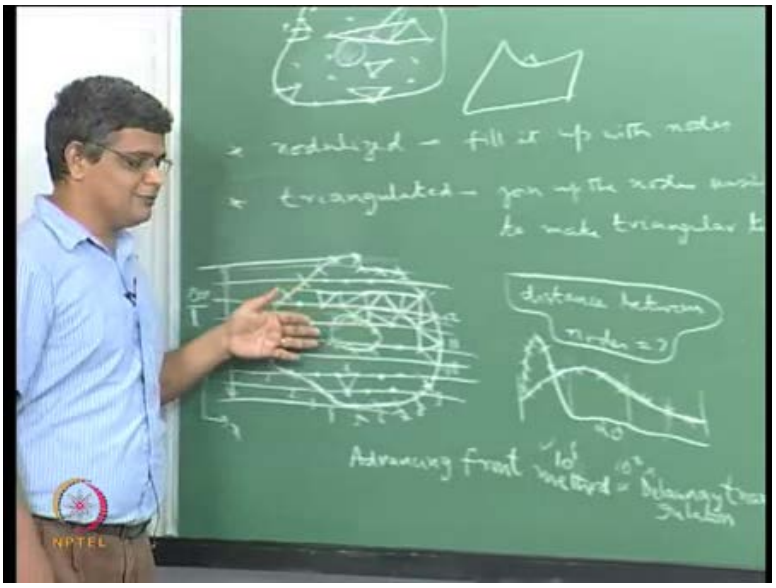
So, I join here; another triangle is formed; and then, I define the edge like this; and then, I can go on here again; and then, may be here. So, I can see that each time I am forming a triangle; and then, I can go on like this. Sometimes we may reach a dead end in which case we have to stop that. Dead end is possible when for example, you have already come here and you have formed a triangle and there is nothing else for us to go. So, in which case, I can go back to one more edge and then start with that, continue the process. So, in this way I can redefine my front which is trying to find, which is composed of boundary edge elements. The boundary is such that if you have some notion of the counter clockwise direction, and then for each edge, you try to find out what is the nearest qualified node with which to form a triangle. And then, we can continuously do this. And then, eventually, we will get all of them in decomposed triangles. So, this is what is known as the advancing front method, in which at any time, you have one front consisting of all the nodes on the boundary; and, the edges on this advancing front are to be joined with the available qualified nodes, which are to the left of it; and, by making

sure that we are always looking at the left side of it in the counter clockwise direction, we can eliminate the possibility of choosing points which lie outside the domain. So, that is how we are trying to deal with the concave geometry; that is, by maintaining directionality in defining the advancing front and also by looking for points which are to the left of this.

This progresses (Refer Slide Time: 50:00) in an element by element starting with some edge and then looking at what node is to be done at… So, the information that is needed to make an advance here is what are those nodes which form this boundary and the interior, which are to the left side of the current edge. And, once we do that, that can be looked at based on the cross product of this vector and this vector. So, in that way, we can find out. And, having decided all the possible nodes to the left of this edge, then we try to find out which of them is the nearest node by looking at the length l square – l 1 square and l 2 square for each node. And, that since we know the boundary points, that is, x i y j of all the nodes here, one can easily determine the lengths; and then, using that, we can select the most appropriate one and then go through this. So, this is a good way of progressing with triangulation after the nodalization done in this way.

And, this (Refer Slide Time: 51:21) ensures us that we will not take consider points which lie outside the domain, so that triangulation is done only from the interior points. And, by taking out these nodes as soon as a triangle is formed, then we can make sure that we go through the process until every node is associated with one triangle or other triangle, so that there is no node which is left unused and each node is associated into a tile in an appropriate way. So, the essence of the advancing front method is that it can deal with a complicated geometry in a systematic way and there is some sort of book keeping and all that thing. But, in terms of actual solution of equations, there are none here; we only have algebraic relations to find out which nodes are to the left and which nodes give us the smallest overall length.

(Refer Slide Time: 52:38)



So, the mathematics part of it is very small both in terms of triangulation and in terms of nodalization. But, the logical part of it, that is, to write a program to do all these checks and balances and all that is more tedious and one needs to be a good programmer in order to make this algorithm work.

In the next class, we will look at triangulation using Delaunay method, so that we can see what differences we have between advancing front method and the Delaunays method.