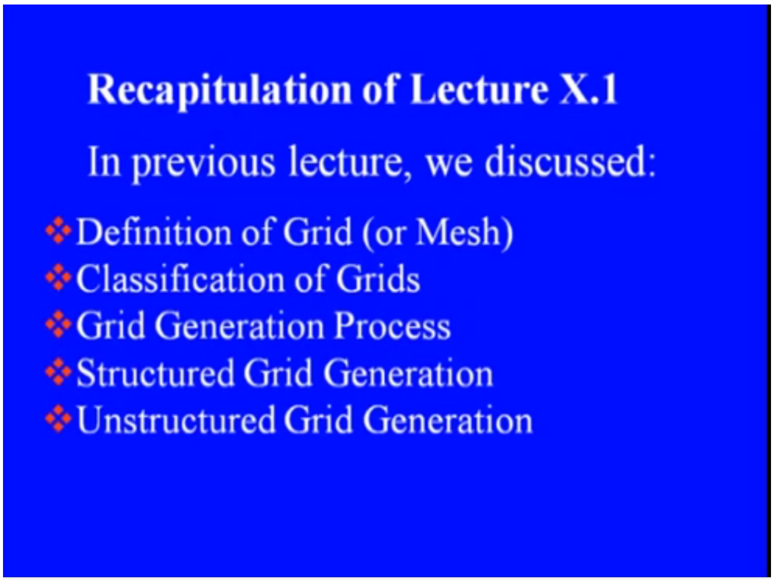
Computational Fluid Dynamics
Dr. Krishna M. Singh
Department of Mechanical and Industrial Engineering
Indian Institute of Technology - Roorkee

Lecture - 43
Aspects of Practical CFD Analysis

Welcome to the second lecture in module 10 on grid generation and aspects of real life CFD analysis. This is also a last lecture in the course. Let us have brief look at the module outline. We had discussed in the previous lecture brief introduction to grid generation and today we will have a look at the aspects, which are important in practical CFD analysis.

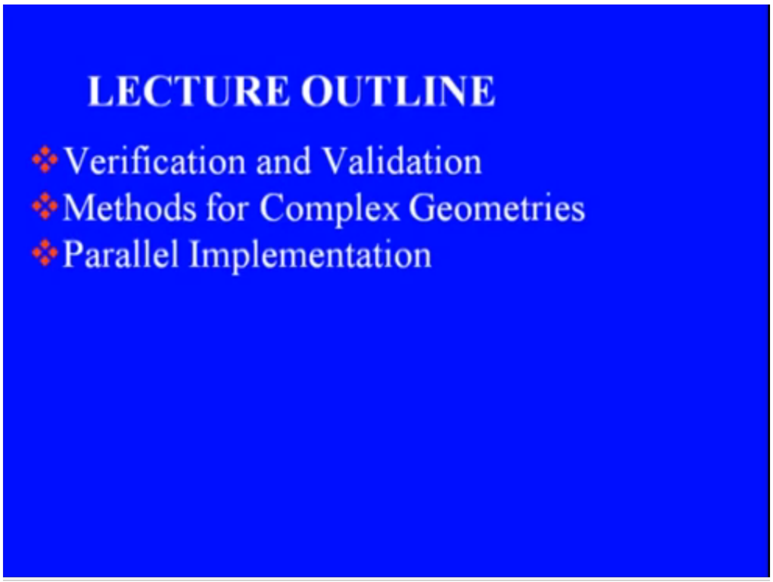
(Refer Slide Time: 00:51)



Let us have a recapitulation of what we did in previous lecture. We discussed formulae, the definition of grid, classification of grids and we had a look at the fundamental steps involved in grid generation process and then we discussed briefly about 2 primary grids and their generation that is structured grid generation processes and the methods for unstructured grid generation. So these are the topics which we briefly test upon in the previous lecture.

Further details you refer to the appropriate literature. In the second lecture in smart concluding module, we will look at some of the aspects, which are required or which we must keep in mind while applying CFD analysis for practical enduring analysis or research purposes. So we will primarily look at 2 aspects, the first one is what we call verification and the second one is validation.

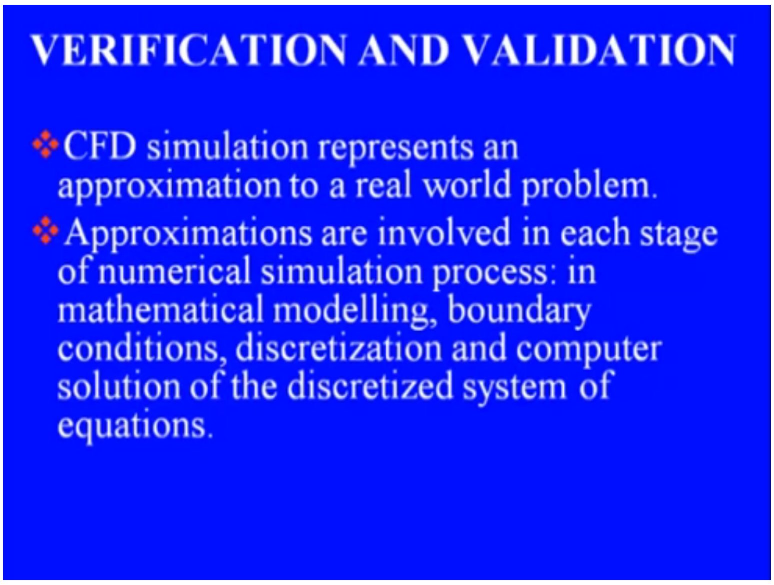
(Refer Slide Time: 02:06)



Then we will briefly touch upon the methods, which are used for complex geometry flows and the last topic which we are going to touch upon is the parallel implementation. So let us start with the first topic that is verification and validation. The CFD analysis whether by homegrown software or using commercial software would give us some numbers in response to the solution, which we sake for flow field variables.

Majority of the CFD solvers will give us these numbers and their beautiful presentation using various post-processing tools, but do these numbers mean anything or they any useful? And thus wherein this process of verification and validation is very, very important and let us remember that CFD simulation represents an approximation to a real world problem.

(Refer Slide Time: 03:09)



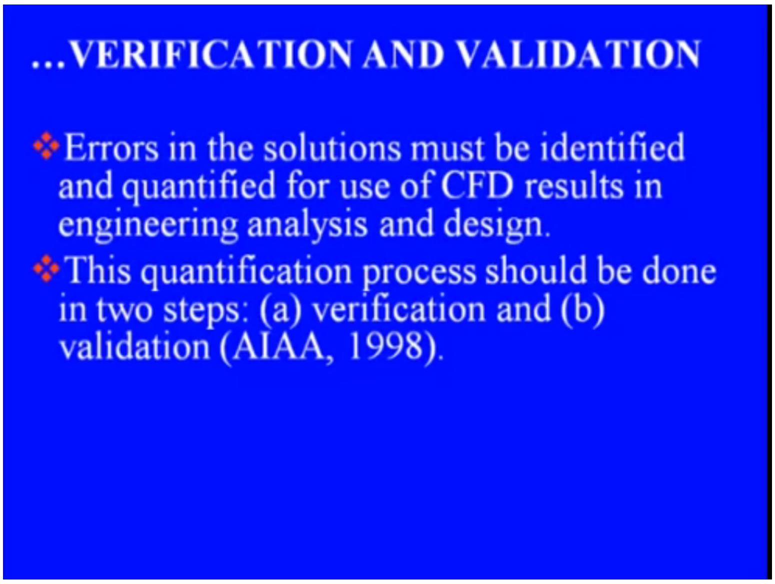
We have made different stages of CFD simulation many, many assumptions. So approximations are involved in each step of numerical simulation process. They approximate our constitutive models. We make certain assumptions and we derive our mathematical model. The boundary conditions also may not be represented exactly the way they occur in our physical system.

So this approximation involved in the boundary conditions. Then we have already seen that whether we choose finite difference, finite element or finite volume discretizing procedure, there is an approximation involved. This discretizing process yields a set of algebraic equations and these systems cannot be solved exactly either on the computer, which has got finite precision.

And moreover we would be working with most of the time with iterative solution processes so there are some errors of approximations involved in these iterative solution of discretized system equations. So CFD analysis must be aware of each of these approximations. They were able to interpret the results of analysis properly and these errors in the solution must be identified.

And we should be able to quantify them for use of CFD results in engineering analysis and design. We can say that what confidence we have in the numerical results or approximate solutions, which we have obtained. What are the error bounds?

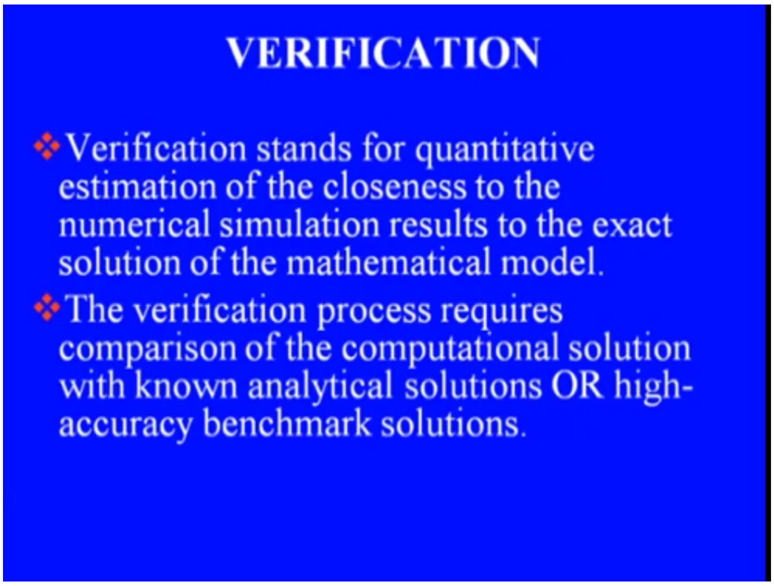
(Refer Slide Time: 04:45)



And we should ideally be able to get a quantitative bound on the errors in our numerical simulation. Only then we can fruitfully or confidently use these CFD simulation results in our design process. Now this quantification process should be done in 2 steps, the first is called verification and the second one is validation. For further details you can refer to an AIAA document published in 1998.

And more recently a similar document prepared in Europe called (()) (05:20). Now let us discuss what we mean by verification?

(Refer Slide Time: 05:45)



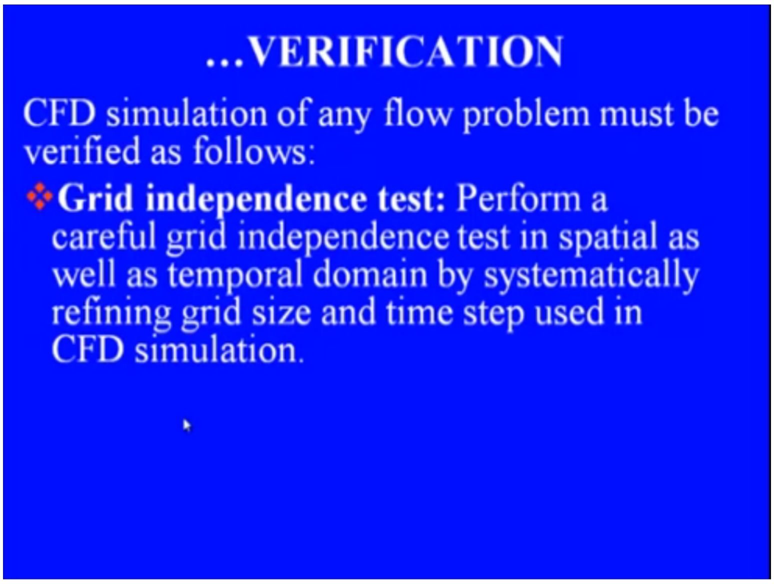
So verification stands for quantitative estimation of the closeness to the numerical simulation results to the exact solution of mathematical model. So in our CFD analysis what we did where first step was we obtained a mathematical model, which was in the form of partial differential equation or in the form of an integral equation in our integral formulation. Now then we applied a discretizing process, which had its own built-in approximation or errors involved there.

And then we solved it using an approximate numerical solution process. Now we would like to quantify and we would like to say that the final approximate solution, which we have obtained how close is it to the exact solution of the mathematical model if that were possible to obtain? So that is what is verification, to get a quantitative bound of the closeness of the CFD results to the exact mathematical solution.

Now this particular process would ideally require that we should be able to compare the computational solution or CFD solution with known analytical solution. In fact that is not going to happen most of the time specifically if you are trying to solve Navier-Stokes simulations in complex geometries, exact analytical solutions are simply ruled out. So in that situation our only recourse is what we call high-accuracy benchmark solutions, which have been obtained using specified numerical simulation process with very, very fine grids.

So there are 2 options for us if we have analytical solution to our problem or a simplified version, try first in that simplified version and see if a numerical simulation process matches with the analytical solution, which is known to us or then compare our solution with the high-accuracy benchmark solution to compute the verification and get a quantitative bound on the accuracy of CFD results.

(Refer Slide Time: 07:43)

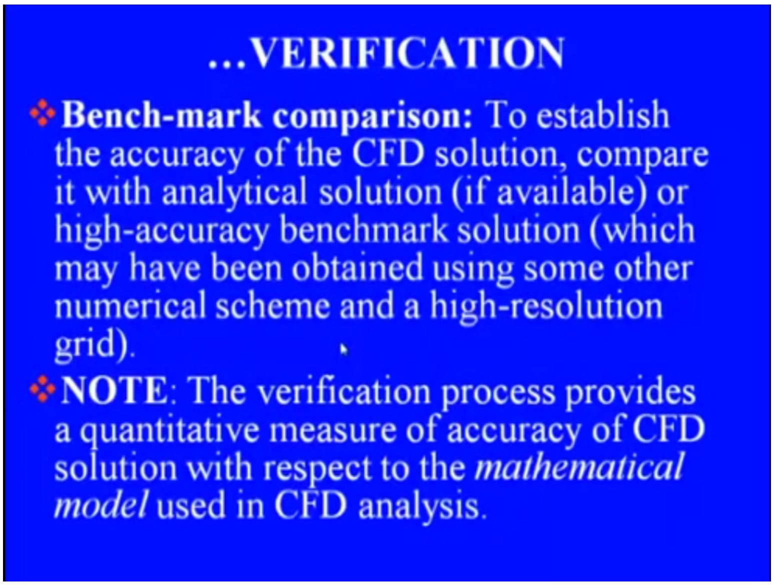


Now for this verification process, we have to follow few steps. The first step which we commonly use is what we call grid independence test that is to say the numerical grid, which we have used in a space or in time, our solutions would not be dependent on the choice of the grid. We should be able to get a numerical solution, which will not vary if we change our grid significantly.

So this is what is known as grid independence test. So what do we do? Perform a careful grid independence test in spatial as well as temporal domain by systematically refining grid size and time step used in CFD simulation and this step we take a sequence of refined grids, a coarse grid, a medium refined grid and a very fine grid. So we will take at least 3 grids in the space as similar step we have to perform in time.

And then obtain solution with each of these combinations and see if we are able to get a solution, which does not vary by refining our grid any further. So that solution would be what we call a grid independent solution. So this step would yield a grid independent solution and the next thing is comparison with the benchmark solution. This benchmark solution could be our analytical solution, which were available.

(Refer Slide Time: 09:13)



Or it could be a high-accuracy benchmark solution, which may have been obtained using some other numerical scheme and high resolution grid. So there are 2 fundamental steps involved in verification process, establish grid independence and the compare our grid independent solution to the high-accuracy analytical solution or numerical solution.

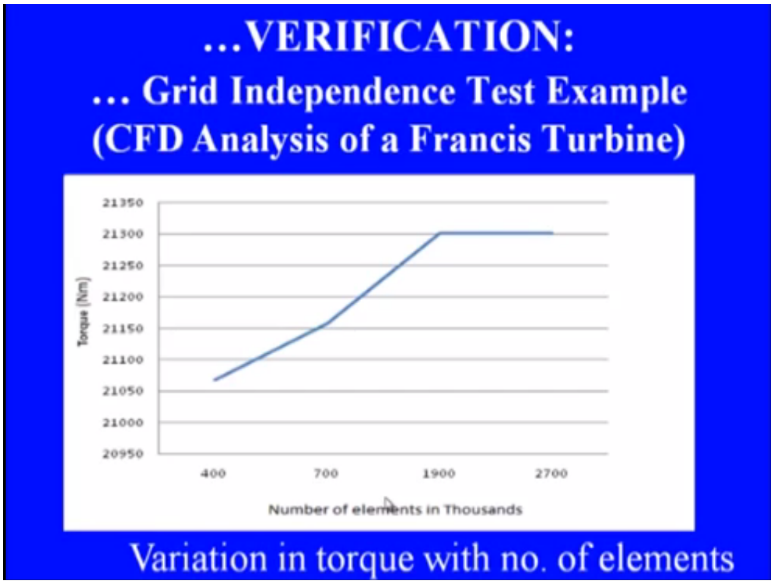
Now please note that this verification process provides us a quantitative measure of the accuracy of CFD solution with respect to the mathematical model, which we have used in CFD analysis. It does not give us an estimate of the accuracy in our real life problem. We have just established that how close the CFD solution is to the exact solution of our mathematical model.

So this is what we established through verification process. So let us have a look at one simple example, which we have done in our group.

(Refer Slide Time: 10:15)

CFD analysis of a Francis turbine, which is a hydraulic turbine. So let us have a look, this is the geometry and grid, it can give you some idea the way the grid was generated you can usually say the few sub blocks in which the geometry was decomposed before attempting to generate the grid. So there are separate grids generated in each of these sub domains and they were merged together to obtain the overall grid for this turbine.

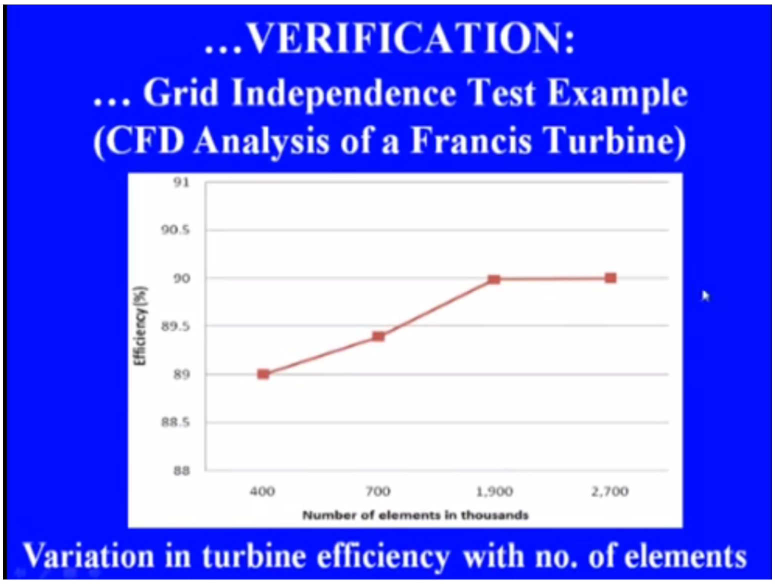
(Refer Slide Time: 10:51)



And then for verification purposes with there were sequence of grid generated. The coldest one was which involved 400,000 elements then 700, 1900 and 2700. So we have taken 4 grids and we obtained an estimate of the torque, which this turbine produces and the efficiency of the turbine. Now these 2 are the scalar parameters, which are obtained as integrated effect of the numerical solution.

So these are easy to use for the verification purpose to establish the grid independence. So what we can say if the grid size of 1.9 million or 2.7 million with both of these there is no change in the numerical result.

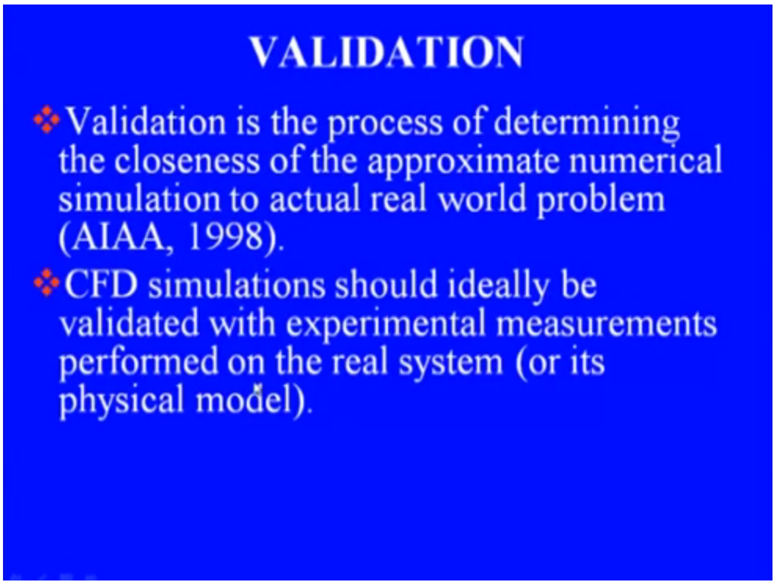
(Refer Slide Time: 11:47)



Similarly, we also plotted the results for these with these 4 grids for the hydraulic efficiency of the fastest turbine and once again we can say that whether we use 1.9 million grids or let us say close to 2 million grids or 2.7 million grids, the results do not change much. So that establishes that look of the solution, which we would obtain with either with these grids would be grid independent solution.

So this is typical way of performing a grid independent solution. Take a pretty coarse grid, fine out the solution, then take a finer grid, then still finer grid and very fine grid and in design a traditional cycle we would use either of these 2 finer grids, which have established the solution does not vary with use of either of them.

(Refer Slide Time: 12:37)

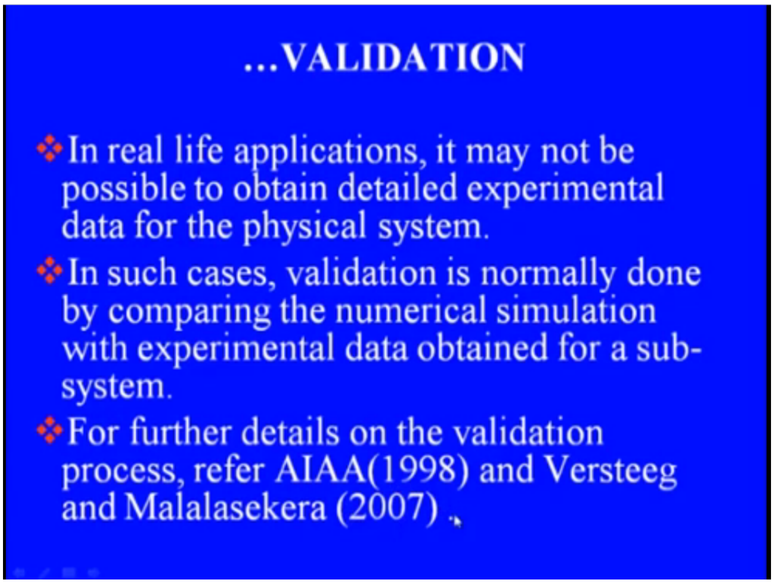


Next, let us have a look at the validation process and the validation is the process of determining the closeness of approximate numerical simulation to actual real world problem. We shall forget about further mathematical approximations were made. We want to see how close our CFD results are to the actual field problem. So this definition is what for the validation process has been proposed by AIAA in 1998.

And the CFD simulation should ideally be validated with experimental measurements, which are performed on real systems or its physical model, but this is the ideal situation. In real life, it may not be possible for us to perform high accuracy experiments on our real systems, some of the systems for instance we are performing or we using CFD analysis in the design stage, our real system still does not adjust this prototype is yet to be made.

So we cannot perform the experiments even if we had all the facilities in the world. Then what do we do here? So in actual real life application it may not be possible to obtain detailed experimental data for the physical system, which we have analyzed using our CFD simulation. Then what do we do?

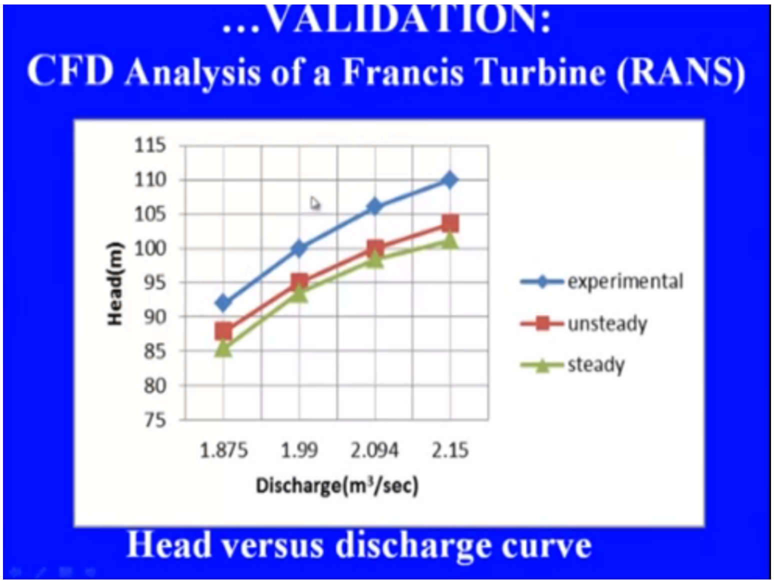
(Refer Slide Time: 13:58)



In such cases, the validation is normally done by comparing the numerical simulation with the experimental data obtained for a subsystem that is a small part of system. If we have experimental data for that, use it for validation purposes. Now further details of validation process please refer to the guidelines of AIAA published in 1998 and its summary you can also find in the CFD book by Versteeg and Malalasekara published in 2007.

Now let us have a look at few examples of validation process. The first one would be our CFD analysis of Francis turbine, which we had earlier seen in verification stage. So we had some experimental data available on the turbine provided by our additional partner.

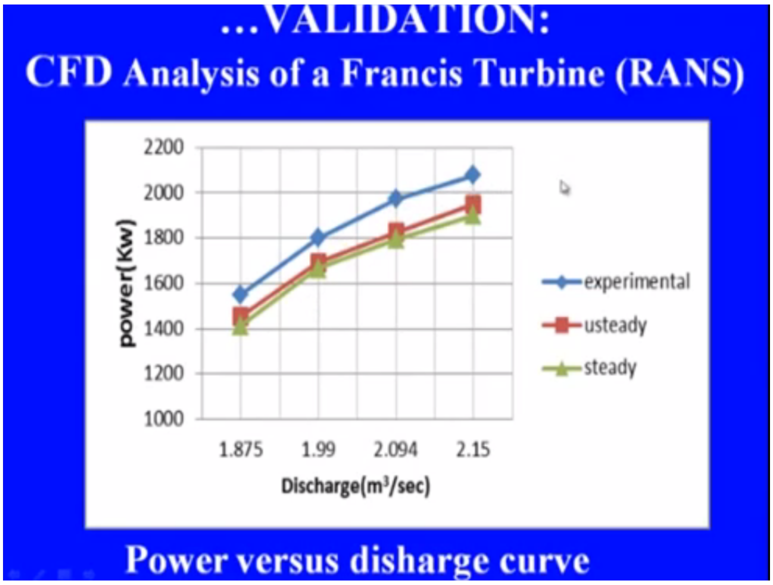
(Refer Slide Time: 14:46)



So the experimental data were available for head power generated and the efficiency. So the blue line here, this represent our experimental data and the red and green the 2 sets of numerical simulation results based on CFD analysis. The first one corresponds to what we call unsteady RANS. This is turbulent flow here and this Francis turbines we have used Reynolds averaged Navier-Stokes simulations.

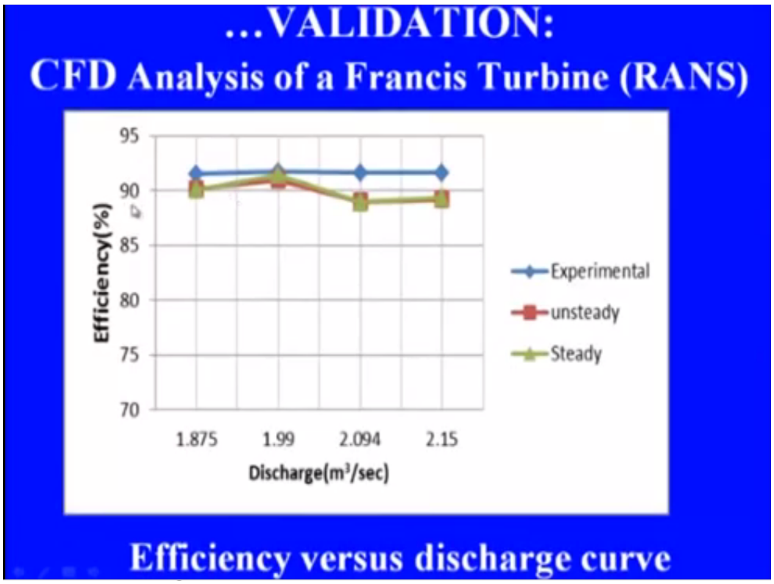
So red one is based on what we call unsteady RANS and the green ones are the results obtained using steady state RANS simulations. So you can easily see there is some difference but the trend is fairly similar to what is shown by the actual physical experiments.

(Refer Slide Time: 15:34)



The same trend could be seen with the power versus discharge core again where CFD results they give us lower values of the power generated compared to the actual experimental data. In a sense, it is a good thing that we are on our conservative side. Next about the efficiency, the experiments predict an efficiency of close to 92%.

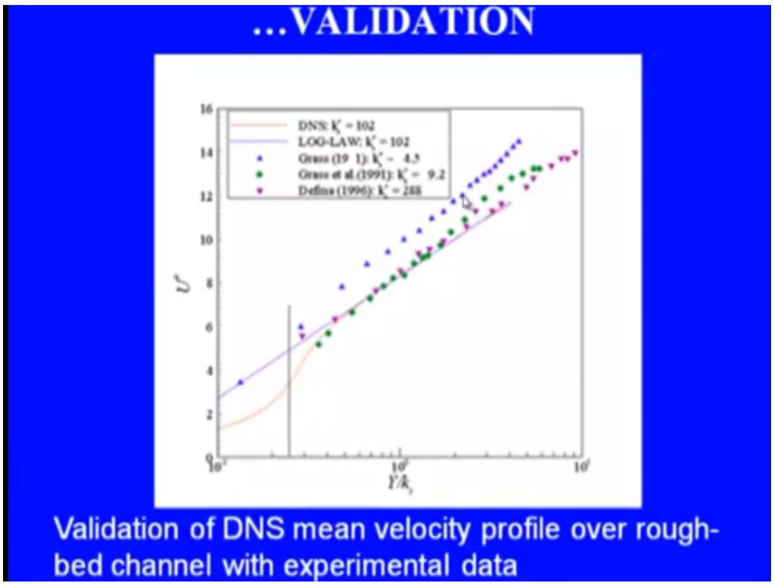
(Refer Slide Time: 15:58)



The CFD results give us an efficiency around 90% using both unsteady and steady state simulation where it gives us some confidence that we can go ahead and use our CFD results for further analysis or design modifications because we get fairly close estimate of the efficiency, powered generation or the head produced by the turbine and please remember even the experimental data may not be exact we did not have the bounds to plot here.

But remember this experimental data would also have certain uncertainties involved because of the errors in the measurement process. So the differences which we see here in the efficiency there would be termed as acceptable. Now next one is a very high accuracy DNS, a direct numerical simulation, which we had performed on a flow over a rough bed channel.

(Refer Slide Time: 16:55)



Now to validate our direct numerical simulation results, we had compared our numerical results with available experimental results of similar time. Here we have used a parameter, which is called reference Reynolds number. So our simulation was based on reference Reynolds number of 102 and we have taken few experimental results with the varying reference Reynolds number of 45 and 92 and 208.

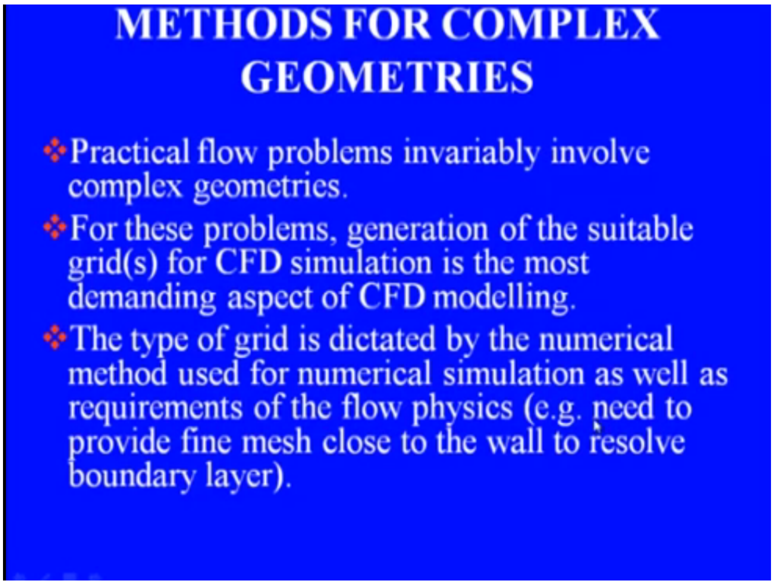
And we could is a similar trend okay not just that we had the log law of the wall for the same reference Reynolds number and we can say in the logarithmic region is very difficult to distinguish our DNS results with the log law and similarly in this region for the corresponding of the close values of KS+, the DNS and experimental results they are very close by together.

The one which is far off that was done long time ago in 1971 so it contains lots of errors. So this gives us confidence that our DNS is actually very accurate not just that. We can now use the DNS results, which we have got close to the reference elements. The solved line here vertical line tells us the top of the reference elements. So say that now the results which we have got close to the reference elements or inside the reference elements using DNS are close to the exact values, which we can expect.

No experiments can be performed here and even if we can use sudden probes they will give errors so that is why once we validate it our DNS results for (()) (18:45) rough surfaces then we can confidently use our DNS data for use as exact results for our problem. So we need to validate not just the RANS simulation results we have to validate the numerical simulation results obtained with large eddy simulation.

Or even with direct numerical simulation before we can confidently say that yes our DNS data is very, very accurate and it can be used as a very close approximation almost an exact solution of a flow problem.

(Refer Slide Time: 19:20)



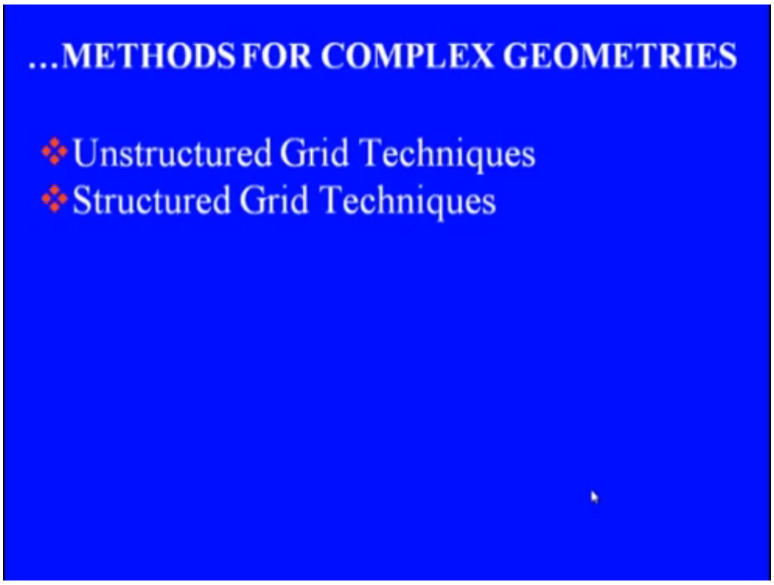
Our next topic which we are going to pick up today for the real life analysis is the methods for complex geometries, few of them we have already seen earlier in our grid generation process. This is more or less a recap of what we have done earlier and why we need to recap it or discuss it again is that the practical flow problems invariably involve complex geometries.

So we should be able to handle these geometries on numerical simulation process and for these problems the most important aspect is generation of suitable grid and we also discussed yesterday this is one of the mostly wanting aspect of CFD modeling in industry when we are modeling a fluid machines with complicated geometrical passages. So they are not just the CAD modeling it is the grid generation, which is the most time consuming process in our CFD simulation.

And we also discussed that type of grid, which we would use that is dictated by the numerical method used for numerical simulations. For instance whether we use finite difference, finite volume or finite element method that would dictate for type of grid we would use and it will also depend on the requirements of flow physics. For instance, a few got solid surfaces involved, we need to provide fine mesh close to the wall to resolve the boundary layer.

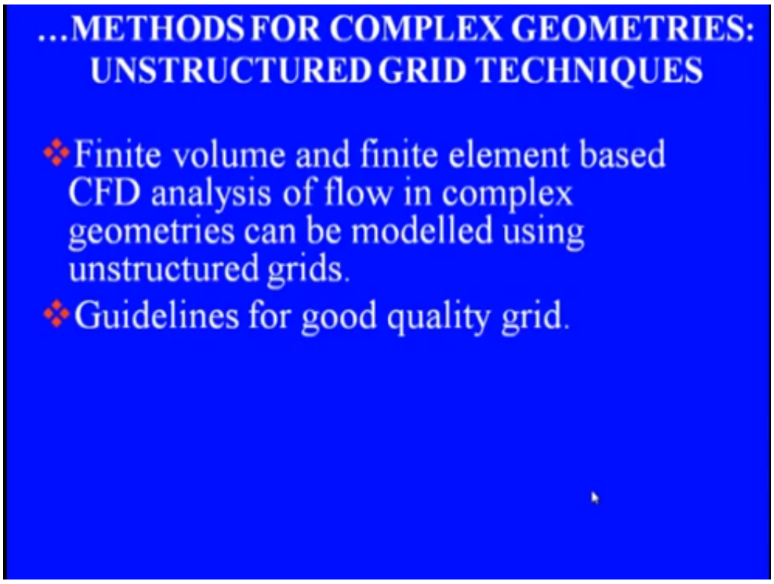
Say when you are performing RANS simulations for turbulent flows of walls, we have to provide enough number of grid points close to the wall to resolve the boundary layer properly. So grid design has to take care of this aspect as well.

(Refer Slide Time: 21:01)



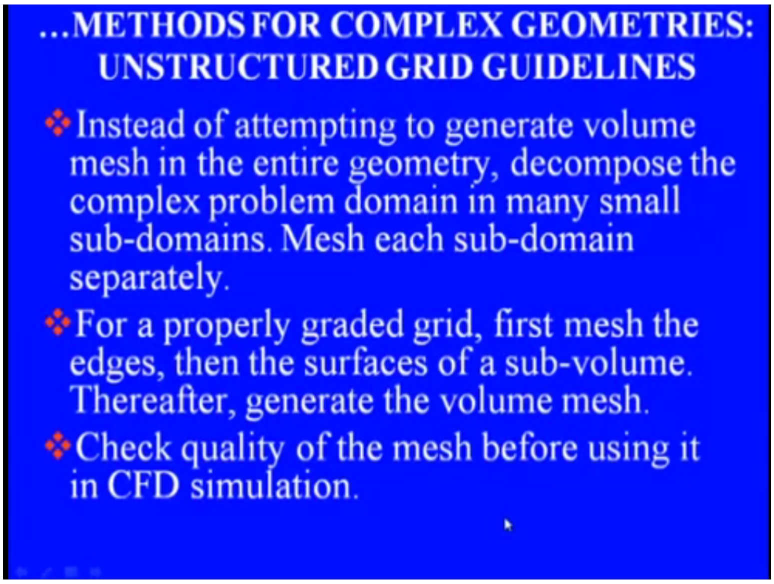
Now we can use both unstructured as well as structured grid techniques for modeling of a complex geometries. So let us have a brief look at both of these types once again.

(Refer Slide Time: 21:16)



Now unstructured grids we will primarily use in finite volume and finite element based CFD analysis and we have to follow certain guidelines to obtain good quality grid, which can ensure that we can get fairly accurate numerical solutions to a mathematical model. So let us have a look at these unstructured grid guidelines.

(Refer Slide Time: 21:40)



For a beginner, it would be very, very tempting specifically when you are working with commercial CFD software, which will have its own built-in modeling interface where you can generate a CAD model. It also provides you a mesh generator with variety of options. So there would be a temptation to just click on one of the buttons, which provides volume mesh and try to mesh the entire geometrically.

Now this would be what I would term as lazy bone approach. We would be able to get a grid but that would not be a good quality grid, which we would require for our numerical simulation process. So instead of attempting to generate the volume mesh in the entire geometry what we should do is we should decompose the complex problem domain in many small sub-domains and then mesh each sub-domain separately.

This will ensure that we get a properly graded and good quality mesh in each sub-domain. So that is the first guideline, which we must follow in real life CFD analysis and similarly to obtain a properly graded mesh by properly grade for instance if you want to resolve a boundary layer we must have very fine grid close to the surfaces and relatively cruder ones away from the walls this what we mean by grid gradation.

Similarly even to say we have got in compressible flow the presence of shock waves, the grids should be finer in those areas. So you have to provide appropriate gradation of the grid and one more aspect which is to take care this would not be abrupt jump in the mesh size. So the gradation against would be as smooth as possible to obtain accurate numerical results. So how we should proceed?

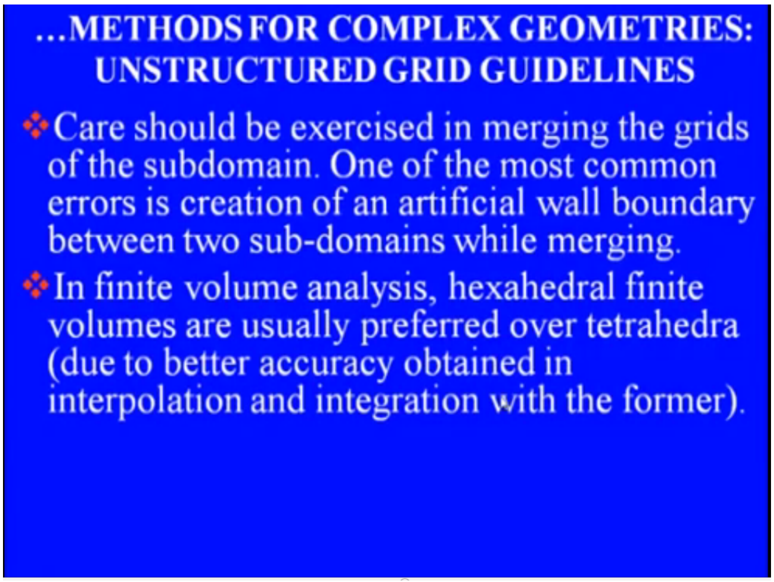
The first step would be that first generate grid on the edges of each sub-domain, then identify the surfaces of each sub volume, mesh that surface based on our measures or our requirements of the gradation and only when this process is complete then we should attempt filling the volume of the sub-domain with a grid or a mesh. So the steps would be grid the edges, from edges go to the surfaces.

And then starting from the surfaces obtain a volume mesh. So both of these steps or guidelines are very important. Decomposition in sub-domains and then a stepwise procedure for meshing each sub-domain and before we proceed further to merge all the sub-domain grids, we should check quality of the mesh in each sub-domain whether it is suitable for the CFD simulation, whether the angles of different elements are within proper ranges or not.

So only we have to check the grid quality of each sub-domain we should proceed further with the merging them. Otherwise once you have merged it and then we obtain the grid wall quality estimates it would be difficult to re-grid and remerge the things again. In fact, we will have to do then have an issue whereas if you have done the grid quality check at each sub volume or sub domain we need to focus on only those sub-domains where we have not got a good quality mesh, refine it or amend it and then combine all the sub-domains together.

And this is a very important guideline that we have to exercise at most care in merging the grids of the sub-domain. In fact, the care should have been exercised when we generate it our sub-domains by using what we call artificial surfaces. So there are certain options which have to be chosen properly in commercial grid generation packages.

(Refer Slide Time: 25:57)

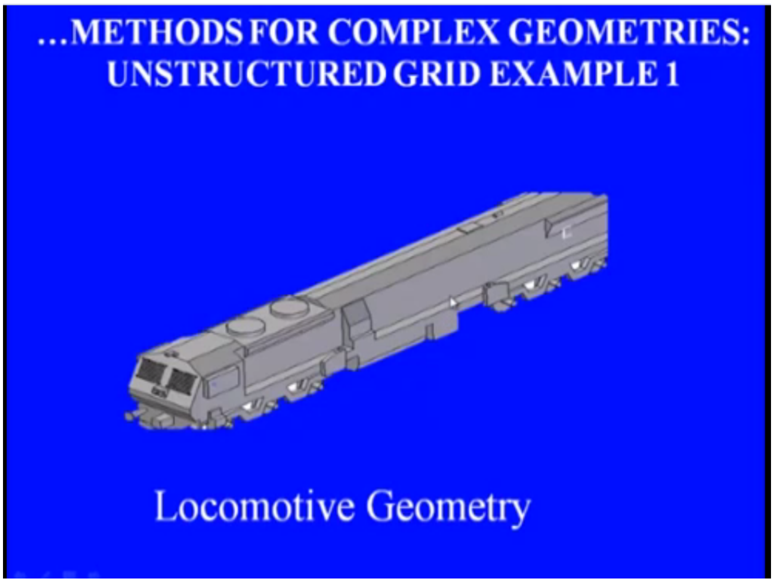


Now one of the most common errors, which users make is creation of an artificial wall boundary between 2 sub-domains while merging. So before we do that let us take care of this aspect that once 2 sub-domains are merged they would be merged smoothly without creation of any artificial separating surfaces between 2 sub-domains.

When we were discussing finite volume methods, we saw that if you are dealing with quadrilaterals or hexahedral finite volumes it is easier to obtain better accuracy in interpolation as well as integration so that is why in finite volume analysis hexahedral finite volumes are usually preferred over tetrahedral ones due to better accuracy obtained in interpolation and integration with the hexahedral elements.

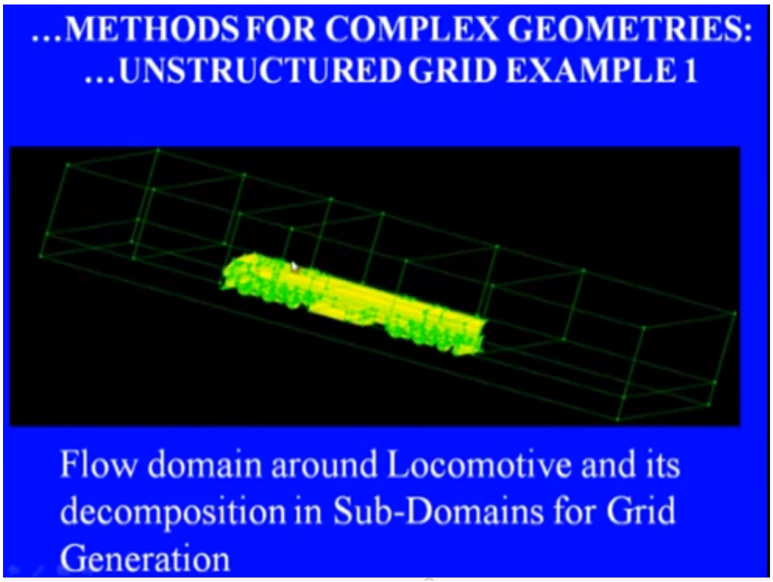
So we should try to use as much of these elements as possible. So what we would normally do is that we would reserve tetrahedral elements, we will use them, this would be confined to the regions too complex to mesh using hexahedral elements elsewhere use hexahedral elements. To give you an example let us have a look at 2 examples of unstructured grid generation one which was performed on a locomotive model.

(Refer Slide Time: 27:15)



It is very high speed train model, which was given to us by Indian Railways. We have made a CFD model. This is CAD model of the engine and you can see the various protrusions involved. There are very fine geometric features, which would make the process of grid generation fairly difficult. So now this is just the solid surface, which is provided by the CAD model.

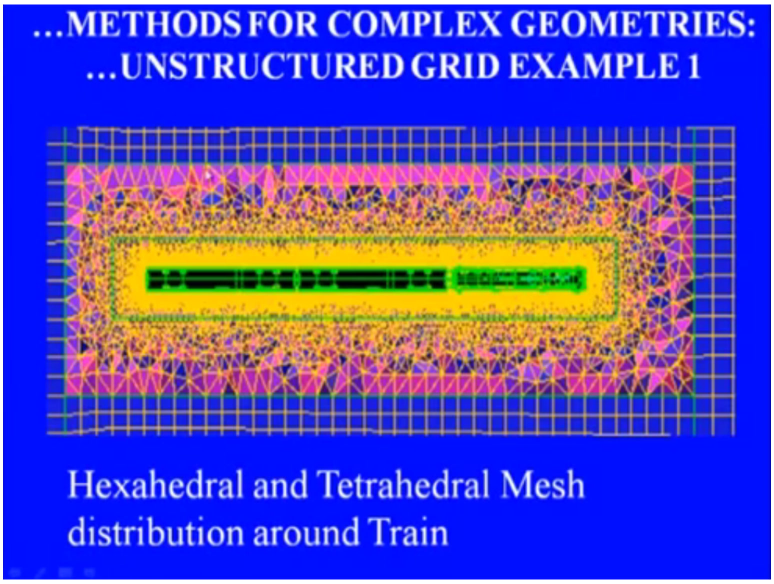
(Refer Slide Time: 27:52)



Our actual solution domain is outside it and how do we generate grid for this problem? So this is a computational mesh, the outline here but the green line tells us what we call so called computational domain. The flow inlet and the flow outlet here, there are side boundaries and this bright yellow green is our surface of the locomotive. You can clearly see that we have subdivided our computational domain into many, many different sub-domains.

And different types of meshes were generated over each of these sub-domains and then they were merged together. Now this is one possible way of decomposition in fact we should decompose it in even more number of sub-domains for updater quality grid.

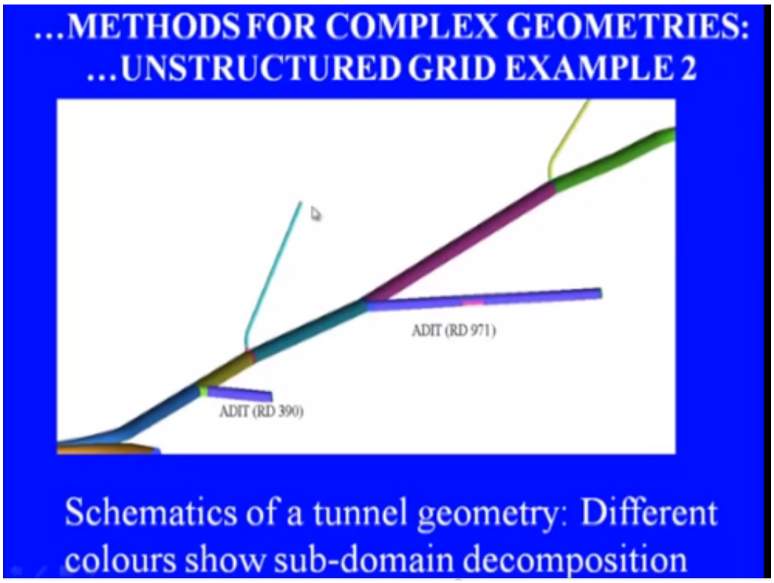
(Refer Slide Time: 28:34)



And this is one snapshot of the grid, which has been obtained, it is rather a cross-sectional snapshot of the grid, which consists of the hexahedral elements away from the locomotive body and the tetrahedral elements close to it because close to it we had so many protrusions and there generation of hexahedral elements would have been pretty difficult. So that is why close to the boundaries in those sub-domains we have got tetrahedral grids and wedge elements.

And in these sub-domains which are far away from the solid boundaries we have used hexahedral grid.

(Refer Slide Time: 29:11)

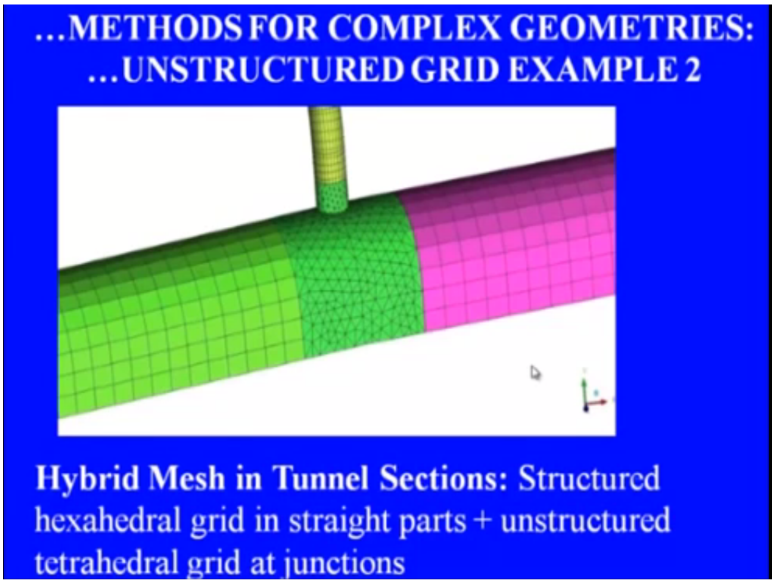


Let us have a look at one more example. This corresponds to a highway tunnel. So this is our long highway tunnel and this has got many adits and these are what we call shafts for the outlet air. The air is suck from there and discharged to the outer atmosphere and these adits are for what we call additional ducts through which we can access the tunnel during emergency situation.

They are also used to supply the fresh air inside the tunnel. So this is our geometry of the tunnel and different colors which you can say here each one represents decomposition of sub-domain. So each color stands one sub-domain so this whole tunnel we have divided into many, many, many different sub-domains 1, 2, 3, 4, you can count, there are more than 10 different sub-domains depending on what type of features we had at that point.

And when we generate the mesh if you are dealing with the straight tunnel portions, we can go for structured hexahedral generation, but when we come to the junction points here that is impossible for us to generate structured mesh or to have a continuation of the structured mesh from both the sides. So these are areas where we would go for what we call unstructured tetrahedral meshes.

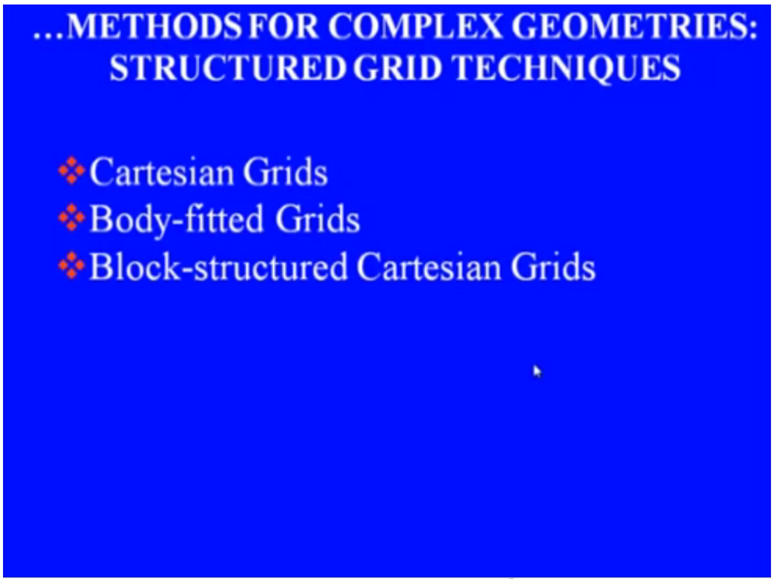
(Refer Slide Time: 30:40)



So this is the view of the mesh, which had been generated, this is what would call hybrid mesh that we have got a hybrid of structure, mesh on 2 sides and this outer junctions, so this is shaft, the tunnel body. Both the sides we have got hexahedral structured meshes and at junction point we have used tetrahedral unstructured meshes. So such things are called what we call hybrid meshes.

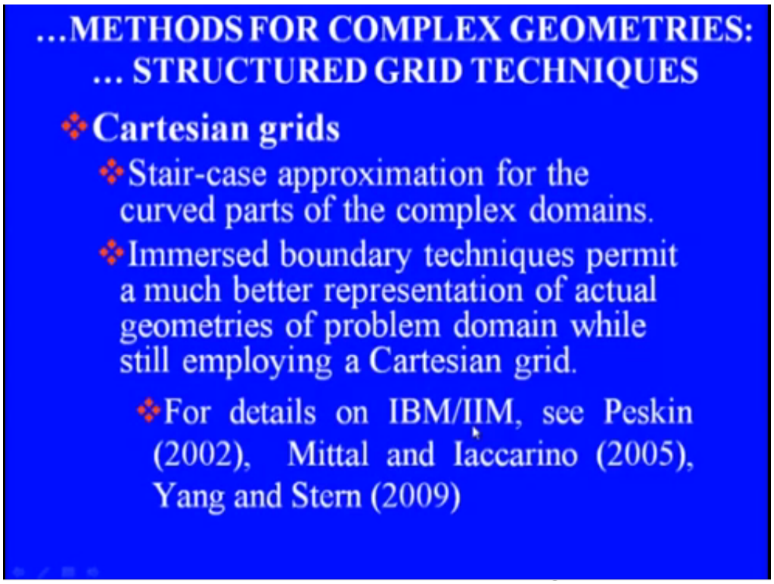
So these 2 examples, which we had a fairly complex geometries they give us a fair idea that how we should proceed about grid generation process, break our geometry overall problem domain in as many sub-domains as required. In fluid machinery even more complications would be there. So we have to break into maybe 100s of sub-domains and then generate appropriate mesh in each sub-domain and merge them together for final analysis.

(Refer Slide Time: 31:38)



Now let us have a brief look at on this structured grid techniques. There are 3 structured grids, which we discussed yesterday. This Cartesian grids, body-fitted grids and block-structured Cartesian grids.

(Refer Slide Time: 31:50)



Now when we want to use simple Cartesian grids in the context of finite difference or finite volume analysis to represent the curved portions or curved surfaces of the problem domain, we would use what is called as stair-case approximation. Now this stair-case approximation this leads to very crude approximation of the geometric features of the problem domain and it might also pollute our solution.

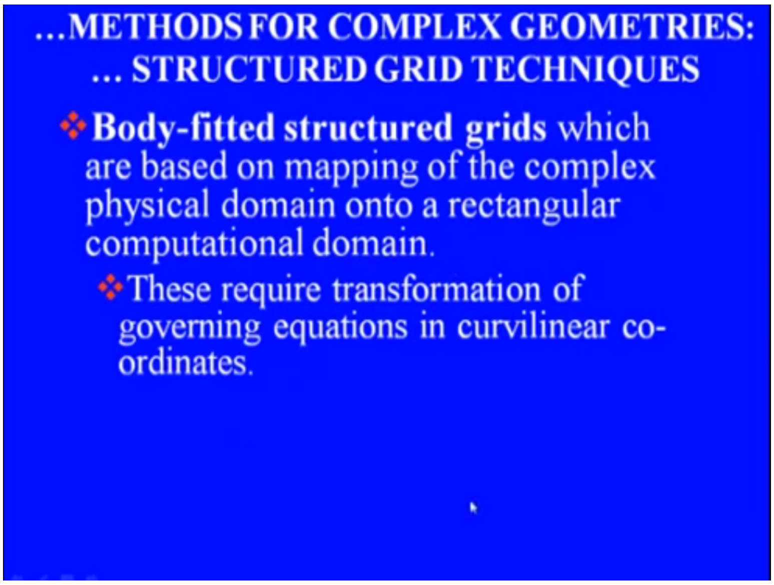
And recently immersed boundary techniques have become very popular in the context of Cartesian grids though these techniques were invented long, long ago by Peskin, but recently with advent of parallel computers wherein use of Cartesian grids is very favorable for load balancing purposes. Immersed boundary techniques have become now the flavor of the day and lots of research work is being currently done on various types of immersed boundary techniques, which can be used in different context.

So these immersed boundary techniques, they are based on Cartesian grids, but they permit a much better representation of actual geometry of the problem. We can have almost exact representation of the problem geometry using what we call immersed interface techniques. So though our CFD formulation is based on a Cartesian grid, we will introduce certain features or certain forcing terms, which can give the effect of the presence of curved geometric features exactly the way they were.

So programming is bit complicated, but then we can obtain very accurate solutions using immersed boundary techniques, not just that using immersed boundary techniques we can also exploit all the solution techniques, which are developed for Cartesian grids. So further details of immersed boundary method are more recently relative called immersed interface method.

See Peskin, Mittal and Iaccarino and Yang and Stern, I have listed these references in full towards the end of the lecture.

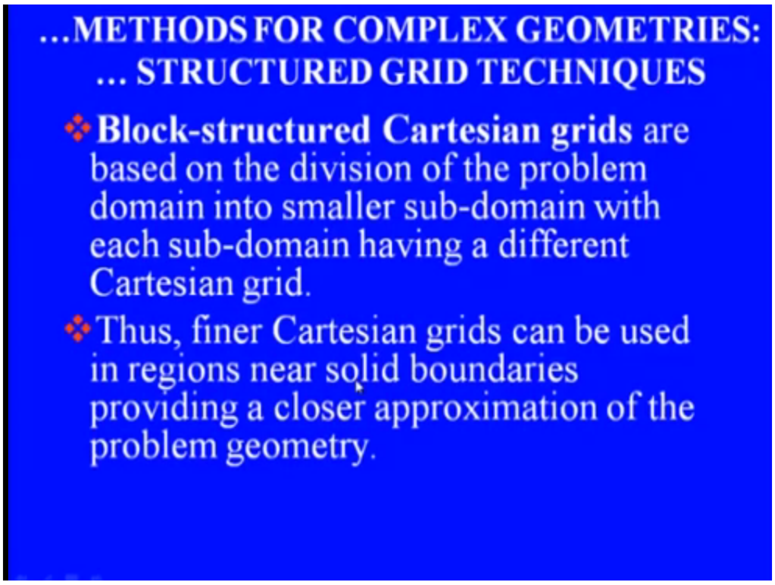
(Refer Slide Time: 34:02)



Body-fitted structured grids we have already seen that we can obtain mapping for complex physical domain on to a rectangular computational domain and we need to solve a system of partial differential equation to obtain our grid. So these were very popular in the beginning, but right now it has given way to Cartesian grids or blocked structured grids.

Because the transformation, which we require would be very difficult to obtain if we attempt to generate a body-fitted grid for a big complex problem say for instance of turbine geometry.

(Refer Slide Time: 34:47)

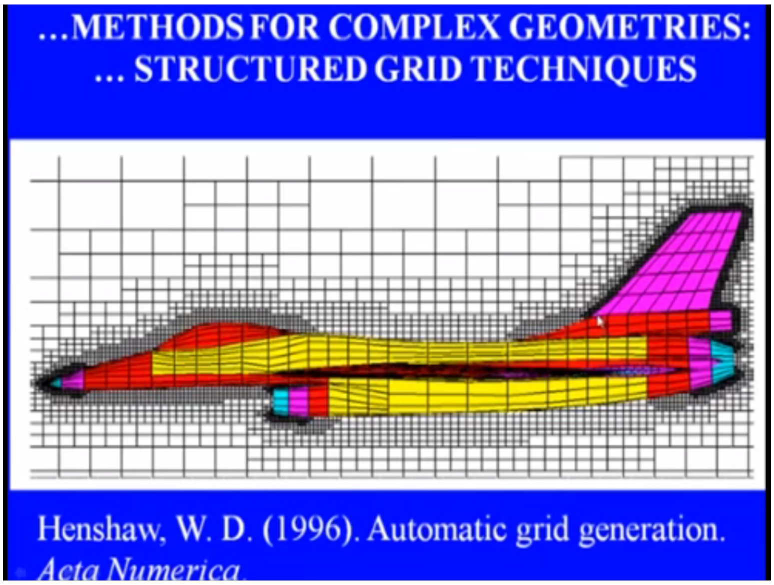


So the next thing is what we can do is instead of trying to fit in a body-fitted grid for a complex geometry at one go, we can use decomposition sub-domains and we can have a structured grid on each of those sub-domains. Now each sub-domain can have different structured grid. We can have a Cartesian grid on one sub-domain and body-fitted grid in another sub-domain. The choice is ours depending on the geometry of the problem.

It is also possible for us to have Cartesian grids everywhere. So this particular technique is called blocked-structured Cartesian grids. They are the flavor of the day very recently. In context of finite difference and finite volume based CFD analysis wherein we can have smallest sub-domains with each sub-domain having a different Cartesian grid and we can have very fine Cartesian grids in the regions near solid boundaries.

Or where we have got discontinuities of shockwaves to provide a very close approximation of the problem geometry or the sharp gradients, which are present in the physical solution.

(Refer Slide Time: 35:55)

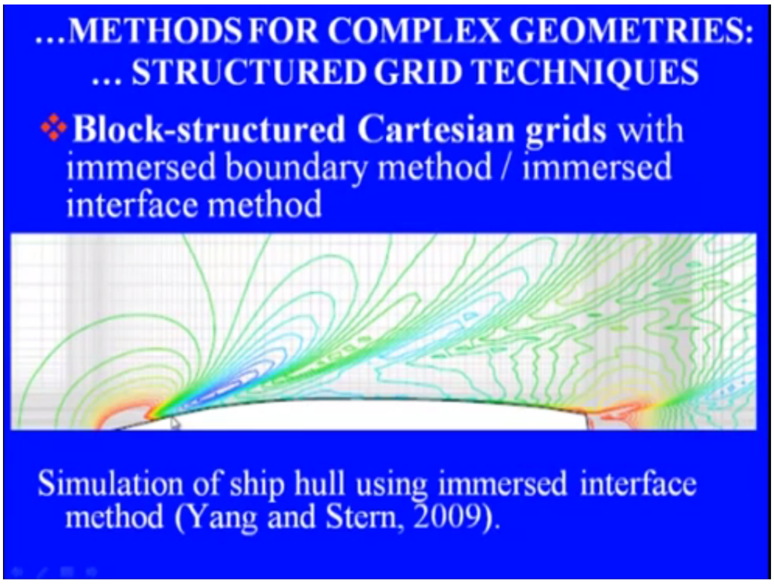


This one typical example taken from Henshaw in paper on automatic grid generation published on Acta Numerica. So this is a full body aircraft and this is the one cross-section of it. It is a fighter aircraft and you can say blocked-structured Cartesian grids close to the surfaces in different parts, we have got the grids of various or different densities (()) (36:13) we have got very fine grid.

Similarly they are (()) (36:15) we have got very fine grids and so on close to the wing surfaces or this tail surface again we have got very fine grids and away from the body of the aircraft, we have got relatively coarser grid. So this is what is called structured or rather blocked-structured Cartesian grid around a complete aircraft. Such grids are very easy to generate. This is also very easy for us to divide our problem domain into small sub-domain.

And map each of this sub-domain depending on the number of nodes we have got in that and the load balancing on the parallel machine also becomes pretty straight forward while using blocked-structured Cartesian grids and this is one of the reasons why recently it has become very, very popular to use blocked-structured Cartesian grids along with immersed boundary techniques, which can help us to take care of the curved boundary surfaces with almost 100% procedure.

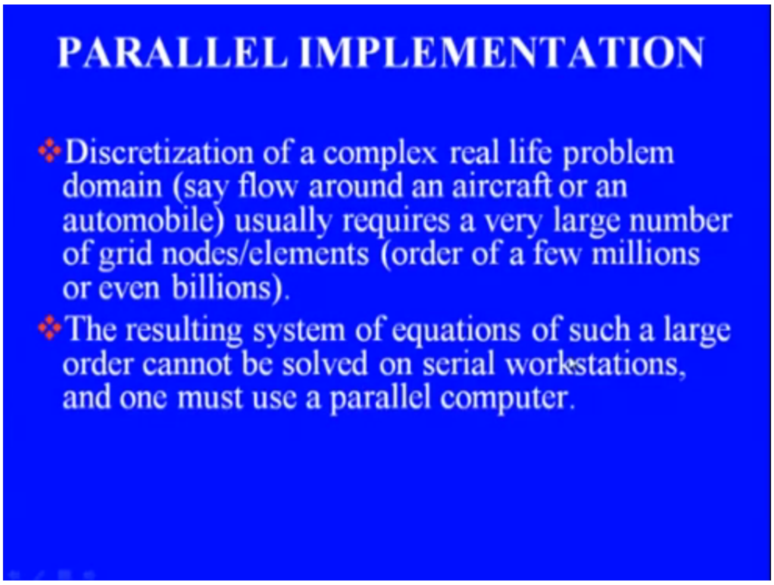
(Refer Slide Time: 37:27)



Yet another example of blocked structured Cartesian grid, which has been used for simulation of a ship hull using immersed interface method by Yang and Stern. So once again you can say this background the grid is there with results in there. We have got Cartesian grids of different densities and closed surfaces to take care of the presence of the solid boundaries immersed interface method has been used, which can give us exact representation or it can account exactly the presence of the coarse surface of the ship hull.

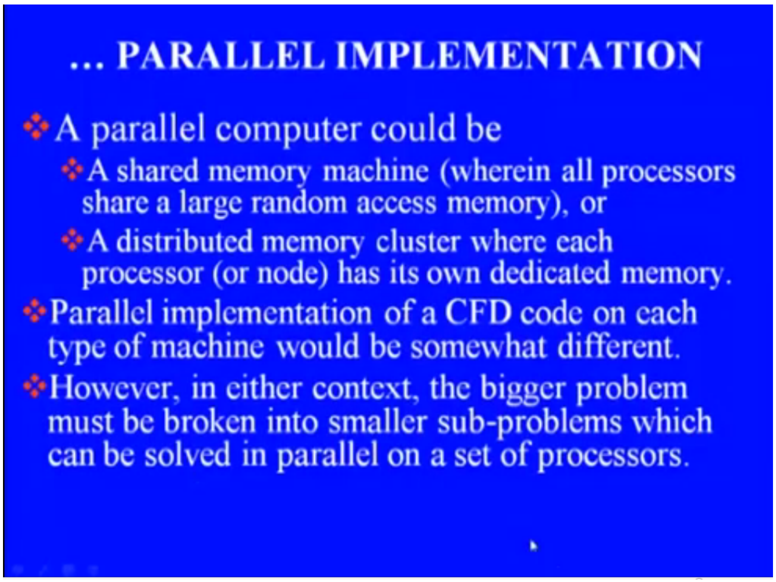
The next topic which will briefly touch upon today is the parallel implementation. This is one of the most important aspects in real life CFD analysis and why to become very clear that if you want to discretize your complex real life domain say the flow down the aircraft a turbine or an automobile. This would require a very large number of grid nodes or elements the minimum which we have in RANS simulation as few millions.

(Refer Slide Time: 38:47)



In fact for a flow of aircraft a typical number of the nodes would be in many billions. Now such large size problems, they cannot be solved on the serial work issues and no matter how powerful they are because we would be dealing with very large order of algebraic equations okay. So in such situations we must use a parallel computer, which is typically referred to as HPC in short as high performance parallel clusters.

(Refer Slide Time: 39:18)



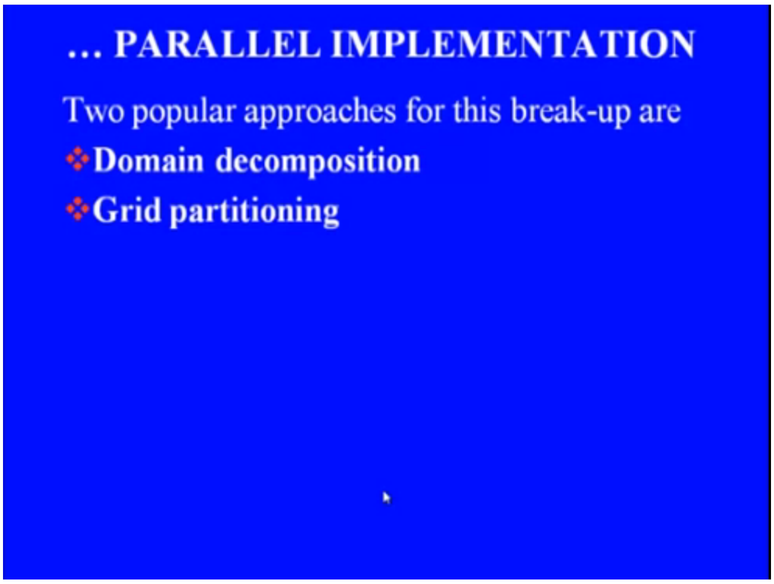
Now a parallel computer could be a shared memory machine wherein all processor share a large random access memory or rather an image of it which is accessible to each processor on that parallel machine since they are called SMD clusters or we can have a distributed memory cluster where each processor or node has its own dedicated memory and these clusters we must communicate data between themselves.

Now parallel implementation of a CFD code on each type of machine would be somewhat different since it would be slightly different for a shared memory machine than on distributed memory machine because of the memory sharing. In the case of shared memory machine, the message-passing would be very small almost negligible wherein distributed memory machine there must be separate message-passing interface, which is to be provided.

Because the data at the boundary of each sub-domain, which is mapped to one node that has to be shared to the other node using message-passing so that would be one of the differences between the implementation on these 2 types of machines. However, in both the cases we have to do one thing that is we have to break our big problem into small problems.

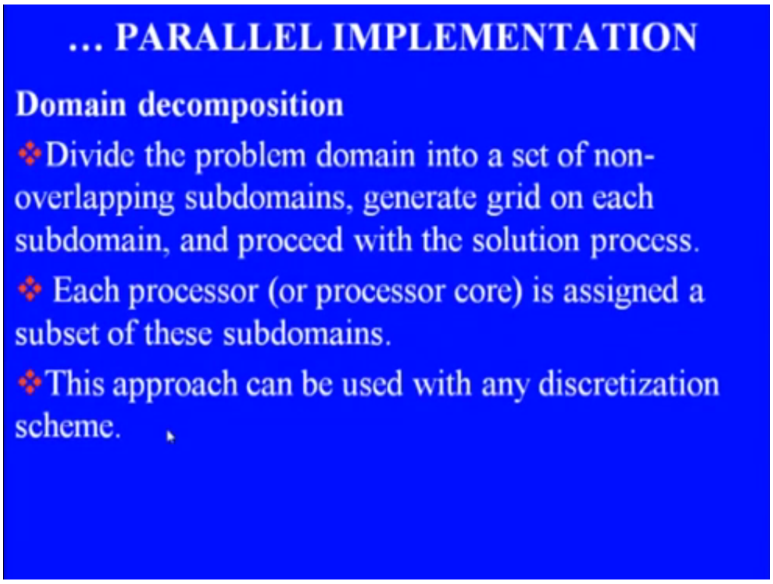
See in other context that is in the context of SMD or distributed memory clusters, the bigger problem must be broken into smaller sub problems, which can be solved in parallel on a set of processors. Now how do you break it? That is the next aspect, which we must look into. The 2 popular approaches for this breakup or big problem into small problem, the first one is called domain decomposition.

(Refer Slide Time: 40:56)



That is we have got big complex problem domain the way we did for our grid generation, use a similar sort of decomposition, decompose your problem into smaller sub-domains and then try to map each sub-domain to each processor and second one is what we call grid partitioning that is we generate a grid somehow and then like this grid is partitioned using certain graph partitioning tools.

(Refer Slide Time: 41:21)



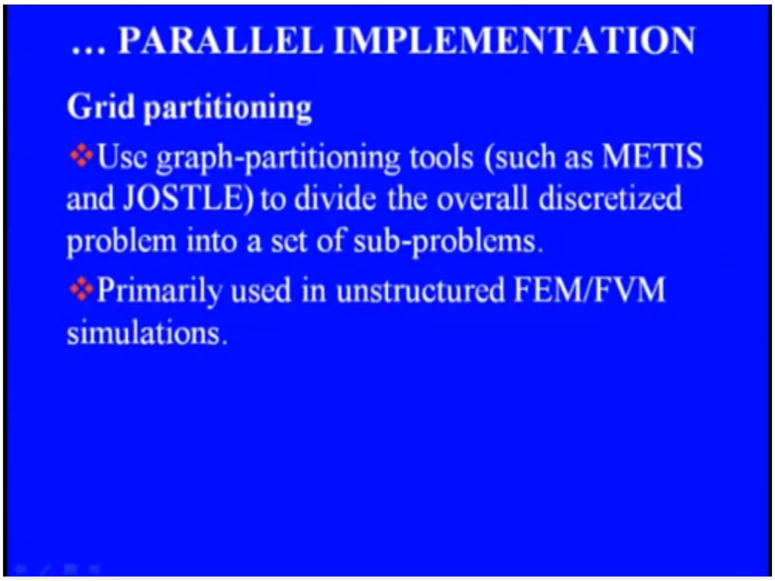
Now domain decomposition approach we divide the problem domain into a set of non-overlapping sub-domains. Generate grid on each sub-domain and proceed with the solution process, each sub-domain would be mapped to a separate processor, it is to happen if you got so many number of sub-domains maybe many sub-domains are mapped to a single processor that is possible.

So depending on the number of processors, which are available on our machine we can decide that how many sub-domains we want to divide our problem into or we can divide a problem into let us say 1000s of sub-domains and then depending on the number of nodes or processors available distribute a given number of sub-domains on to each processor or processor core.

So this is what we have got a set of non-overlapping sub-domains. Each processor of processor core like recently we have got multicore machines. Each processor might have 10 to 16 cores okay. So each processor core behaves like a separate processor so each processor core or processor is assigned a subset of these sub-domains, which we have obtained in domain decomposition process.

Now this domain decomposition process is very general. We can use it with finite element, finite difference or finite volume discretizing scheme with structured grid or unstructured grid.

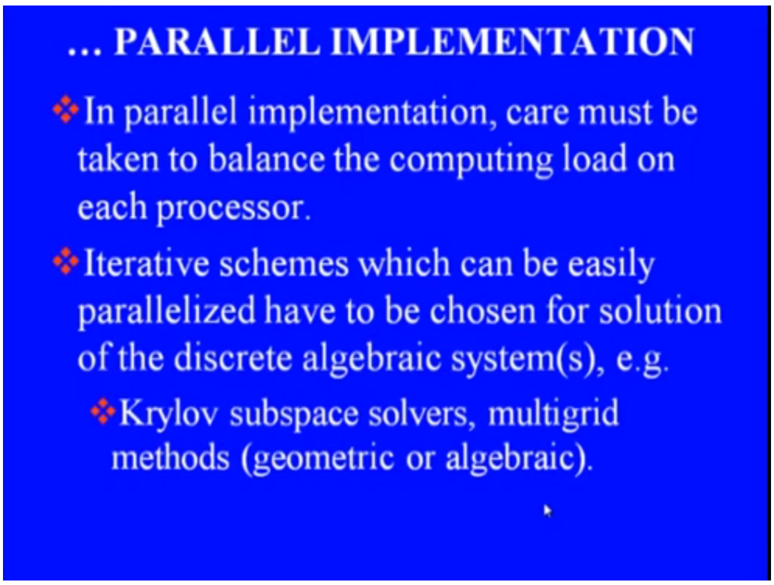
(Refer Slide Time: 42:57)



The next one is our grid partitioning approach. Now here people use graph-partitioning tools to popular field available tools are METIS and JOSTLE. The differences are provided at the end of the lecture. These graph-partitioned tools are used to divide the overall discretized problem into a set of sub problems. By overall discretized problem we mean the total grid which we have obtained.

That overall grid is broken to a subset of small, small grids, which can be mapped to separate processors for Helyx. Now this grid partitioning approach is primarily used in unstructured finite element or finite volume simulations. In the case of finite differences or structured finite volume or finite element formulations, it is better to use domain decomposition method for decomposing a problem into smaller sub problems.

(Refer Slide Time: 43:58)



And there is one more aspect, which we must carefully take care of is we have divided a problem domain into small, small sub-domains and this certain number of sub-domains have been mapped to certain set of processors. The care must be taken to what we call balance the computing load on each processor otherwise it may happen that some processors are heavily loaded.

But since we are solving the problem, the solution process is linked together on all the sub-domains. So other processors would remain ideal until the (()) (44:33) processors have computed or completed their part of computations. So this is why it is essential for us to do appropriate load balancing. Now grid partitioning approach or domain decomposition approach in both the cases we have to take care of this load balancing at that level itself.

And we have also got to be very careful about the choice of the solution schemes. So we have to use those iterative schemes, which can be easily parallelized for the solution of discrete algebraic systems. We cannot choose any scheme howsoever if we send it might be for a serial machine. Take for instance of early decomposition based solution process for linear equations.

Early decomposition is very difficult parallelize. So even if it were possible for us to use that we will not use it. We have to use an iterative scheme, which can be parallelized easily. So the most popular choices are the Krylov subspace solvers, which require basically matrix vector multiplications in iterative solution process and this matrix vector multiplications they can be what we call decomposed or distributed over a set of processors or processor codes very easily.

So load balancing is very easy or parallelizing these Krylov subspace solvers whether they are PCG or GMRES or BiCGSTAB all of these methods can be easily parallelized. The same holds good with multigrid methods. Multigrid are what we call multilevel methods of geometric or algebraic time. There are many of them which can be parallelized. In fact, their basic design itself is inherently parallel.

So that is why nowadays in our commercial CFD solvers multigrid and Krylov subspace solvers are the methods of choice for the solution of a problem because they are inherently parallel or they can be easily parallelized. So these are the aspects which we must take care of in parallel implementation specifically for writing your own code. So this way we would put a full stop to our discussions on the parallel implementation aspects and the practical implementation aspects.

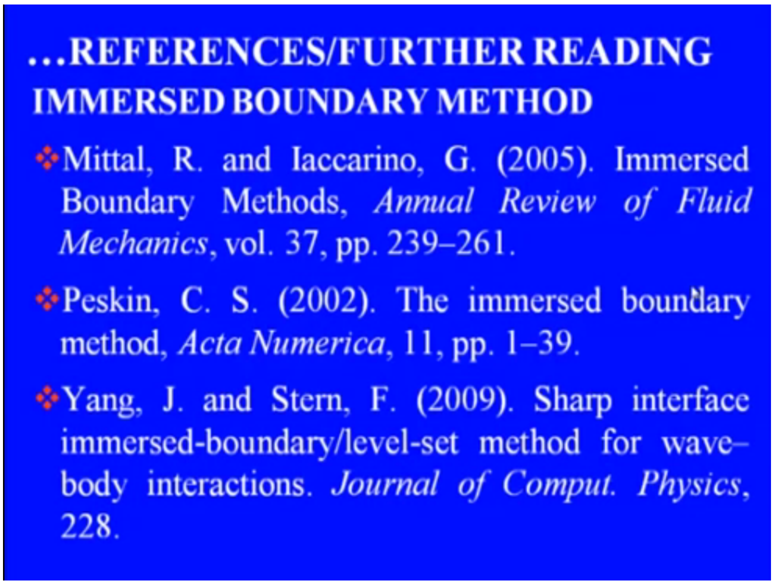
And we come to the end of our course. I hope you have enjoyed the course, enjoyed learning various nuances of CFD starting from mathematical modeling to numerical discretization schemes both for spatial discretization as well as temporal discretization and our discussions on Navier-Stokes solutions and solution of turbulent flows. We did not have much time in this introductory course to discuss in detail about grid generation and parallel implementation aspects.

So we had a very sketchy overview of these processors, but for those of who are interested to further explode the beautiful science of CFD or other science and art of CFD, computational fluid dynamics is as much an art as it is science for the simulation or solution of our real life problems. I would refer you to further references. We have already seen in the course of different modules appropriate preferences.

(Refer Slide Time: 48:11)

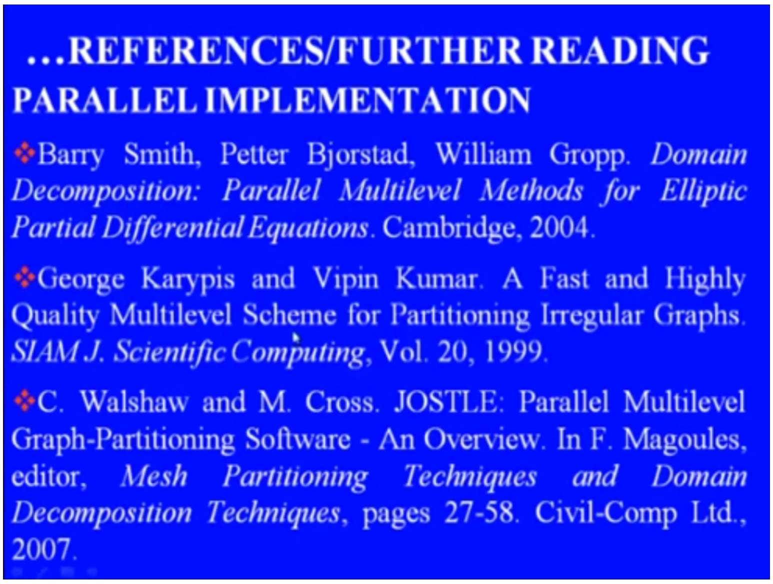
The references for this module for verification and validation you can look at the AIAA guide published in 1988, guide for verification and validation of computational fluid dynamics simulations. You can also look into the text book of Versteeg and Peric this is one of the very compact text books of CFD and similarly Versteeg and Malalasekara's book published in 2007 also provides a very compact description of both AIAA and aircraft guidelines.

(Refer Slide Time: 48:57)



For immersed boundary method there are many, many research papers available. You can look at the review paper of Mittal and Iaccarino published in 2005 in annual reviews of fluid mechanics or you can look at the classic paper of Peskin in 2002 on immersed boundary method published in Acta America or recent ones by Yang and Stern published in general computational physics and shop interface immersed boundary levels of method for wave body interactions.

(Refer Slide Time: 00:00)



And this particular methodology, which helps us to get near exact representation of a complex interface or curve interface this wise called shop interface immersed boundary level method. Similarly on parallel implementation you can look the text book of Barry Smith, Petter Bjorstad and William Gropp, domain decomposition parallel multilevel method for elliptic partial differential equation.

This particular book provides a very nice description of multigrid methods as well as source methods and their parallel implementation. Then for the 2 grid partitioning tools which we mentioned you can have a look at Karypis and Vipin Kumar’s paper on fast and high quality multilevel scheme partitioning irregular graphs this is the basis of METIS and then similarly for JOSTLE.

You can have a look at the chapter on this contributed volume by Walshaw and Cross JOSTLE parallel multilevel graph partitioning software and overview. So thank you very much. I hope you have enjoyed the course thoroughly and you can built on the knowledge, which we or what we learnt in this introductory course to perform well informed CFD analysis using let us say commercial CFD tools.

And you should also be able to write your own code keeping in view what we have learnt in this course. So happy numerical analysis and make use of CFD techniques in solving problems in your own domain whether they might be from civil engineering, aerospace or chemical sciences or even biological sciences. There are so many applications available for the CFD. Thank you once again.