



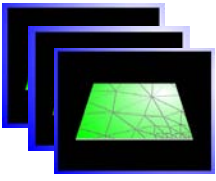

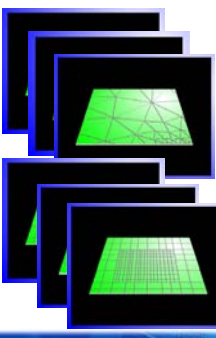







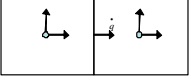



Mesh Technologies for CFD: Pros and Cons

	<p>The title of the presentation is Mesh Technologies for CFD: pros and cons.</p>
<p style="text-align: center;">Topics</p> <ul style="list-style-type: none"> ▪ Needs and Choices ▪ Factors affecting Choice of Grid System ▪ Cell Shape and Effect on “Grid Quality” ▪ Representing Non-Rectangular Geometries <ul style="list-style-type: none"> ▪ Arrangement - Structured or Unstructured ▪ Mesh Generation ▪ Examples ▪ EFD’s Rectangular Adaptive Mesh Technology 	<p>These are the topics that I will be going through. I am going to talk about the needs and choices when it comes to a mesh, looking at the factors affecting the choice of grid system that we use. I am going to talk a little bit about cell shape and the effect of that on essentially grid quality, talk about the representation of non-rectangular geometries and the importance of that and the impact that has on essentially the choices that we have for the mesh, the arrangement of the mesh, whether it’s structured or unstructured, how the mesh is generated, then we’re going to look at some examples both from the past and more recently of meshes and mesh generation approaches and then finally talk a little bit about EFD’s rectangular adaptive mesh technology.</p>
<p style="text-align: center;">Needs and Choices</p> <ul style="list-style-type: none"> ▪ Why a Mesh? <ul style="list-style-type: none"> ▪ Navier-Stokes and heat conservation equations insoluble for practical cases ▪ Need to ‘discretise’ the equations: ▪ Split space into many small volumes 	<p>Okay, we can talk about needs and choices of the mesh. Firstly, why do we need a mesh? Well, the Navier-Stokes equations and the heat conservation equation are essentially not solvable for practical cases, so what we need to do is take those differential equations and discretise them to produce algebraic equations which are actually much easier to solve, and that’s done by splitting the region of interest into many small volumes.</p>
<p style="text-align: center;">Needs and Choices</p> <ul style="list-style-type: none"> ▪ Why a Mesh? <ul style="list-style-type: none"> ▪ Navier-Stokes and heat conservation equations insoluble for practical cases ▪ Need to ‘discretise’ the equations: ▪ Split space into many small volumes ▪ Make assumptions <ul style="list-style-type: none"> ▪ E.g. heat source uniform over cell volume, ▪ Velocity uniform over cell face ▪ Reduces terms in PDE to simple algebraic equations (solved iteratively) 	<p>And then we make assumptions in order to convert the partial differential equations into algebraic equations which we actually solve, so for example, a heat source is considered to be uniform perhaps over the entire cell volume, the velocity is assumed to be uniform over a cell face, and so on. Obviously, to capture the variation of these things over the region of interest, we have to split the region of interest into many small volumes, which is why we need a mesh with many, many cells in it. So it reduces the terms in the partial differential equations to simple algebraic equations that we can solve iteratively.</p>

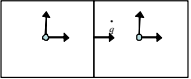

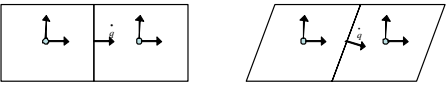



Mesh Technologies for CFD: Pros and Cons

<p style="text-align: center;">Needs and Choices</p> <ul style="list-style-type: none"> ▪ What are the choices of mesh? ▪ Shape: <ul style="list-style-type: none"> ▪ Cartesian ▪ Hexahedral ▪ Tetrahedral ▪ Polyhedral  	<p>Moving on, what are the choices of mesh? Well, broadly speaking we can look at this in two ways, one is to talk about the shape of the cells and perhaps the simplest is Cartesian, as the little image shows there. Hexahedral cells we can produce just by essentially taking that Cartesian mesh and distorting it, and then we can produce cells which are tetrahedral or, going up from that, polyhedral, and there are various options there.</p>
<p style="text-align: center;">Needs and Choices</p> <ul style="list-style-type: none"> ▪ What are the choices of mesh? ▪ Shape: <ul style="list-style-type: none"> ▪ Cartesian ▪ Hexahedral ▪ Tetrahedral ▪ Polyhedral ▪ Arrangement, or organization: <ul style="list-style-type: none"> ▪ Structured ▪ Unstructured ▪ Partly structured  	<p>The other way of looking at it is from an arrangement point of view, we can have structured grid, and the Cartesian grid and hexahedral grid that we saw earlier are structured, or unstructured, and generally tetrahedral polyhedral grids are fully unstructured in that you don't know exactly who the neighbour cells are. In a structured mesh, you know you have a neighbour to the left, a neighbour to the right, a neighbour above, a neighbour below, and so on, so it makes it much easier to find information when you come to actually solve the equations. Then there is partially structured, which is actually a technology that we use in our Flotherm software, of having locally fine Cartesian grids embedded within a coarser Cartesian grid, and this is a way in which we have found we can traverse the large change in length scales associated in electronics problems.</p>
<p style="text-align: center;">Factors Affecting Choice of Grid System</p> <ul style="list-style-type: none"> ▪ These include: <ul style="list-style-type: none"> ▪ Cell Shape and Its Effect on "Grid Quality" ▪ Representation of Non-Rectangular Geometries ▪ Choice of Grid Arrangement <ul style="list-style-type: none"> ▪ Structured or Unstructured ▪ Grid Generation <ul style="list-style-type: none"> ▪ Effort required to create an acceptable mesh 	<p>Talking about the factors affecting the choice of grid, these include the cell shape and its effect on grid quality, as we have said, whether or not we need to represent non-rectangular geometries. There is a choice of grid arrangement, structured or unstructured. The grid generation itself, the effort required to actually create an acceptable mesh, is essentially part of the mix of factors that affect our choice of grid system.</p>
<p style="text-align: center;">Cell Shape and Its Effect on "Grid Quality"</p> <ul style="list-style-type: none"> ▪ Orthogonality: <ul style="list-style-type: none"> ▪ The quality of lying or intersecting at right angles ▪ Line between cell centers is normal to face ▪ Simplifies discretization of the PDEs ▪ Fewer approximations in formulating algebraic equations 	<p>Orthogonality is the quality of things either lying or intersecting at right angles. In the context of a mesh for CFD, what we're really referring to is the line between the cell centres on either side of a cell face is actually normal to that cell face, and this is important, because it partly simplifies the discretisation of the differential equations that we're trying to solve, but it also links to fewer approximations in the algebraic equations that we do solve.</p>





Mesh Technologies for CFD: Pros and Cons

<p style="text-align: center;">Cell Shape and Its Effect on “Grid Quality”</p> <ul style="list-style-type: none"> ▪ Non-orthogonality leads to secondary terms <ul style="list-style-type: none"> ▪ Many more than for an orthogonal mesh in 3D ▪ Potential implications are: <ul style="list-style-type: none"> ▪ Additional CPU time ▪ Additional memory overhead 	<p>So non-orthogonality leads to secondary terms, there are many more terms in a non-orthogonal 3D mesh than there are in an orthogonal 3D mesh and there are a number of implications for that. Additional CPU time wasn't one of the things that people particularly picked up on but certainly there is a cost associated with calculating those additional terms. Some of them have to be recalculated every iteration but others simply have to be calculated at the start and then stored in memory. So as well as an additional CPU overhead, there's also an additional memory overhead associated with non-orthogonal meshes.</p>
<p style="text-align: center;">Cell Shape and Its Effect on “Grid Quality”</p> <ul style="list-style-type: none"> ▪ Non-orthogonality leads to secondary terms <ul style="list-style-type: none"> ▪ Many more than for an orthogonal mesh in 3D ▪ Potential implications are: <ul style="list-style-type: none"> ▪ Additional CPU time ▪ Additional memory overhead ▪ Reduced accuracy <ul style="list-style-type: none"> ▪ due to approximations ▪ Reduced stability <ul style="list-style-type: none"> ▪ Less implicit treatment of secondary terms (e.g. added as sources and sinks) 	<p>We also tend to find reduced accuracy. This is partly due to approximations in the secondary terms and partly because of the difficulties associated with solving the equations when you have the secondary terms present, because you tend to find that they also tend to reduce stability in the system, and this is because the secondary terms are usually handled in what's called a less implicit way than the connections to the neighbour cells. So overall it's not a good thing.</p>
<p style="text-align: center;">Cell Shape and Its Effect on “Grid Quality”</p>  <ul style="list-style-type: none"> ▪ Orthogonal* <ul style="list-style-type: none"> ▪ Total heat flow only depends on dT/dx <p><small>*Cell face normal to line connecting the cell centres</small></p> 	<p>What I wanted to do was just show you a bit of an example of this to give you a flavour of the importance of it, so we're going to talk a little bit about it but without going into it in too much detail. Here we have an orthogonal example of heat flow from one cell to another and here the total amount of heat that flows from one cell to the next in a conduction-only situation purely depends on the temperature gradient in that direction. I have called that direction X for the purposes of this discussion.</p>
<p style="text-align: center;">Cell Shape and Its Effect on “Grid Quality”</p>  <ul style="list-style-type: none"> ▪ Orthogonal* <ul style="list-style-type: none"> ▪ Total heat flow only depends on dT/dx ▪ Non-orthogonal <ul style="list-style-type: none"> ▪ Total heat flow depends on dT/dx and dT/dy <p><small>*Cell face normal to line connecting the cell centres</small></p> 	<p>Here is a non-orthogonal grid, here the heat flux normal to that cell face depends not only on the temperature gradient in X but it also depends on the temperature gradient in Y, the vertical direction, and in three dimensions in Z, as well.</p>





Mesh Technologies for CFD: Pros and Cons

<p style="text-align: center;">Cell Shape and Its Effect on “Grid Quality”</p>  <ul style="list-style-type: none"> ▪ Orthogonal <ul style="list-style-type: none"> ▪ Total mass flux only depends on u ▪ Force only depends on dp/dx ▪ Convected heat and momentum depend only on x-direction neighbour values (simple upwind)** <p><small>**More for higher order differencing schemes</small></p>  <p style="font-size: small;"><small>John Peery, Mesh Technologies for CFD, May 2008</small></p>	<p>If we move on and talk a little bit about flow within the system, here we have mass flow from one cell to the next, again, in an orthogonal system, the mass flow only depends on the velocity of that cell face, as in the X velocity at that cell face. The force on that X velocity only depends on the pressure gradient in the X direction, so again, it's rather simpler than the non-orthogonal case and any convective quantity, if we're using a relatively simple scheme of saying what actually leaves the cell is the value in the cell, which is called simple upwind as a differencing scheme, it means that we only have to consider the value of the variable in that cell and not in other grid cells.</p>
<p style="text-align: center;">Cell Shape and Its Effect on “Grid Quality”</p>  <ul style="list-style-type: none"> ▪ Orthogonal <ul style="list-style-type: none"> ▪ Total mass flux only depends on u ▪ Force only depends on dp/dx ▪ Convected heat and momentum depend only on x-direction neighbour values (simple upwind)** ▪ Non-orthogonal <ul style="list-style-type: none"> ▪ Total mass flux depends on u and v ▪ Force depends on dp/dx and dp/dy ▪ Convected quantities depend on x- and y-neighbour values <p><small>**More for higher order differencing schemes</small></p>  <p style="font-size: small;"><small>John Peery, Mesh Technologies for CFD, May 2008</small></p>	<p>So again, looking at it from a non-orthogonal point of view, we have a non-orthogonal grid. The mass flow normal to that cell face depends not only on the U velocity but also on the V velocity. We have to come to an approximation as to how much of the V velocity from each of the grid cells contributes to that, the force on the flow through that cell face depends on the pressure gradient in both X and Y and in Z in 3D, and any convective quantities, what is convected through that cell face, for example, if it's heat that's being convected, depends not only on the temperature in the neighbour cell to the left there but it will also depend on the neighbours in the Z and the Y directions.</p>
<p style="text-align: center;">Cell Shape and Its Effect on “Grid Quality”</p> <ul style="list-style-type: none"> ▪ Cartesian is the natural choice: <ul style="list-style-type: none"> ▪ Governing equations are naturally formulated in Cartesian coordinates (i.e. u, v and w)  <p style="font-size: small;"><small>John Peery, Mesh Technologies for CFD, May 2008</small></p>	<p>So non-orthogonality introduces a lot more complexity into the system and in that sense, Cartesian is actually a natural choice for a grid system for CFD. Indeed, when CFD started, CFD started on Cartesian meshes and there are a number of reasons for that, partly the reasons we have already covered, but the governing equation is naturally formulated in Cartesian coordinates.</p>
<p style="text-align: center;">Cell Shape and Its Effect on “Grid Quality”</p> <ul style="list-style-type: none"> ▪ Cartesian is the natural choice: <ul style="list-style-type: none"> ▪ Governing equations are naturally formulated in Cartesian coordinates (i.e. u, v and w) <ul style="list-style-type: none"> ▪ Further simplifies discretization task ▪ Users of non-orthogonal systems go to great lengths to prevent poor-quality grids: <ul style="list-style-type: none"> ▪ Grid generation usually involves some manual “adjustment” of any automatically-generated grid  <p style="font-size: small;"><small>John Peery, Mesh Technologies for CFD, May 2008</small></p>	<p>It simplifies the discretisation task that we've just looked at, and users of non-orthogonal systems tend to go to great lengths to prevent poor quality grids, and usually that involves some manual adjustment to any automatically generated grid in order to improve its quality. It is something that automatic grid generators don't do that well, even today, for non-orthogonal grid systems.</p>


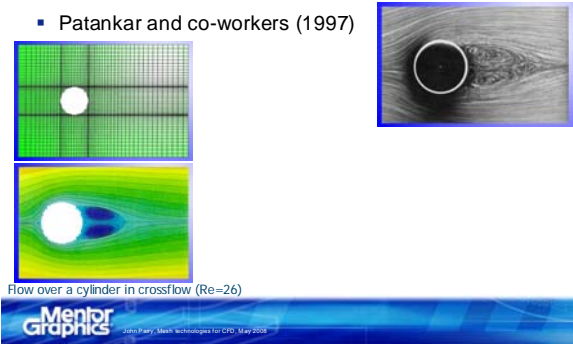

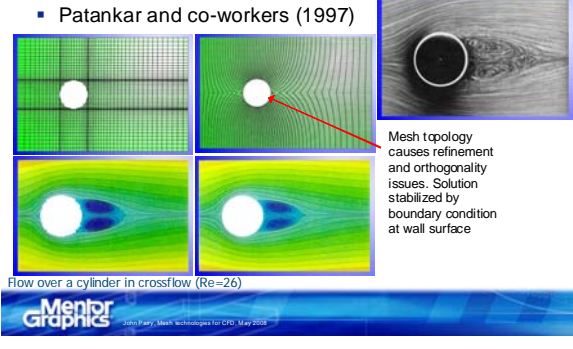

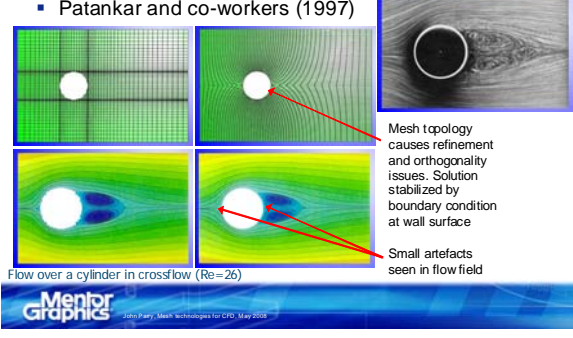

Mesh Technologies for CFD: Pros and Cons

<p style="text-align: center;">Cell Shape and Its Effect on “Grid Quality”</p> <ul style="list-style-type: none"> ▪ Cartesian is the natural choice: <ul style="list-style-type: none"> ▪ Governing equations are naturally formulated in Cartesian coordinates (i.e. u, v and w) <ul style="list-style-type: none"> ▪ Further simplifies discretization task ▪ Users of non-orthogonal systems go to great lengths to prevent poor-quality grids: <ul style="list-style-type: none"> ▪ Grid generation usually involves some manual “adjustment” of any automatically-generated grid ▪ Can become the most time consuming part of the entire process of CFD analysis 	<p>Indeed, in a non-orthogonal system, this can actually become the most time-consuming part of the entire CFD process. Depending on the application and depending on the complexity, people can certainly spend days, weeks, and indeed in some cases actually months, coming up with a mesh in order to solve a particular problem.</p>
<p style="text-align: center;">Representation of Non-Rectangular Geometries</p> <ul style="list-style-type: none"> ▪ Only motivation to not use a Cartesian grid ▪ Two different generic approaches possible <ul style="list-style-type: none"> ▪ Boundary First and Boundary Last 	<p>We’re going on move on now to talk about the representation of non-rectangular geometries. Really this is the only motivation as it were for not using a Cartesian grid, it’s the fact that we need to have a representation of an object that is not itself Cartesian, as in rectilinear and aligned with the coordinate directions. There are two generic approaches to this, which I’m going to introduce in the terminology that we, Flomerics, have developed for it, which is Boundary First and Boundary Last.</p>
<p style="text-align: center;">Representation of Non-Rectangular Geometries</p> <ul style="list-style-type: none"> ▪ Only motivation to not use a Cartesian grid ▪ Two different generic approaches possible <ul style="list-style-type: none"> ▪ Boundary First and Boundary Last ▪ Boundary First (Body fitted or BREP meshes) <ul style="list-style-type: none"> ▪ Surface of geometry is meshed ▪ Surface mesh creates faces in volume mesh 	<p>Boundary First is the traditional way of creating meshes for CFD for non-rectangular geometries, which is to use a body fitted treatment, and the reason it’s called boundary first in our terminology is that essentially the process starts with the surface of the geometry being mesh, so you end up with a two-dimensional mesh or a surface mesh on the surface of the geometry and that surface mesh creates cell faces that appear in the volume mesh for the surrounding fluid region, or the solid region if you’re considering heat transfer in the solid as well.</p>
<p style="text-align: center;">Representation of Non-Rectangular Geometries</p> <ul style="list-style-type: none"> ▪ Only motivation to not use a Cartesian grid ▪ Two different generic approaches possible <ul style="list-style-type: none"> ▪ Boundary First and Boundary Last ▪ Boundary First (Body fitted or BREP meshes) <ul style="list-style-type: none"> ▪ Surface of geometry is meshed ▪ Surface mesh creates faces in volume mesh ▪ Boundary Last: <ul style="list-style-type: none"> ▪ Fluid volume is meshed ▪ Intersection of geometry with volume mesh defines solid/fluid boundary ▪ Various treatments possible for these ‘cut’ cells 	<p>The second is Boundary Last which is in a sense the approach that we tend to take, where the fluid volume is mesh and then the intersection of the geometry with the volume mesh essentially defines the solid/fluid boundary, and we can arrange to improve that representation by having the mesh fine near to the surface. But what you essentially have are then cells that are cut by the solid surface and there are various ways of treating those cut cells, but it’s the cut of the geometry with the volume mesh that’s created first that produces, if you like, the surface mesh and therefore we think of this as Boundary Last.</p>


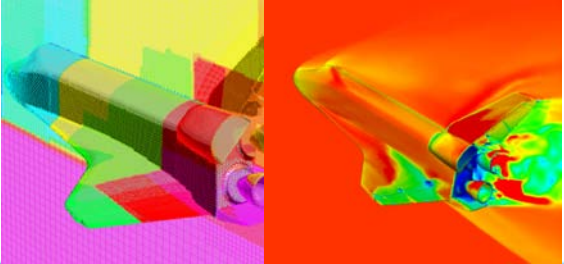

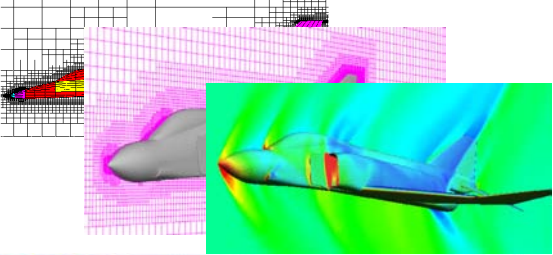

Mesh Technologies for CFD: Pros and Cons

<p style="text-align: center;">Representation of Non-Rectangular Geometries</p> <ul style="list-style-type: none"> ▪ Body-fitted (or BREP) meshes <ul style="list-style-type: none"> ▪ Apparent benefit of fitting mesh to geometry, but: ▪ Requires automatic (i.e. mechanistic) mesh generation 	<p>Body-fitted or BREP meshes, as they're sometimes referred to, have the apparent benefit of fitting the mesh to the geometry, and indeed it is a benefit, but it does require what I have called here automatic mesh generation. The term "automatic" is a little bit misleading in the sense that it is not necessarily entirely automatic in that it does not require any input from the user, quite the opposite, in fact, what it means is that the mesh is actually generated according to some mechanistic approach, so there's an algorithm that's being used to create the mesh.</p>
<p style="text-align: center;">Representation of Non-Rectangular Geometries</p> <ul style="list-style-type: none"> ▪ Body-fitted (or BREP) meshes <ul style="list-style-type: none"> ▪ Apparent benefit of fitting mesh to geometry, but: ▪ Requires automatic (i.e. mechanistic) mesh generation ▪ Only fully automatic for tetrahedral mesh <ul style="list-style-type: none"> ▪ Cell size change causes non-orthogonality <ul style="list-style-type: none"> ▪ Tends to result in high cell counts ▪ Poor capture of wall-bounded flows (use prisms) ▪ Can be hard to control near-wall mesh 	<p>It's only really fully automatic for tetrahedral meshes, so in that sense tetrahedral meshes have an advantage because you can fully automatically generate a mesh. The problem is one of non-orthogonality, in that a regular tetrahedral mesh is in fact fully orthogonal, where each face of the tetrahedron is an equilateral triangle. The problem really comes when you try to change the cell size from a small cell to a large cell and one can't do that very quickly without running into orthogonality problems. What that tends to result then in is pretty high mesh counts in terms of the number of cells and also they suffer from having a relatively poor capture of the surface, the wall bounded flows, so getting a fine mesh near to the surface is also a problem with tetrahedral meshes generally, so as I was saying, hard to control the near wall mesh.</p>
<p style="text-align: center;">Representation of Non-Rectangular Geometries</p> <ul style="list-style-type: none"> ▪ Body-fitted (or BREP) meshes <ul style="list-style-type: none"> ▪ Apparent benefit of fitting mesh to geometry, but: ▪ Requires automatic (i.e. mechanistic) mesh generation ▪ Only fully automatic for tetrahedral mesh <ul style="list-style-type: none"> ▪ Cell size change causes non-orthogonality <ul style="list-style-type: none"> ▪ Tends to result in high cell counts ▪ Poor capture of wall-bounded flows (use prisms) ▪ Can be hard to control near-wall mesh ▪ Meshing both inside and outside geometry can be problematic 	<p>The other issue really is that often if you want to solve heat transfer in solids as well as flow around an object, then the Boundary First method of meshing the surface and then fitting a volume mesh to that surface mesh can cause problems because a surface mesh that's suitable for the outside mesh may not be suitable for the inside mesh, so sometimes meshing both inside and outside of solid objects can lead to problems with this.</p>
<p style="text-align: center;">Representation of Non-Rectangular Geometries</p> <ul style="list-style-type: none"> ▪ Alternative approach is to allow geometry to 'cut' the mesh: <ul style="list-style-type: none"> ▪ Mesh can be Cartesian with inherent advantages ▪ Meshes both inside and outside bodies <ul style="list-style-type: none"> ▪ good for conjugate problems 	<p>The alternative approach is to have the geometry simply cut the mesh, which is our Boundary Last approach. The advantage of this is that the mesh can then be Cartesian, you can mesh both outside and inside the body without any difficulties, so it's quite good for conjugate heat problems, which is essentially the application area where Flomerics has been very active since its start, we started looking at heat transfer in buildings and in electronics systems, where conjugate heat transfer, which means conduction in solids as well as convection in the air, within a single calculation was actually something we had to do fairly implicitly, whereas it was actually at the time rather the exception than the rule for CFD.</p>

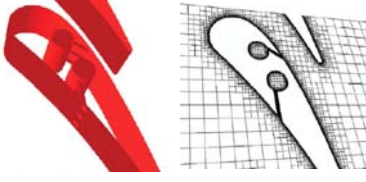


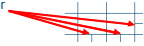

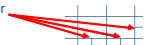

Mesh Technologies for CFD: Pros and Cons

<p style="text-align: center;">Representation of Non-Rectangular Geometries</p> <ul style="list-style-type: none"> ▪ Alternative approach is to allow geometry to 'cut' the mesh: <ul style="list-style-type: none"> ▪ Mesh can be Cartesian with inherent advantages ▪ Meshes both inside and outside bodies <ul style="list-style-type: none"> ▪ good for conjugate problems ▪ Finding increasing use (a renaissance) ▪ Test cases show can give as good results as body-fitted meshes 	<p>What we'll see from some of the examples is that this boundary last approach is actually finding something of a renaissance in use, because it was essentially where CFD started and its popularity is increasing again. Test cases tend to show it gives as good results as body- fitted meshes and what we'll do now is look at some examples.</p>
<p style="text-align: center;">Representation of Non-Rectangular Geometries</p> <ul style="list-style-type: none"> ▪ Patankar and co-workers (1997)  <p style="text-align: center;">Flow over a cylinder in crossflow (Re=26)</p> 	<p>This is quite an old one, from Patankar, who is one of the founding fathers of CFD, as it were, working with Brian Spalding at Imperial College in the early 1980s. The case looks at just flow over a cylinder, so it's a cylinder in cross flow, this is a picture of what the flow actually looks like. Then they've looked at a Cartesian mesh and the results for a Cartesian mesh purely looking at flow, not looking at heat transfer here.</p>
<p style="text-align: center;">Representation of Non-Rectangular Geometries</p> <ul style="list-style-type: none"> ▪ Patankar and co-workers (1997)  <p style="text-align: center;">Flow over a cylinder in crossflow (Re=26)</p> 	<p>And then a hexahedral mesh. Well, it's actually not the best Cartesian mesh that you would create, and it is certainly not the best hexahedral mesh that you would create around this cylinder but nonetheless what we can see in the hexahedral mesh is that there's quite a lot of mesh distortion at the leading edge and trailing edge of the cylinder. Those grid cells being that distorted are actually stabilised in this case because of the boundary condition at the surface, which fixes the velocity to zero. If those cells were actually out in the free stream somewhere it might be much harder to get this case to converge.</p>
<p style="text-align: center;">Representation of Non-Rectangular Geometries</p> <ul style="list-style-type: none"> ▪ Patankar and co-workers (1997)  <p style="text-align: center;">Flow over a cylinder in crossflow (Re=26)</p> 	<p>What we can see, just, is that there are some small artefacts in the flow field that come from this level of mesh distortion, not too bad, because of the stabilisation due to the wall surface but some small artefacts there nonetheless. What you can see is that broadly speaking they gave the same results, neither one can be said to be better than the other by comparison with the actual flow.</p>

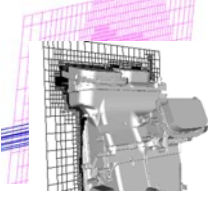
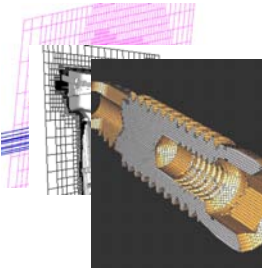
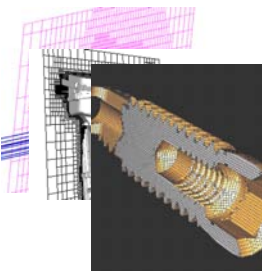
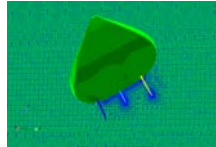
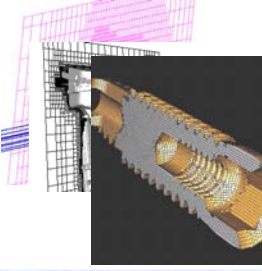

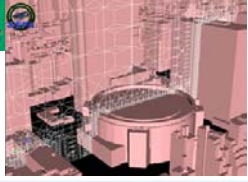
Mesh Technologies for CFD: Pros and Cons

<p style="text-align: center;">Representation of Non-Rectangular Geometries</p> <ul style="list-style-type: none"> ▪ NASA AMES (Aftosmis, 1999 onwards) <ul style="list-style-type: none"> ▪ Long-time advocate of direct use of CAD models in analysis ▪ Avoids “versionitis”: staleness of design due to work on inexact, non-native representations of geometry ▪ Use surface triangulation to describe geometry <ul style="list-style-type: none"> ▪ surface is manifold <p><small>Cartesian octree-structured grid for computing flow around complex geometry</small></p> 	<p>Another set of examples comes from NASA, the NASA AMES lab have been doing work in this area since the late 1990s. They are a long-time advocate of the use of CAD models in analysis and this is actually a theme that is going to come in at this point, where we talk about the difference really between analysis and design, and I’ll come on to that more in a moment. The reason they advocate this is that it avoids the problem of “versionitis”, as Aftosmis refers to it, which is essentially staleness in the design, due to working on inexact non-native representations of the geometry, in that the analysis process takes on a life of its own while the design itself is actually evolving along a different path and so one ends up with this versionitis problem. What they tend to do is use surface triangulation just to describe the geometry because it ensures that the surface is manifold, which means that there are no gaps in it, there are no leaks, so that any volume mesh that’s then created sees a continuous surface.</p>
<p style="text-align: center;">Representation of Non-Rectangular Geometries</p> <ul style="list-style-type: none"> ▪ NASA AMES (Aftosmis, 1999 onwards)  	<p>Just looking at some examples here, the space shuttle, which they were obviously looking at in those days, the surface mesh is entirely triangulated but if you look, the mesh technology they’re using to mesh the volume is actually a Cartesian mesh, using an octree-type approach, where cells are subdivided into four in 2D and eight in 3D, to refine near to the surface in any regions of flow that are of interest, and they’ve certainly shown that they can get very good results on this type of mesh.</p>
<p style="text-align: center;">Representation of Non-Rectangular Geometries</p> <ul style="list-style-type: none"> ▪ NASA AMES (Aftosmis, 1999 onwards)  	<p>Other examples they’ve looked at, as you can imagine, very strong aeronautical fields to their applications, this I think is the F14 Fighter, and again, meshes around, as you can see, although they’ve used surface triangulation to capture the surface, it’s an octree mesh that’s being used in the volume of the flow and again getting very good results, it’s actually capturing the shocks that form on the plane in supersonic flow.</p>

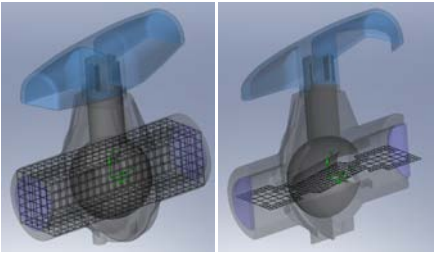


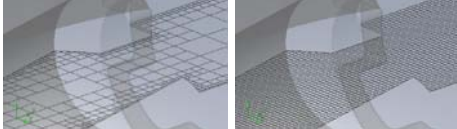

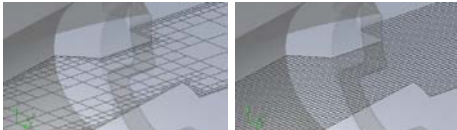

Mesh Technologies for CFD: Pros and Cons

<p style="text-align: center;">Representation of Non-Rectangular Geometries</p> <ul style="list-style-type: none"> ▪ University of Cambridge (Prof. W. Dawes, 2007)  <ul style="list-style-type: none"> ▪ Turbine blade example: considers introducing a hole <ul style="list-style-type: none"> ▪ Mesh only recomputed local to hole 	<p>The next example is much more recent actually, so we jump almost ten years from the world that started in the last two examples to this, to Professor Bill Dawes at Cambridge, who has actually a huge pedigree in turbo machinery flows, again, one of the founding fathers of CFD, really. What he has been looking at is using octree-type meshes, partly because it gives very good scalability on to parallel hardware but also because it helps with the design process. What they consider is an example of introducing holes or adjusting holes in the turbine blades to help cool the blades, and one of the real advantages is that it's only local to the geometry change that you need to remesh, the rest of the mesh can be left exactly as it was.</p>
<p style="text-align: center;">Representation of Non-Rectangular Geometries</p> <ul style="list-style-type: none"> ▪ University of Cambridge (Prof. W. Dawes, 2007) <ul style="list-style-type: none"> ▪ Interesting because work is more recent <ul style="list-style-type: none"> ▪ Turbo machinery pioneered use of CFD on unstructured BREP meshes ▪ Work demonstrates local re-meshing ▪ Prof. Dawes argues body-fitted meshes are now holding back the use of CFD in <i>design</i> <ul style="list-style-type: none"> ▪ CAD model often dirty, requiring healing ▪ Design meaning geometric changes ▪ Necessary to explore design space 	<p>He makes a number of interesting points. As I've said, the work is interesting because it's more recent, it's also interesting because turbo machinery really pioneered the use of CFD with unstructured meshes. Professor Dawes is arguing now that using body-fitted meshes is now actually holding back the use of CFD in design, and he differentiates design from analysis in that design is all about changing the geometry, and typically one has a CAD model which is often dirty, the geometry isn't perfect, design very often means geometric changes, and one has to explore the design space, so one has to make many, many changes to the geometry in order to come up with the best design, or at least this is the scenario that he's considering.</p>
<p style="text-align: center;">Representation of Non-Rectangular Geometries</p> <ul style="list-style-type: none"> ▪ Harpoon – the extreme mesher (Sharc Ltd.) <ul style="list-style-type: none"> ▪ Fully automatic hex-dominated mesher (Octree*) <p style="font-size: small;">*Hanging nodes can be removed for solvers that don't support them</p>  	<p>A sort of final example is actually not of the use of Cartesian-type meshes for CFD but is actually a meshing tool, harpoon, from Shark Limited. It's a fully automatic hex-dominated mesher, as they refer to it, which essentially means it creates octree meshes. The hanging nodes that are formed in the octree, which are, if you like, the node that you get in the centre of a cell face as you go from one cell to the next, across that four to one split, can be removed for solvers that don't support them. So they're kind of covering all bases, really, with this.</p>
<p style="text-align: center;">Representation of Non-Rectangular Geometries</p> <ul style="list-style-type: none"> ▪ Harpoon – the extreme mesher (Sharc Ltd.) <ul style="list-style-type: none"> ▪ Fully automatic hex-dominated mesher (Octree*) <p style="font-size: small;">*Hanging nodes can be removed for solvers that don't support them</p>  <ul style="list-style-type: none"> ▪ Moves nodes of hexas onto surface (body-fitted) ▪ Gives good volume mesh for body-fitted codes ▪ Resolves features larger than a user-set tolerance ▪ Very fast, reducing CAD-to-solution time ▪ Robust, automated repair of leaks ▪ Smoothing available to improve mesh quality 	<p>The way they create this mesh is to create essentially the volume mesh as we've said, so it's very much a Boundary Last approach, but then the way they actually deal with the surface is not actually to split the cells but simply to move nodes in the volume mesh down on to the surface in order to capture the surface shape, so it gives very good volume meshes for body-fitted codes, codes that are, if you like, locked into that body-fitted paradigm. It resolves any features that are larger than a user-set tolerance, it is fast and does definitely reduce CAD to solution time, which is really what's important in a design type environment, so it has a number of advantages over perhaps traditional meshers for CFD.</p>

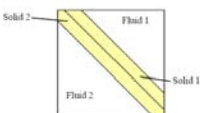

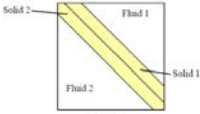
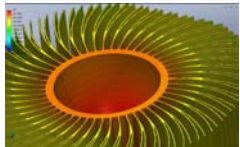

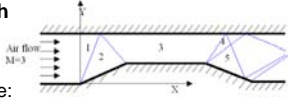

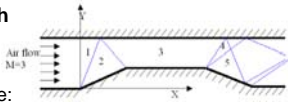
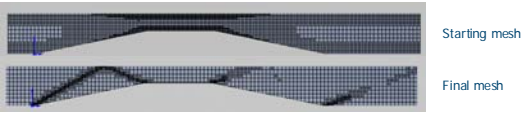

Mesh Technologies for CFD: Pros and Cons

<p>Representation of Non-Rectangular Geometries</p> <p>Hex-dominated Harpoon mesh examples:</p>  <p>Mentor Graphics John Paery, Mesh Technologies for CFD, May 2008</p>	<p>Here are some examples, again there is a strong external aerodynamics feel to this, because this is the sort of application that produces the best graphics, as it were. Here is an example of part of an engine, I believe.</p>
<p>Representation of Non-Rectangular Geometries</p> <p>Hex-dominated Harpoon mesh examples:</p>  <p>Mentor Graphics John Paery, Mesh Technologies for CFD, May 2008</p>	<p>The final one is quite interesting, because it's a screw thread, and part of this valve body, as you can see, it's not the sort of geometry that would necessarily be well suited to a Cartesian mesh but they've managed to mesh it very effectively by moving cells onto the surface.</p>
<p>Representation of Non-Rectangular Geometries</p> <p>Hex-dominated Harpoon mesh examples: With hanging nodes removed:</p>   <p>Mentor Graphics John Paery, Mesh Technologies for CFD, May 2008</p>	<p>Here are some examples with the hanging nodes removed. What you can see essentially is that the cells actually have triangles added, in 2D at least, so that we can go from this large cell here through to the two smaller cells below it, without this hanging node being present, so the hanging node here is removed by the introduction of the two triangles.</p>
<p>Representation of Non-Rectangular Geometries</p> <p>Hex-dominated Harpoon mesh examples: With hanging nodes removed:</p>    <p>Mentor Graphics John Paery, Mesh Technologies for CFD, May 2008</p>	<p>Here is the second example of, this is flow over potentially a landscape of buildings, again with the hanging nodes removed. They provide smoothing, because removing these hanging nodes in this way does introduce orthogonality issues, as you can probably appreciate, but nonetheless it does provide a fast way of getting a mesh.</p>

Mesh Technologies for CFD: Pros and Cons

<p>EFD's Rectangular Adaptive Mesh</p> <ul style="list-style-type: none"> ▪ Octree mesh constructed from Cartesian base mesh  	<p>So octree mesh constructed from a Cartesian base mesh is essentially what we deal with in EFD, so we're moving on now to talk about EFD's rectangular adaptive mesh. To give you a flavour of this, this ball valve case is an example we use very heavily, because it does show the benefits really of CAD-embedded CFD. Here is the base mesh as a starting point and then shown in relief on a cross section. This is it refined, and the refinement here essentially is refining it near to surfaces.</p>
<p>EFD's Rectangular Adaptive Mesh</p> <ul style="list-style-type: none"> ▪ Separate refinement of: <ul style="list-style-type: none"> ▪ solid, fluid and partial cells ▪ Curvature, small features and narrow channels; ▪ Local refinement of regions, faces and edges 	<p>Essentially, what we have, going from that base mesh to the final octree mesh that we finish up with, we have the ability to refine separately for solid, fluid, and partial grid cells. We can resolve in regions of high curvature, we can resolve particularly small features, we can resolve narrow channels, which are, if you like, gaps between two discrete pieces of CAD geometry that otherwise the system doesn't know about, and we can apply these refinements globally or we can apply them just local to regions, so a region being a three-dimensional region of space, or to a face or faces of an object or edges of an object.</p>
<p>EFD's Rectangular Adaptive Mesh</p> <ul style="list-style-type: none"> ▪ Separate refinement of: <ul style="list-style-type: none"> ▪ solid, fluid and partial cells ▪ Curvature, small features and narrow channels; ▪ Local refinement of regions, faces and edges  	<p>Here is the ball valve example again and what I've done is refined the partial cells in this case, so you can just see how it picks out the surface. If I apply the same fineness of mesh just to the fluid region, so I mesh all of the fluid region at the same level of fineness, I get this mesh.</p>
<p>EFD's Rectangular Adaptive Mesh</p> <ul style="list-style-type: none"> ▪ Separate refinement of: <ul style="list-style-type: none"> ▪ solid, fluid and partial cells ▪ Curvature, small features and narrow channels; ▪ Local refinement of regions, faces and edges  <ul style="list-style-type: none"> ▪ Directly uses CAD geometry so solid surface areas and volumes are known 	<p>One of the advantages of essentially being CAD embedded is that the CFD knows, if you like, from the CAD system, how much volume is in each grid cell, how much surface area is in each grid cell, so we're not really using the discretisation of the surface with the mesh actually to give us the correct areas and volumes, it's something that we can get directly from the CAD system.</p>

Mesh Technologies for CFD: Pros and Cons

<p>EFD's Rectangular Adaptive Mesh</p> <ul style="list-style-type: none"> EFD uses multiple control volumes per cell, e.g. <ul style="list-style-type: none"> at solid-fluid interfaces to capture thin geometry Results are plotted on the actual CAD geometry Produces surprisingly good results on relatively coarse meshes  <p><small>Fig. 1.8 One mesh cell can contain more than one fluid and/or solid volume, during calculation each volume has an individual set of parameters depending on its type (fluid or solid).</small></p> 	<p>EFD uses multiple control volumes per grid cell. This is, if you like, relatively unique technology, I think, and it's the way in which we resolve solid/fluid interfaces. It's also used to capture thin geometry, and here is an example of what we can do, it can go beyond this but this gives you a flavour of what it can do, where one grid cell within the mesh actually has four control volumes, two solid control volumes and two fluid control volumes, and we use this to capture essentially thin geometry. The results are plotted on the actual CAD geometry and what we find is that we can then get surprisingly good results on what are essentially relatively coarse meshes.</p>
<p>EFD's Rectangular Adaptive Mesh</p> <ul style="list-style-type: none"> EFD uses multiple control volumes per cell, e.g. <ul style="list-style-type: none"> at solid-fluid interfaces to capture thin geometry Results are plotted on the actual CAD geometry Produces surprisingly good results on relatively coarse meshes  <p><small>Fig. 1.8 One mesh cell can contain more than one fluid and/or solid volume, during calculation each volume has an individual set of parameters depending on its type (fluid or solid).</small></p>  	<p>I will show you an example of the kind of geometry that this approach is particularly good for. This is a fan sink, so it's a heat sink that's mounted on top of a processor, which has its own integral fan. As you can imagine, the fan blows air down through the heat sink fins and that's used to cool the component underneath. This is the geometry in EFD with temperatures shown on that geometry, and if we refine, you can see that we can actually get good results for the heat spreading in and along those fins. As I say, the mesh that's actually used to create this is actually much coarser than you would expect.</p>
<p>EFD's Rectangular Adaptive Mesh</p> <ul style="list-style-type: none"> Solution Adaptive Mesh <ul style="list-style-type: none"> Local refine/un-refine <ul style="list-style-type: none"> Based on gradients Validation case example:  	<p>I'm going to talk now about EFD's rectangular adaptive mesh. It's essentially a solution adaptive mesh, and what it allows us to do is to refine and unrefine depending on gradients that are predicted by the solver as it progresses. What I'm going to use to illustrate this is actually one of the validation cases that we have with the software. It's a convergent-divergent channel with flow that enters at Mach 3 and goes through the different sections, and the geometry and the Mach number are chosen such that we know where the shocks are and we also know the Mach number in the different regions that are numbered here, 1 through to 5.</p>
<p>EFD's Rectangular Adaptive Mesh</p> <ul style="list-style-type: none"> Solution Adaptive Mesh <ul style="list-style-type: none"> Local refine/un-refine <ul style="list-style-type: none"> Based on gradients Validation case example:   	<p>To create a mesh in EFD, obviously we don't know where the shocks are and what we want to do is capture those shocks, so we don't really have any chance of creating a good mesh to begin with, so I've come up with this as the starting mesh. Once we've done the analysis and refined and unrefined the mesh where it's needed and where it's not needed, we end up with this as the final mesh that was used to capture the shocks . . .</p>

Mesh Technologies for CFD: Pros and Cons

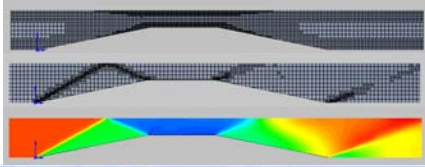
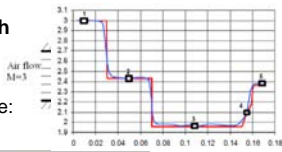
EFD's Rectangular Adaptive Mesh

- **Solution Adaptive Mesh**

- Local refine/un-refine

- Based on gradients

- Validation case example:



Starting mesh

Final mesh

Pressure solution on final mesh



... and here is a plot of pressure on that mesh, and you can see the very sharp capture of the first two, three shocks. Looking at the results along a line through the channel, those different locations, one through to five, what we can see is we are actually capturing correctly the Mach number for each of those locations.

THANK YOU FOR LISTENING!

Dr John Parry

Email: John.Parry@mentor.com

Phone: +44 208 487 3108

MECHANICAL ANALYSIS DIVISION

Mentor Graphics

That's all I was going to cover, thank you very much for listening.