Turbomachinery computational fluid dynamics: asymptotes and paradigm shifts

By W. N. Dawes*

Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK

This paper reviews the development of computational fluid dynamics (CFD) specifically for turbomachinery simulations and with a particular focus on application to problems with complex geometry. The review is structured by considering this development as a series of paradigm shifts, followed by asymptotes. The original S1–S2 blade–blade-throughflow model is briefly described, followed by the development of two-dimensional then three-dimensional blade–blade analysis. This in turn evolved from inviscid to viscous analysis and then from steady to unsteady flow simulations. This development trajectory led over a surprisingly small number of years to an accepted approach—a ‘CFD orthodoxy’.

A very important current area of intense interest and activity in turbomachinery simulation is in accounting for real geometry effects, not just in the secondary air and turbine cooling systems but also associated with the primary path. The requirements here are threefold: capturing and representing these geometries in a computer model; making rapid design changes to these complex geometries; and managing the very large associated computational models on PC clusters. Accordingly, the challenges in the application of the current CFD orthodoxy to complex geometries are described in some detail.

The main aim of this paper is to argue that the current CFD orthodoxy is on a new asymptote and is not in fact suited for application to complex geometries and that a paradigm shift must be sought. In particular, the new paradigm must be geometry centric and inherently parallel without serial bottlenecks. The main contribution of this paper is to describe such a potential paradigm shift, inspired by the animation industry, based on a fundamental shift in perspective from explicit to implicit geometry and then illustrate this with a number of applications to turbomachinery.

Keywords: turbomachinery; CFD; mesh generation; integrated design systems

1. Introduction

Turbomachinery flows represent an extreme challenge to modelling and simulation. Space and time scales can be vast: from blade–blade flow where small-scale details of blade curvature can be critical, ranging to whole multistage machine analysis for matching and distortion transfer, tone noise from trailing edge scale vortex structures to machine scale stall and surge events, coupling and interaction of components at the scale of cooling hole efflux up to compressor exit
diffuser–combustor inlet coupling. In addition, there are varied and strongly interacting fluid/thermal phenomena: a fan normally has a highly three-dimensional shock structure, over-tip leakage flow and probably some hub separation; probably also some interaction with hub leakage flow; certainly some interaction with downstream rows (cyclic back pressure leading to flutter and noise, up-running transmission of noise generated in the stator rows by interaction with down-running fan wakes); interactions also affecting stability and performance with nacelle distortions; and so on. Turbomachinery flows are characterized by very large accelerations and diffusions, very large streamline curvature, high levels of swirl, often with elevated Mach numbers and with large shear and skew. If that were not enough, all this takes place in geometries that range from the relatively smooth and clean primary blade path (albeit with shrouds, air take-offs, casing treatment and so on) to complicated secondary air systems to almost arbitrarily complex turbine cooling configurations.

Turbomachinery computational fluid dynamics (CFD) was developed from the 1960s onwards in a very specialized way, only superficially related to methods designed for simulation of external aerospace flows (for which the classically coupled external flow–boundary layer paradigm is often still adequate even today). Despite the difficulties and challenges, turbomachinery was actively benefiting from simulation earlier than other areas of aerospace (and other areas where fluid dynamics is important like automotive) because the challenges are equally severe for relevant experimental investigation and testing. Simulation has always at its core offered physics-based help in understanding flow physics; and historically, the earliest users were those with the most difficult-to-understand flows—first turbochargers, then axial compressors and then turbines.

To confront this simulation challenge, the turbomachinery CFD development trajectory has been characterized by a number of paradigms that were then patiently developed out onto their asymptotes followed by further paradigm shifts. In the beginning, simulation used very simple models based on little more than velocity triangles. The first paradigm shift was to the S1–S2 or blade–blade-throughflow model of Wu (1951). This was developed to ever higher levels of sophistication until essentially it was replaced by time-marching solutions of compressible Euler or Navier–Stokes equations on simple structured meshes—the second paradigm shift. From this period, the historical development trajectory of turbomachinery CFD became based on a ‘body-fitted’ paradigm, which has in turn led us to Boundary REPresentation (BREP) geometry models.

However, as will be described in more detail later, this BREP paradigm when extended towards real, complex geometries has led to challenging meshing and flow solving and great time penalties associated with making design changes—creativity is effectively stifled. In practice, the complex geometries are still designed using methods based on one-dimensional correlations—real CFD is not actually used in design, only a posteriori analysis.

The essential difficulty of applying orthodox BREP-based CFD to the design process is the complexity of the geometry combined with the vulnerability of the CFD process to the need to change the geometry frequently and rapidly. Even with obvious short cuts like parameterizing the CAD model and templating the CFD process, it takes an unacceptable time to turn around simulations of new geometries—sometimes over a week per new geometry. This is exacerbated by the need, often to make frequent topological changes to the geometry: consider a
new cooling concept—a blade always looks like a blade but the internal cooling system could have one or three or five passages with or without interconnection with an *a priori* unknown number of film cooling passages in *a priori* unknown locations. This means that in practice there is little scope within realistic design time-scales to try innovative design.

This paper first briefly describes the development trajectory of turbomachinery CFD from its origins to the current orthodoxy. Then, and in more detail, the case is made that this orthodoxy is on its asymptote and is not in fact suited for application to complex geometries and that a paradigm shift must be sought. The case is strongly argued that any new paradigm must be geometry-centric and inherently parallel. Then, this paper describes a potential paradigm shift, inspired by the animation industry, and based on a fundamental shift in perspective from explicit to implicit geometry. The paper then closes by showing a number of applications to complex, topical problems in turbomachinery.

2. The trajectory: the S1–S2 model, steady and unsteady blade–blade CFD and BREP-based simulations of flows in complex geometry

(a) The classical S1–S2 model

The background to all turbomachinery simulations is the classical S1–S2 or throughflow and blade–blade model introduced by Wu (1951). The earliest S1 simulations were based simply on annulus area and simple radial equilibrium; the earliest S2 simulations were based on velocity triangles with very simple correlations for loss and deviation. Then, more sophisticated methods were developed for blade–blade and throughflow in turn and applied in a coupled manner. There were field-based methods solving potential flow or stream function equations (e.g. Marsh 1968 or Whitehead & Newton 1985) or methods based on more physical modelling like streamline curvature (e.g. Novak 1967; Wilkinson 1972; Denton 1978). The coupling between the S1 and S2 models, illustrated in figure 1, at its most basic, transferred stream-sheet thickness and its radius from S1 to S2 and returned blade–blade flows angles (deviation) and losses from S2 to S1.
Attempts were made (Jennions & Stow 1984) to achieve this coupling in a fully consistent manner, exchanging not just the basic parameters described above, but factors taking into account the departure of the circumferentially non-uniform blade–blade flow from the axisymmetry assumed in the throughflow. The goal was to recreate the fully three-dimensional flowfield iteratively. However, this coupling was unwieldy, slow to converge and never really achieved a satisfactory manner—the S1–S2 approach was on its asymptote. In time, computers became fast enough that the fully three-dimensional flow could be simulated directly with fewer modelling compromises—this was the first paradigm shift.

(b) Steady blade–blade CFD

Blade–blade simulations started as the quasi-three-dimensional S2 part of the S1–S2 model but, once the advantages were realized, application was extended very quickly into three dimensions. These methods became the key to understanding and managing three-dimensional blade flows. The development of these methods focused almost entirely on time-marching simulations of a density-based conservation set of equations. In the early days, the equations were simply inviscid; later, increasingly sophisticated turbulence models were deployed. It is not the intention here to describe this in much detail—excellent reviews are given by Hirsch & Deconinck (1985) and Denton (1987). The issues confronted by all developers can be summarized as: choice of mesh system; control volumes for flux balancing; turbulence modelling (including wall functions); artificial smoothing for stability and shock capture; multigrid convergence acceleration; and post-processing three-dimensional field data into self-consistent one-dimensional averages.

During this development, a key decision was taken to use body-fitted mesh systems. As will become clear later, this was a critical architectural decision, understandable at the time, as it was the easiest way to get useful predictive accuracy on the then very coarse meshes. However, it locked the turbomachinery CFD development trajectory—not just solvers but mesh generators, geometry engines and kernels and interactions with other disciplines (like blade stress)—into a BREP paradigm.

By the mid 1980s, perfectly useable two-dimensional blade–blade simulations were routine—using quite coarse meshes. Figure 2 shows a fan section at three different flow points simulated on only 3000 mesh cells displaying perfectly acceptable agreement with experiment over the working range (Dawes 1985). Ten years later, essentially the same software with the same turbulence modelling, but run on a million nodes, was producing really quite good agreement with measured secondary flow structures in a linear cascade of turbine blades; see figure 3 taken from Friedrichs (1997).

The apparent big advance in simulation capability from 1987 to 1997 was not due to algorithmic developments or improved turbulence modelling but was simply due to much increased mesh sizes—and this was enabled by developments in computer hardware driven not by CFD but by the demands of home entertainment.

Already in the 1980s applications had been extended, led by Adamczyk (1985) and Denton (1987), to multistage simulations with ‘steadiness’ relative to each blade row enforced by varieties of inter-blade circumferential averaging. Some
Figure 2. Simulations of two-dimensional fan section at different flows on 3k nodes (Dawes 1985).
approaches, with strong echoes of the classical S₁–S₂ coupling models (Atkins et al. 1981), used so-called deterministic stresses to try to model terms associated with the inter-row averaging enforcing periodicity on a non-uniform blade–blade flow. This sort of tool is the current workhorse in turbomachinery multistage analysis having almost totally replaced the classical S₁–S₂ models.

Meanwhile, simple, single blade–blade simulations, especially in two dimensions, on modern computer PC clusters have become so fast that it is now practical for them to be used as part of software-driven automated design optimization—Shelton et al. (1993) is an early example. However, in multistage turbomachinery, a blade row must be designed to operate in its environment rather than in isolation and so practical successes of automated optimization have to date been rather limited (e.g. reducing blade exit circumferential non-uniformity to reduce downstream interaction losses).

(c) Unsteady blade–blade CFD

The next point on the development trajectory was obviously to aim at unsteady simulations. Indeed, some of the very earliest turbomachinery CFD had been for unsteady flow (Erdos et al. 1977) since a turbomachine is clearly an unsteady device (see really early work by Dean & Senoo (1960) and Kerrebrock & Mikolajczak (1970) for example).

Early simulations included upstream wake/downstream blade interaction (e.g. Giles 1988) but the impact on actual design was rather limited since the transition/turbulence modelling could not (and in fact still today cannot)
respond appropriately to the wake/blade suction side boundary layer interaction. Nevertheless, transonic stage simulations, like Fritsch & Giles (1993) for a turbine (illustrated in figure 4) and Dorney & Sharma (1997) for a compressor, gave excellent insight into nonlinear interactions and mixing losses and helped build physics-based deterministic stress models for multistage simulations.

The second simulation shown in figure 4, from Abhari & Epstein (1992), shows how quickly real geometry effects became important as the turbomachinery CFD state-of-the-art neared the top of its development trajectory. In this nonlinear turbine stator–rotor interaction, the intra-row shock motion creates very large unsteady pressure variations around the rotor. There are points during the cycle when the pressure outside the film cooling holes is greater than the plenum pressure inside—but does hot gas enter and if so how far does it get? Hot gas ingress is to be avoided at all costs and so in this and similar problems, it became essential to consider how to include more realistic geometry in simulations.

Figure 4. Unsteady simulations in a transonic turbine stage (Abhari & Epstein 1992; Fritsch et al. 1993).
Although not covered within the context of this review, there is also a very important range of unsteady flow applications related to noise, stall and flutter. In CFD terms, these are all characterized by the dramatic increase of necessary computer power—even for clean blade–blade flows, quite apart from the additional complexity of real geometry. The issue is the spatial scale—essentially the whole annulus with many stages involved (the stalling behaviour of an isolated compressor rotor is quite different from that as part of a stage; Hynes & Longley 1989)—and the time-scale—stall occurs over several rotor revolutions—millions of time-steps for standard unsteady flow solvers.

(d) BREP-based simulations of flows in complex geometries

By the start of the 1990s, then, clean primary path turbomachinery CFD had developed very well and was proving very useful and became widely disseminated. This had all happened surprisingly quickly, over only about 10 years algorithms and so on had converged, although improvements to turbulence modelling (and especially transition modelling) are still needed and when available are implemented within this framework. Design of turbomachines duly improved—not just in terms of their actual aero-thermal performance but also in reduced time taken to achieve a given performance (which is arguably worth much more in terms of financial optimization).

It became clear that the next range of problems which had to be addressed were those involving complex but real geometry features (shroud flows, secondary/primary interaction, map-width enhancement slots, etc.). And obviously, real geometry is the entry level for many problems of critical interest: secondary air systems, turbine internal cooling, compressor casing treatment and so on. In addition, the trend is towards coupling of components and simulating their interactions in their actual environments, e.g. shroud flows plus blade–blade flow plus cooling. Figure 5 shows examples of this sort of thing: the secondary air system on the left is a complex system in its own right and is also coupled with the primary flow; their interactions can cause considerable degradation to primary path performance. The turbine cooling system is extremely complex—essentially arbitrary in topology—its performance is absolutely central to the life and integrity of the engine.

By now, there was a large installed set of CFD software systems all based on the BREP paradigm and a large user base and so it was only natural simply to extend these approaches.

The most obvious problem was building a mesh to represent the geometry. Two contrasting methodologies were developed: multi-block structured meshes (e.g. Thompson 1988) and fully unstructured meshes using tetrahedra, hexahedra, prisms and pyramids (e.g. Peraire et al. 1988 or Löhner 1994). The former produce the highest quality meshes from the point of view of solver accuracy but are almost impossible to automate (human intervention is almost always needed) and do not scale well on PC clusters. By contrast, fully unstructured meshes are fast to generate, can be automated and scale well on clusters but do not allow solvers to deliver their highest quality solutions. However, neither meshing approach is inherently parallel in operation—this introduces an absolutely crippling serial bottleneck limiting problem size and setup time.
A critical associated problem is the source of the geometry itself and its import into the meshing system. Often there are heterogeneous geometry sources: some from a CAD engine, some from a set of point strings representing the blade, some just from a drawing. The geometry exported from a CAD system is almost always ‘dirty’, especially if exported via International Graphical Exchange Standard (IGES) but often even if exported direct from the solid modelling kernel (even a ‘solid part’ can be a ‘non-Boolean sum’). This ‘dirtiness’ or lack of ‘water-tightness’ is due to numerical tolerancing issues within the CAD system (often exacerbated by greatly differing scales within the geometry) and lack of numerical compatibility between various geometrical representations—ruled surfaces, NURBS patches, surfaces of revolution, etc. A very substantial effort is needed to clean up and condition this geometry representation so that the meshing software can run satisfactorily.

Figure 6, taken from Dawes et al. (2001), shows a typical solution to these problems and issues. The top row shows the CAD representations of a cooled turbine stage (a) and an aero-derivative gas turbine combustor (b). Even for simple geometries like these, a very great number of patches and curves are needed—each curve must lie ‘on’ (i.e. ‘near’, subject to some tolerance) a patch, each curve must have two patch neighbours and all the patches and curves must thereby be ‘bound’ into a watertight solid—this is the BREP paradigm. These two plots are screen shots from software designed to detect and repair these sorts of geometric compatibilities. The lower two plots in figure 6 show these geometries (shrouded turbine blade on the left and the combustor on the right) ‘captured’ via a triangulated surface mesh; the volume so defined is then filled with a volume mesh—tetrahedra in this case but more usually these days a hybrid mesh of hexahedra and tetrahedra (with pyramids and prisms to manage transitions between them).

All current CFD analysis software systems, whether those commercially available or developed in-house, follow the same BREP paradigm and adopt basically the same approach (FLUENT, CFX, CD-ADAPCO, ...).

3. For complex geometry, has the CFD development trajectory ended up where we need to be?

(a) Analysis versus design

We are now ready to confront the central question posed in this paper and to do that it is important to understand the difference between analysis and design in the context of turbomachinery CFD. Analysis consists of importing a geometry, building a mesh, running a flow simulation and post-processing the results. Hopefully, this then leads to confirmation of existing insight or generation of new understanding. Design is quite different and consists of developing a geometry (which may have a priori unknown form and topology) in the context of a flow, maybe buried deep in an engine, which will deliver the desired functionality in terms of life, performance and so on.

It is crucial to recognize that design involves changing geometry. Equally importantly—but much less well recognized—the flexibility and generality with which the geometry can be changed is the critical enabling factor in our ability to explore the design space properly and change the flow for the better (e.g. Samareh 1999, 2001; Dawes et al. 2003).
How is geometry changed? At the *creative* level it is changed in response to the insight and understanding developed from experience—perhaps obtained from analysis or physical experiment. The rate at which designs can be changed is limited by how quickly this experience can be accumulated. There is much current work on automatic design optimization—and many see this as simultaneously solving the problems of generating this experience faster as well as just improving productivity.

At the *functional* level, design operates generally on three classes of geometry description. The first, and most pragmatic, simply intercepts and edits whatever basic ASCII datasets are in routine use to describe a turbomachinery blade profile—or the parameterized model that writes that dataset (Harvey et al. 2003). The second class assumes a BREP geometry model and edits the boundary patches either by directly modifying control points or solving surrogate equations (like Bloor & Wilson 1990), or using techniques such as ‘free-form deformation’ (Lamousin & Waggenspack 1994; Kellar 2003). The third class acts directly on a solid model description—obviously a parameterized one—as described by Sederberg & Parry (1986) or Haimes 1997 for example.

(b) Analysis with high-order modelling but design with low-order modelling

Low-order modelling like the classic S1–S2 paradigm is very fast, but it is critically dependent on correlated blade element performance (and many correlations in use today date back to the 1960s). The S1–S2 method completely fails to correlate modern design practice—like three-dimensional blade lean and sweep (or realistically account for stall margin, etc.) and so in practice can only be relied on to produce design conservatively and within current practice. The
position is even worse in areas like turbine cooling or for secondary air systems where correlations accounting for all the possible geometrical variations do not exist and would be simply too expensive to imagine acquiring.

By contrast, high-order modelling, like three-dimensional CFD, can address innovative and novel design—with due care with turbulence modelling, calibration, skill and understanding. However, to move away from the clean, primary blade–blade path, high-order modelling needs fully resolved geometry. From the point of view of design, the critical factor is: how fast can the geometry be changed and re-meshed?

The current status in turbomachinery is that analysis is performed with high-order modelling but design is still largely carried out with low-order modelling. This is because the current CFD orthodoxy simply cannot cope with rapid changes to complex geometry.

Figure 6. CAD BREP representations of (a) a turbine housing and (b) a combustor; and surface unstructured meshes capturing (c) a shrouded turbine and (d) a combustor (Dawes et al. 2001).
Assuming someone has created a CAD model in the first place, an orthodox ‘CFD Process’ must contain the following functionality:

— import geometries from CAD (IGES, STEP, STL, Parasolid x_t part, etc.);
— create and heal the geometry representation into a watertight surface;
— create a surface mesh, create a volume mesh;
— run a flow solver (possibly with mesh adaption—which may need geometry); and
— perform post-processing.

This ‘process’ is usually assembled from a combination of ‘best-in-class’ pieces of software—some commercial, off-the-shelf and some custom-written. Figure 7 shows the typical way a CFD process chain is built in industry consisting of the functionality described above all scripted together.

This is where the CFD trajectory has ended up—the current asymptote. For a typical problem—design of a turbine cooling system—the go-around time is approximately two to four weeks. In practice, therefore, one design is analysed—if it is not unacceptable it is passed on to the next stage of the engine design. Within realistic, and ever shortening, industrial product cycle times there is simply not enough time to try any wide-ranging conceptual design or to develop much new experience or understanding for the next design. It is important to emphasize that it is not the asymptotic performance of the approach that is the problem, it is just that it takes so long to deploy that it cannot participate in actual industrial design.

*Phil. Trans. R. Soc. A* (2007)
Why is it so slow? First of all, it is a big challenge to make this a robust, guaranteed and fully human-independent procedure. A number of bottlenecks exist that must be overcome in various ways (Dawes et al. 2001); otherwise, the process cannot be used quickly by a designer that is ever less automated. For example, manual intervention may be needed to import the edited CAD model into the mesh generator or the CAD modelling itself may not permit a sufficiently automated or general or flexible way to edit the model—for example to permit topology (genus) change in the model. The mesh generation may well fail and need careful nursing to produce an acceptable mesh for a really complex geometry—and generating the associated viscous layers remains a nightmare.

Second, in process terms, each piece of software has different input/output requirements and formats—even different interpretations of supposedly vendor-neutral formats like CGNS. The scripting together of this range of software and maintaining this script as each individual piece of software is updated without synchronization with its neighbours are major challenges and expenses.

Third, there are number of serial bottlenecks associated not just with the software inhabitants of the CFD process but with the data transfer between them.

Therefore, the key question to pose is: how much faster could the conventional CFD process actually become? What would it take to reduce a go-around time of four weeks to 4 h or to 4 min?

\[(d)\] Serial versus parallel

Part of the answer must lie in exploiting parallelism. So, where are the serial bottlenecks?

Over the years, the flow solver itself was considered to be the bottleneck and virtually all parallelization strategies and techniques were devised for and applied to the solver. These days, all front rank solvers have good parallel performance and many can sustain that performance on meshes well into the 100 M+ range.

Anyone with real-world experience of large simulations quickly realizes that generating and partitioning the mesh represents a key serial bottleneck. The move towards 64-bit architectures will obviously permit bigger meshes to be generated on a given PC but the speed of the generation will become unacceptably slow (64-bit machines are little different in speed to 32-bit ones but simply overcome the 4-Mb address limit of the latter). There is some research on parallel mesh generation but it is not obvious how to partition efficiently the tried and tested approaches of multi-block structured or Delaunay-based unstructured. Certainly, little is available commercially and current limits are of order 60 M cells. Most approaches to building bigger meshes are like ‘chunking’ whereby the CAD model is cut up into smaller pieces (assuming the availability of suitable tools) and then meshed separately with common inter-domain boundaries to be merged later (but achieving contiguous viscous layers rapidly become a nightmare...in practice this is the absolute limiting factor on meshable geometry size and complexity).

Similar issues of size and speed make post-processing a serial bottleneck that cannot be resolved with the shift to 64-bit architectures. One of the very few available parallel post-processing packages is the pioneering pV3 (Haines 1997).
but even this is limited by the rendering rate (the numbers of triangles per second that the graphics engine can comfortably handle) and would need parallel decimation strategies to be devised and implemented.

Then there is the geometry model itself. It would appear that the current geometry modelling paradigm, based on BREP patches/edges/topology bindings, is inherently serial. Why does this matter? It matters because geometries are getting bigger and more complex (again, 64 bit will increase model size but with little improvement in processing speed) and also because the trend in automatic design optimization is to interact directly with the CAD. Software like CAPRI (Haimes & Follen 1998) enables direct interrogation of the solid modelling kernel underpinning the (hopefully parameterized) CAD model, but for large simulations and/or extensive optimizations everything must run in parallel—already one figure of merit for the optimizer is to minimize the number of CAD licences tied up (figure 8).

Finally, there is the distinctly unglamorous but vital area of data exchange between the various pieces of software in the process. Currently, there are no protocols or data formats that support doing this in parallel.

4. A different paradigm inspired by the games industry

(a) Explicit and implicit geometry

The purpose of this paper is to suggest that a new paradigm is needed, both to allow unlimited, geometric flexibility and to allow parallelism to be fully exploited across the whole CFD process. It is clear that this must be a different paradigm from the one on which the orthodox CFD trajectory is based. In particular, the perspective must change to a geometry-centric one, rather than the current flow-centric viewpoint, and all aspects of the CFD process chain must be inherently parallel.

The conventional CFD trajectory is fundamentally based on a body-fitted BREP paradigm—this is sometimes called ‘explicit’ geometry. However, there is an alternative so-called ‘implicit’ paradigm where a distance field, stored and
manipulated just like any other field variable, represents the geometry. Abandoning the explicit paradigm, and embracing implicit geometry, is the crucial step.

Accordingly, this paper will now describe the building blocks for a quite different approach with at its heart the integration of implicit solid modelling directly with the mesh generation and the flow solution. The following sections will first summarize the key features of solid modelling and then a relatively new field of three-dimensional computer graphics—interactive volume sculpting—together with some highlights of ‘physics-based animation’. Then, the author’s current research into an integrated flow solver/geometry engine will be presented. Finally, some example applications will be described to demonstrate the merit of the new approach.

(b) Solid modelling basics

It will be helpful here first to review the principles of solid modelling. There are three basic types of solid model.

(i) **Constructive solid geometry** (CSG) is based on the definition and manipulation of simple analytical bodies (cube, sphere, ellipse, …) that can be scaled, translated and rotated. The body primitives are then combined via Boolean logic together with inside/outside conventions to enable quite complex solid models to be produced. The advantages are simplicity and compactness of storage; the disadvantages are difficulty in representing the more free-form sorts of shapes encountered in aerodynamics.

(ii) **Boundary representation** (BREP) solid modelling is based on combining patches, edges and topology bindings into a ‘watertight’ solid. Typically, NURBS patches are used and complex, multiple curved surfaces can be stored very economically (the geometry needs only to be reconstructed for viewing or manipulation). The main disadvantages are in producing—and maintaining under manipulation—the watertight bindings and supporting topology (genus) change.

(iii) **Spatial occupancy** solid models consist of Cartesian hexahedral cells (perhaps stored in an octree data structure for efficiency) with cells occupied or vacant or cut (with perhaps some local surface shape data stored). The main advantages are simplicity and generality—topology change is trivially supported—but the disadvantage is the elevated storage overhead compared with BREP and CSG.

Commercial CAD uses mostly BREP with CSG constructs, where possible, coupled with octree-like data structures, if needed, to make searching operations efficient.

The kernel vendors licence access (e.g. UG sits on Parasolid) and direct read of the kernel can interrogate the solid model to return data like curvature, inside/outside, etc. Of particular interest is access to the tessellation of the surface of the solid model that is used to render it. We have used CAPRI (Haimes & Follen 1998), which combines kernel queries with visualization and is implemented for UG (using the Parasolid kernel), Pro/E (Pro/TOOLKIT) and CATIA (CATIAgeom). However, the tessellations themselves are not directly suitable as a flow mesh and
thus must be used as a driver for a more flow oriented mesher. In a more general sense, tessellated surfaces—especially those held in STL or VRML formats—have become the de facto standard geometry representation.

(c) **Topology editing: interactive design via sculpting**

Imagine being able to walk into a virtual reality tunnel within which sat an F1 car with its flowfield displayed and then bending a barge-board, or modifying a wing endplate, with a VR glove and seeing the predicted flowfield respond in real time; imagine being able to make topological changes to a turbine cooling geometry, adding and removing cooling holes with a simple tool, just as if working with clay! It is difficult to imagine achieving these levels of generality with an orthodox CFD process linked to conventional CAD modelling.

However, a very interesting recent development in the field of computer graphics has been real-time sculpting in virtual reality—first proposed by Galyean & Hughes (1991). This draws heavily on the rapidly expanding field of three-dimensional volume graphics, which is itself driven particularly by medical imaging and by animation. The basic principles are very simple: space is divided up into a three-dimensional mesh of cells—‘voxels’; a workpiece and a tool are defined and manipulated as basic spatial occupancy solid models; interactions between the tool and the workpiece result in modified geometry; these interactions can be via adding or removing of material (basic CSG type operations) or spraying of new material on to an existing solid or anything else that can be devised. The whole procedure is similar in concept to sculpting with clay—and quite complex and sophisticated results can be produced (e.g. Bremer et al. 2001).

Of course, the surface of the emerging solid body must be managed carefully. There are two main approaches: one is based on the classic graphics rendering algorithm called ‘marching cubes’, which simply aims to reduce aliasing (Perng et al. 2001); a more sophisticated approach is based on level set techniques (e.g. Baerentzen 2001)—this is capable of faithfully representing multiply curved surfaces. Figure 9 illustrates this via the sculpting of a head—it is impossible to imagine achieving this sort of geometric flexibility with a conventional, explicit BREP paradigm. The level set approach is central and will be described more fully in §4e.

A related and very relevant field is the management of a spatial density field, extracted by medical imagining (scanning) and expressed on a Cartesian mesh as illustrated in figure 10 (taken from Jones & Satherley (2000)). Rendering the distance field at different offsets allows a variety of information to be extracted—this isosurface extraction must be very rapid (a surgeon works in real time) and a variety of very efficient algorithms have been developed that we can draw on. Again, the level set is a central concept.

(d) **Physics-based animation**

Animation for movies pre-dates CFD by 50 years. In the early days, every image was drawn by hand—this was very slow and expensive for films of any length. Next, the ‘master’ drew key images and then the ‘apprentices’ filled in the gaps. Later, aided and inspired by the advent of computers, and especially computer graphics, increasingly complicated interpolation schemes were devised to try to connect the ‘master’ images. About 10 years ago, it was realized that the most effective way to interpolate in a physically realistic manner was to solve
physics-based equations. For scenes involving fluid mechanics, this led to simulations using the Navier–Stokes equations, often for flows with free surfaces. High-quality plots of the resulting unsteady flow, rendered with great artistry, could be directly inserted into the movie. To be economic, these solution methods had to be fast and robust—the results had to look physically realistic, even if in strict scientific terms, they might validate poorly with experiment.
More interesting, however, than the flow solutions (which could easily be improved by the obvious means) was the approach that found most success in dealing with complex geometry. Movie geometries might include the entire set, a castle perhaps, with human actors captured into virtual models and inserted into the action and animated along with the other characters. The set—and the actors—might change at short notice but nevertheless physical realism must not suffer. It was immediately clear that the conventional BREP geometry paradigm would be too slow and inflexible for this sort of work. The answer was to move away from that explicit geometrical world to an implicit one with geometries represented by distance fields and managed and manipulated using level sets—just as described in §4c on virtual sculpting.

A selection of images produced using ‘physics-based animation’ is shown in figure 11. Figure 11a shows a lead and a wooden ball dropped into a container of water (Carlson et al. 2004), (b) shows a wax candle melting (Carlson & Mucha 2002), (c) shows the ‘capture’ of a dancer into a voxelized image for subsequent animation (Hasenfratz et al. 2003), and (d) shows the dust trail of a vehicle (Chen et al. 1999).

More interesting, however, than the flow solutions (which could easily be improved by the obvious means) was the approach that found most success in dealing with complex geometry. Movie geometries might include the entire set, a castle perhaps, with human actors captured into virtual models and inserted into the action and animated along with the other characters. The set—and the actors—might change at short notice but nevertheless physical realism must not suffer. It was immediately clear that the conventional BREP geometry paradigm would be too slow and inflexible for this sort of work. The answer was to move away from that explicit geometrical world to an implicit one with geometries represented by distance fields and managed and manipulated using level sets—just as described in §4c on virtual sculpting.

A selection of images produced using ‘physics-based animation’ is shown in figure 11. Figure 11a shows a lead and a wooden ball dropped into a container of water (Carlson et al. 2004). Figure 11b shows a wax candle melting (Carlson & Mucha 2002) simulated using a liquid with strongly temperature-dependent
viscosity. Figure 11c shows the ‘capture’ of a dancer from stereo photography into a voxelized image for subsequent animation (Hasenfratz et al. 2003)—this is called ‘voxel carving’. Finally, figure 11d shows the dust trail of a vehicle (Chen et al. 1999) computed using a series of physical models including the abrading effect of the tyre through to the convection/diffusion environment created by the vehicle and the wind.

(e) Distance fields, level sets and implicit geometry

When a solid model is represented using ‘spatial occupancy’, each cell, or voxel, can have associated with it the signed distance to the nearest point on the body (or bodies). This is known as a ‘distance field’, \( \phi \). Boundaries are represented as the zero isosurface of the level set. The geometry is now represented by a field variable, \( \phi \), and so this is termed an implicit paradigm and is in contrast to the classic BREP paradigm, which is explicit. This is a fundamental shift in viewpoint.

Sculpting, as outlined above, simply means editing the distance field. Topological changes are trivially supported since the underlying solid model is based on spatial occupancy rather than conventional BREP. Typical solid modelling operations from CSG are Boolean sums of the form

\[
C = A \cup B,
\]

(which forms the union of solid \( A \) with \( B \)) are replicated in distance fields via simple and inexpensive voxelwise logic like the following equation:

\[
d_C = \min(d_A, d_B),
\]

with material added or removed simply by selecting the inside/outside sign on the distance itself.

The level set method (Osher & Sethian 1988; Adalsteinsson & Sethian 1995) has as its key idea the representation of a propagating interface as the zero value of a signed distance function

\[
\phi(x, t > 0) = \pm \rho.
\]

It is easy to show that \( \phi \) has an associated evolution equation

\[
\phi_t + F|\nabla \phi| = 0,
\]

where \( F \) is the speed function in the normal direction (a function of curvature, deposition/etch rate, etc.). At any time, the distance function can be re-initialized by solving \( |\nabla \phi| = 1 \) using simple iterative techniques that mimic time-marching evolution.

5. BOXER: an integrated, parallel, geometry engine, mesh generator, flow solver and post-processor

(a) Algorithmic architecture

The background to the new work presented here is the exploration of the possibilities offered by the integration of the solid modelling directly with the mesh generation and with the flow solution. This research combines ideas from

Phil. Trans. R. Soc. A (2007)
solid modelling (e.g. Haimes & Follen 1998; Samareh 1999, 2001) with virtual sculpting (e.g. Galyean & Hughes (1991); Perng et al. 2001; Baerentzen 2001) combined in the context of a simple, cut-Cartesian mesh flow solver (Bussoletti et al. 1995; Aftosmis et al. 1998). A recent publication (Dawes 2005) set out these building blocks and showed their potential as a rapid prototyping design tool.

The core of the new code (‘BOXER’) is a very efficient octree data-structure acting simultaneously as a search engine, as a spatial occupancy solid model and as an adaptive, unstructured mesh for the flow solver. This provides unlimited geometric flexibility and very robust mesh generation. The solid model is initialized by the import of a tessellated surface from a variety of potential sources (most CAD engines have an STL export) or by direct interrogation of the CAD solid model kernel itself. The solid model is captured on the adaptive, unstructured Cartesian hexahedral mesh very efficiently by cutting the tessellated boundaries using basic computer graphics constructs developed for interactive three-dimensional gaming (aimed at real-time collision detection). This geometry capture is very fast; for example, a body represented by approximately 1M surface triangles can be imported into a mesh of around 11M cells (with six to seven levels of refinement) in approximately 2 min on a single, top-end PC. The spatial occupancy solid model is then sampled as a distance field and managed as a level set; this forms a solid modelling kernel to support the activities of the code. Adaptive mesh refine/de-refine for the flow and for the geometry, via the distance field, enables both moving bodies and topology editing—flow sculpting (Dawes 2005). The algorithmic architecture is illustrated in figure 12.

The associated three-dimensional RANS solver was adapted from an existing unstructured mesh RANS solver (Dawes et al. 2001) with the additional complications of handling hanging nodes and the cut cells. The cut cells are the main disadvantage of the Cartesian approach. For the present, rather exploratory research, the simple ghost cell approach of Viecelli (1971) was adopted as it is very robust; this relies on storing for each cut cell, the local body normal. More accurate approaches are certainly possible and, as shown later in this paper, BOXER also has the ability to delete the cut cells and build layered viscous meshes along body normals.

Figure 12. Algorithmic architecture of BOXER.
(b) Software architecture

The algorithmic architecture described above was chosen precisely because each building block is inherently parallelizable. This research was described in Dawes (2006).

The octree-based cut-Cartesian mesh generation algorithm works very nicely in parallel. The hexahedral cells are cut by local interactions with individual triangles from the STL/VRML input, or with reference to the local value of the distance field. This can be partitioned easily over a number of domains (as can the STL/VRML input itself, which may contain millions of triangles). At the moment, load balancing between domains is static and based on the spatial distribution of the input geometry triangulation—roughly equally split between the available partitions; the next challenge is to introduce dynamic balancing. Communication between domains is relatively small and mainly related to the constraint to maintain cell–cell transitions at no more than h–2h.

The flow solver which is straightforward to make parallel and a simple halo cell methodology was implemented as special ghost cells within the code’s existing data structures.

Post-processing is relatively easy to implement in parallel. Unlike streamlines, cut surface extraction and isosurface extraction, for example, do not even need inter-domain communication. Each partition contributes a set of triangles (with each node storing X, Y, Z and a scalar, like the Mach number), which is then sent to the master process, combined with all the other contributions, and then rendered subject to selected transformations and lighting models. In future work, the child processes will also decimate their contributions to maximize the front end rendering rate.

The underpinning solid modelling kernel, the level set representation of the spatial occupancy solid model, is simply based on manipulating a field variable, the distance field, and so can also be trivially parallelized—as can any tools to manipulate the geometry. This appears to offer this sort of implicit solid modelling kernel a main advantage and opportunity compared with the conventional explicit BREP alternatives.

The software structure developed to integrate these building blocks in parallel was an event queue managed, child–parent–sibling hierarchy implemented in a mixture of C and FORTRAN using OpenGL/glut and MPI. This architecture is illustrated in figure 13.

6. Examples

(a) Turbine cooling hole editing

This first example is based on the important practical problem of turbine cooling. Blades in the high-pressure sections of high-performance aero-engines and land-based gas turbines are very commonly cooled using air from compressor exit delivered via an internal air system to within the blade—in complex serpentine cooling passages—and to the blade surface via film cooling holes. Much research, development and testing go into the design of this critical technology. Inadequate cooling is responsible for a significant fraction of in-service problems and rectifying poorly performing design can be very expensive. The development of BOXER was started with exactly this sort of problem area in mind with the...
ultimate ambition of providing a rapid prototyping tool with sufficient speed, generality and accuracy to allow design to be optimized. This section shows the stages in a simple design study on a generic cooled section.

Figure 14 shows the stages in the ‘capture’ of the blade geometry. The geometry is imported as a tessellated surface and then the octree mesh is generated with five refinement levels. The mesh contains 1.4M cells and took approximately 60 s to generate and 400 Mb of RAM on a 2 GHz laptop PC. The two mesh shots show the total mesh and the mesh within the flow domain. The next stage is to extract the distance field, which is shown as the last shot in figure 14; the distance is clipped for economy to $\pm 5\Delta$, the mesh spacing on the finest level, and red is positive, blue is negative.

This is followed by the sculpting stage. In this example, figure 15, a cylindrical tool, shown in blue is selected and via a six degrees of freedom motion applied via the front-end GUI, scaled and manoeuvred towards the blade suction side; a new film cooling passage ‘drilled’ into the internal air supply. This operation takes approximately 30 s.

Then, the distance field associated with the tool and that of the current blade geometry (the zero value of the distance is rendered green in the figure) are combined and a new spatial occupancy solid model is created that is ready for the next flow solve (using the previous solution, which presumably inspired the need for the extra cooling, as its guess). This stage takes approximately 1.5 min and figure 16 shows some shots of this.

The final figure in this example, figure 17, shows a couple of sample shots of the predicted flowfield—the total temperature field near the suction side film cooling holes and the mid-span Mach number field. No more flow analysis will be attempted here (indeed it is clear that the new hole is blowing too strongly) since the aim of this paper is to describe the building blocks and demonstrate their integrated assembly.
It is important to stress that all the stages represented in figures 14–17 were carried out within a single piece of software, all under the control of a GUI—from the moment the initial geometry is captured there is no need to pass data to any other code—nor return to the CAD engine. It is believed that the dramatic speed up afforded to the designer will allow much more creative and extensive conceptual design within a realistic time-scale.

(i) Parallel performance

The parallel mesh generation performance of Boxer will be illustrated in the context of the simple case of a shrouded turbine blade, which can be adequately resolved with a relatively coarse mesh of approximately 2.27 M cells. The hardware
deployed was a simple laptop, front-end connected via the office network to a cluster of 16 PCs. Each of these PCs has a 2.5 GHz Pentium IV processor with only 1 Gb RAM; they are interconnected by simple 1 Gb s$^{-1}$ Ethernet in a dedicated circuit. Figure 18 shows the mesh generation time in seconds on

*Phil. Trans. R. Soc. A* (2007)
1, 2, 4, 8 and 16 PCs together with a reference ‘theoretical’ performance. As can be seen, for this small mesh the generation time becomes communication-bound beyond 8 PCs. Nevertheless, 10 s is a remarkably small time for a mesh for this class of geometry.

(ii) Turbine shroud editing

The next example is based on the important practical problem of turbine shroud design. This section shows the stages in a simple design study on a generic turbine shroud via two example geometry editing sessions.

Figure 19 shows the stages in the ‘capture’ of the blade geometry. The geometry is imported as a tessellated surface and then the octree mesh is generated with three refinement levels. The two mesh shots show the total mesh and the mesh within the flow domain. The next stage is to extract the distance...
field, which is shown as the third shot in figure 15; the distance is clipped for economy to \( \pm 5\Delta \), the mesh spacing on the finest level, and red is positive, blue is negative. The last plot shows the zero isosurface of the level set—the blade.

The first sample edit is the complete removal of the forward fin. This is a straightforward topological edit with BOXER’s integrated approach. Figure 20 illustrates the process. A cylindrical cutter tool is selected from the GUI and then manoeuvred via a six degrees of freedom system into the desired location. Then the distance field is edited (the zero level set, the blade is shown rendered green in figure 20). The new blade shroud surface is created and the mesh rebuilt. This takes place in an interactive manner, just like in a computer game, with an elapsed time of approximately 15 s.

Figure 17. Two snapshots of the new flow solution: (a) total temperature and (b) Mach number.

Figure 18. Capture of a generic industrial gas turbine geometry on a cluster of 8 PCs, (a) is coloured by processor number and (b) shows the mesh generation time versus number of processors.

W. N. Dawes
Phil. Trans. R. Soc. A (2007)
The second example shows how a new fin, again a topology-changing operation, can be readily created. Figure 21 shows the progress of the edit; in each case, the zero isosurface, which represents the surface of the blade geometry, is rendered green and the tool is blue. A deposition tool is selected from the GUI and manoeuvred via six degrees of freedom motion to lie in the desired place on the upper side of the shroud. The tool is then incrementally drawn away from the shroud, leaving behind new solid geometry. The new blade shroud surface and the associated three-dimensional mesh are updated automatically after each increment. This edit takes approximately 2 min—most time being associated with manoeuvring the tool carefully into position (a scripted cutter motion—driven by an optimizer say—would be very much quicker).

*Phil. Trans. R. Soc. A* (2007)
Finally, one major remaining issue to address is, therefore, that of the quality of the flow simulation itself and this depends critically on resolving the key vulnerability of the proposed approach managing the cut cells. The cut cells are the single biggest remaining problem—however, if this problem could be managed well enough to allow predictions with the same quality as standard body fitted meshes then the advantages from a process viewpoint of cut-Cartesian based approaches would be overwhelming. This section shows recent progress in this area—a novel but integrated approach to the generation of viscous layer meshes in place of the cut cells; more information is given in Dawes et al. (2007).

Figure 20. Complete removal of the forward fin: (a) the initial geometry and mesh, (b) the zero level, the blade, (c) the cutter tool in position, and (d) the new geometry.

(b) The cut cells and viscous layers

Finally, one major remaining issue to address is, therefore, that of the quality of the flow simulation itself and this depends critically on resolving the key vulnerability of the proposed approach managing the cut cells. The cut cells are the single biggest remaining problem—however, if this problem could be managed well enough to allow predictions with the same quality as standard body fitted meshes then the advantages from a process viewpoint of cut-Cartesian based approaches would be overwhelming. This section shows recent progress in this area—a novel but integrated approach to the generation of viscous layer meshes in place of the cut cells; more information is given in Dawes et al. (2007).
For unstructured body fitted meshes there is an extensive literature associated with the generation of high-quality viscous layers. The work of Kallinderis & Ward (1993) is an excellent example. The key to generating good layer meshes is in successfully managing the body normals—especially in corners or near extreme convexity.

Figure 21. Creating a new fin with a deposition tool: (a) shows the initial geometry, (b–d) show the tool depositing new solid, and (e–f) show the final geometry and a mesh cut.

For unstructured body fitted meshes there is an extensive literature associated with the generation of high-quality viscous layers. The work of Kallinderis & Ward (1993) is an excellent example. The key to generating good layer meshes is in successfully managing the body normals—especially in corners or near extreme convexity.

Phil. Trans. R. Soc. A (2007)
Figure 22. Viscous layer meshes generated from the background level set showing meshes for geometries before and after geometry editing.
For cut-Cartesian meshes, a number of approaches have been published—the most notable are probably Delanaye et al. (1999) and Wang & Chen (2002). The former attempts to marry and merge conventional advancing viscous layers (like Kalinderis et al. 1993) with Cartesian cells. The latter strips out cut cells and connects nodes to closest points on the body—in principle, this should guarantee no crossing of mesh lines.

The current approach is based on exploiting the nice properties of the level set geometry kernel that underpin the BOXER software paradigm. In particular, the key new idea is to integrate fictitious ‘particle paths’ through the vector field formed by the gradient of the distance field, \( \nabla (D) \), which is directly related both to the body normals \( n = \nabla (D) \) and curvature \( k = \text{div} (n) \). These ‘paths’ cannot, in principle, cross and respond to regions of high curvature, like corners, in a very natural way. These ‘paths’ combined with isosurfaces of the distance field itself then form natural construction lines for the mesh layers. Figure 22 illustrates a variety of viscous layer meshes generated from a level set in this way—including generation before and after the geometry editing procedure.

7. Concluding remarks

This paper has looked at the development of CFD for application to practical turbomachinery simulation from a historical perspective and within the framework of a series of paradigm shifts followed by asymptotes. The particular focus of this paper is real, complex geometries and the case is made that the current ‘CFD orthodoxy’, a scripted process chain of best-in-class software, is in fact not the way forward.

This paper has attempted to show that it is possible to construct an alternative, practical, geometry-centric simulation system that is inherently parallel and scalable by integrating a solid modelling kernel, mesh generator, flow solver and post-processor within a single piece of software. The key conceptual step is to replace explicit geometry with implicit representations.

Furthermore, it is shown that the serial bottlenecks that represent an increasing restriction to more orthodox CFD processes as geometries and problem sizes get bigger and more complex can be overcome by this radical rethink based on its inherent parallelism.

Future work will try to exploit this new paradigm for design optimization in three dimensions in the spirit of rapid prototyping.

References


Baerentzen, A. 2001 Volume sculpting: intuitive, interactive 3D shape modelling. IMM.

Phil. Trans. R. Soc. A (2007)


Denton, J. D. 1987 *Computational methods for turbomachinery flows*. Lectures to ASME Turbomachinery Institute, Ames, IA.


Phil. Trans. R. Soc. A (2007)


