COMPUTATIONAL FLUID DYNAMICS (CFD)

Computational Fluid Dynamics (CFD) is concerned with obtaining numerical solution to fluid flow problems by using computers. The advent of high-speed and large-memory computers has enabled CFD to obtain solutions to many flow problems including those that are compressible or incompressible, laminar or turbulent, chemically reacting or non-reacting.

The equations governing the fluid flow problem are the continuity (conservation of mass), the **Navier-Stokes** (conservation of momentum), and the energy equations. These equations form a system of coupled non-linear partial differential equations (PDEs). Because of the non-linear terms in these PDEs, analytical methods can yield very few solutions. In general, closed form analytical solutions are possible only if these PDEs can be made linear, either because non-linear terms naturally drop out (e.g.., fully developed flows in ducts and flows that are inviscid and irrotational everywhere) or because nonlinear terms are small compared to other terms so that they can be neglected (e.g.., creeping flows, small amplitude sloshing of liquid etc.). If the non-linearities in the governing PDEs cannot be neglected, which is the situation for most engineering flows, then numerical methods are needed to obtain solutions.

CFD is the *art* of replacing the differential equation governing the Fluid Flow, with a set of algebraic equations (the process is called discretization), which in turn can be solved with the aid of a digital computer to get an *approximate* solution. The well known discretization methods used in CFD are Finite Difference Method (FDM), Finite Volume Method (FVM), Finite Element Method (FEM), and Boundary Element Method (BEM).

FDM is the most commonly used method in CFD applications. Here the domain including the boundary of the physical problem is covered by a *grid* or *mesh*. At each of the interior grid point the original Differential Equations are replaced by equivalent finite difference approximations. In making this replacement, we introduce an error which is proportional to the size of the grid. This error can be reduced by making the grid size smaller to get an accurate solution within some specified tolerance.

This brief description is hardly sufficient to demystify the seemingly esoteric world of CFD.

FEM - A brief overview :

Students often ask me about the difference between FEM and CFD. Many of them had an impression that they are related to each other. Below I try to give a brief and a nonrigorous description of FEM.

Finite Element Method (FEM) is a mathematical (numerical) tool (just like Finite Difference Method) used to solve complex physical problems which are not amenable to classical techniques of mathematics. It has found it's applications in the fields of Structural Design, Vibration Analysis, Fluid Dynamics, Heat Transfer, and Magneto hydrodynamics to name a few.

The basic idea in FEM analysis of field problems is as follows:

The solution domain is discretized into number of small sub regions (i.e., Finite Elements).
Select an approximating function known as interpolation polynomial to represent the variation of the depende variable over the elements.
The integration of the governing differential equation (often PDEs) with suitable Weighting Function, over e elements to produce a set of algebraic equations-one equation for each element.
The set of algebraic equations are then solved to get the *approximate* solution of the problem.

In principle, any well posed Boundary Value Problem can be solved by the techniques of FEM.

CFD - A Brief overview :

Computational Fluid Dynamics (CFD) provides a good example of the many areas that a scientific computing project can touch on, and its relationship to Computer Science. Fluid flows are modeled by a set of partial differential equations, the **Navier-Stokes equations**. Except for special cases no closed-form solutions exist to the Navier-Stokes equations, and this fact was one of the motivations John von Neumann provided for the development of electronic computers.

Solving a particular problem generally involves first discretizing the physical domain that the flow occurs in, such as the interior of turbine engine or the radiator system of a car. This discretization is straightforward for very simple geometries such as rectangles or circles, but is a difficult problem in CAD for more complicated objects. Currently automatic ``mesh generators" are simply not adequate, requiring extensive investment of time on the part of the scientist or engineer. This leads to problems in human-computer interfaces (HCI) and CASE tools, as well as fundamental problems in graph theory since the resulting discretization gives a mesh that is best dealt with as a graph.

On the discretized mesh the Navier-Stokes equations take the form of a large system of nonlinear equations; going from the continuum to the discrete set of equations is a problem that combines both physics and numerical analysis; for example, it is important to maintain conservation of mass in the discrete equations. At each node in the mesh, between 3 and 20 variables are associated: the pressure, the three velocity components, density, temperature, etc. Furthermore, capturing physically important phenomena such as turbulence requires extremely fine meshes in parts of the physical domain. Currently meshes with 20 000 to 2 000 000 nodes are common, leading to systems with up to 40 000 000 unknowns.

That system of nonlinear equations is typically solved by a Newton-like method, which in turn requires solving a large, sparse system of equations on each step. Sparsity here means that the matrix of coefficients for the linear system consists mainly of zeros, with only a few nonzero entries. With 4.0e7 unknowns, clearly we cannot store the matrix as a 2D array with 1.6e15 entries! Storing the coefficients requires development of efficient data structures which require little overhead storage but allow the necessary manipulations to be performed efficiently.

Methods for solving large sparse systems of equations are a hot topic right now, since that is often the most time-consuming part of the program, and because the ability to solve them is the limiting factor in the size of problem and complexity of the physics that can be handled. Direct methods, which factor the matrices, require more computer storage than is permissible for all but the smallest problems. Iterative methods use less storage but suffer

from a lack of robustness: they often fail to converge. The solution is to use preconditioning; that is, to premultiply the linear system by some matrix that makes it easier for the iterative method to converge.

CFD problems are at the limits of computational power, so parallel programming methods are used. That brings in the research problem of how to partition the data to assign parts of it to different processors; usually domain decomposition methods are applied. Domain decomposition is often expressed as a graph partitioning problem, namely finding a minimum edge cut partitioning of the discrete mesh, with roughly the same number of nodes in each partition set. This is an NP-hard problem, so rapid heuristics are used to get quick and dirty solutions. An additional problem with parallel programming is that the better methods for solving the resultant linear systems often have inherently sequential characteristics, while parallel solution methods are not robust enough to tackle real world problems.

Once the solution is found, analyzing, validating, and presenting it calls into play visualization and graphics techniques. Those techniques are useful for more than just viewing the computed flow field. Visualization can help with understanding the nature of the problem, the interaction of algorithms with the computer architecture, performance analysis of the code, and, most importantly, debugging!

This example of CFD indicates where computer science comes into play. graph theory and algorithms, computational complexity, numerical analysis, parallel programming, graphics and visualization are all needed here.

Governing Equations:

1. General; Compressible and Viscous: Continuity $\partial\rho/\partial t + \partial(\rho u)/\partial x + \partial(\rho v)/\partial y + \partial(\rho w)/\partial z = 0$	1a
$\partial\rho/\partial t + \partial(\rho u_i)/\partial x_i = 0$	1b
Momentum $\rho[\partial u/\partial t + u\partial u/\partial x + v\partial u/\partial y + w\partial u/\partial z] = \rho B_x - \partial p/\partial x - (2/3)\partial/\partial x[\mu(\partial u/\partial x + \partial v/\partial y + \partial w/\partial z)]$ $+ 2\partial/\partial x(\mu\partial u/\partial x) + \partial/\partial y[\mu(\partial u/\partial y + \partial v/\partial x)] + \partial/\partial z[\mu(\partial u/\partial z + \partial w/\partial x)]$ $\rho[\partial v/\partial t + u\partial v/\partial x + v\partial v/\partial y + w\partial v/\partial z] = \rho B_y - \partial p/\partial y - (2/3)\partial/\partial y[\mu(\partial u/\partial x + \partial v/\partial y + \partial w/\partial z)]$ $+ 2\partial/\partial y(\mu\partial v/\partial y) + \partial/\partial z[\mu(\partial v/\partial z + \partial w/\partial y)] + \partial/\partial x[\mu(\partial v/\partial x + \partial u/\partial y)]$ $\rho[\partial w/\partial t + u\partial w/\partial x + v\partial w/\partial y + w\partial w/\partial z] = \rho B_z - \partial p/\partial z - (2/3)\partial/\partial z[\mu(\partial u/\partial x + \partial v/\partial y + \partial w/\partial z)]$ $+ 2\partial/\partial z(\mu\partial w/\partial z) + \partial/\partial x[\mu(\partial w/\partial x + \partial u/\partial z)] + \partial/\partial y[\mu(\partial w/\partial y + \partial v/\partial z)]$	2a
$\partial(\rho v_i)/\partial t + \partial(\rho v_i v_j)/\partial x_j = \rho B_i - \partial p/\partial x_i - \partial/\partial x_i\,[2/3\mu(\partial v_j/\partial x_j)] + \partial/\partial x_j\,[\mu(\partial v_i/\partial x_j + \partial v_j/\partial x_i)]$	2b

2. Incompressible and Viscous: Continuity $\partial u/\partial x + \partial v/\partial y + \partial w/\partial z = 0$	3a
$\partial v_i/\partial x_i = 0$	3b
Momentum $\rho[\partial u/\partial t + u\partial u/\partial x + v\partial u/\partial y + w\partial u/\partial z] = \rho B_x - \partial p/\partial x + \mu[\partial^2 u/\partial x^2 + \partial^2 u/\partial y^2 + \partial^2 u/\partial z^2]$ $\rho[\partial v/\partial t + u\partial v/\partial x + v\partial v/\partial y + w\partial v/\partial z] = \rho B_y - \partial p/\partial y + \mu[\partial^2 v/\partial x^2 + \partial^2 v/\partial y^2 + \partial^2 v/\partial z^2]$ $\rho[\partial w/\partial t + u\partial w/\partial x + v\partial w/\partial y + w\partial w/\partial z] = \rho B_z - \partial p/\partial z + \mu[\partial^2 w/\partial x^2 + \partial^2 w/\partial y^2 + \partial^2 w/\partial z^2]$	4a
$\rho[\partial v_i/\partial t + \partial(v_i v_i)/\partial x_i] = \rho B_i - \partial p/\partial x_i + \partial/\partial x_i [\mu \partial v_i/\partial x_i]$	4b

Note: Equations 1b, 2b, 3b, and 4b are written in Cartesian tensor form.

Source:

http://www.oocities.org/venkatej/mech/CFD.html