1 CFD in Perspective

Computational fluid dynamics, usually abbreviated CFD, is the subject in which the calculation of solutions to the basic equations of fluid mechanics is achieved using a computer. In this sense, CFD is a subfield of fluid mechanics, but one that relies heavily on developments in fields outside of fluid mechanics proper, in particular *numerical analysis* and *computer science*. CFD is sometimes called numerical fluid mechanics.

It is a prevailing view that the computer revolution has produced a "third mode" of scientific investigation, often called *scientific computing* or *computational science*, that today enters on a par with the more established modes of (theoretical) analysis and (laboratory) experimentation. CFD is this new "third mode" of doing fluid mechanics.

1.1 The Nature of CFD

Our objective is to explore CFD from a basic science perspective. We take as a given the power and elegance of analytical reasoning in fluid mechanics, and the precision and persuasiveness of well-executed experiments and observations. We wish to demonstrate that CFD has ingredients from both these traditional ways of doing fluid mechanics, and, furthermore, injects a set of unique viewpoints, challenges and perspectives. The subject has many very new aspects, due in large part to the rapid and recent development of computer technology. Yet it also has considerable historical and intellectual depth with many contributions of great beauty and profundity that derive directly from cornerstone developments in mathematical analysis and physical theory.

There is often a very special flavor to a CFD investigation. Sometimes this arises from the unique combination of trying to describe a continuum using a finite set of states (and, thus, ideas from discrete mathematics). On other occasions one has the challenge to address a question that is still too difficult for conventional analysis and is largely inaccessible to laboratory experiments. An example of this kind is whether the three-dimensional Euler equations for incompressible flow generate singularities in a finite time starting from smooth initial data. In a computation we can often impose a constraint or symmetry that is very difficult to achieve or maintain experimentally, such as exactly two-dimensional

flow. In a computation we can increase the nonlinearity in the problem gradually, from a regime conforming to a linearized description, that is usually amenable to analysis, to a weakly nonlinear regime (where analysis typically is much harder), and finally, to a fully nonlinear range (where analytical understanding may be altogether lacking). We usually have control over initial and boundary conditions to a degree of precision unimaginable in the laboratory. In a computation we can realize with ease experimentally dubious yet theoretically useful approximations, such as potential flow or periodic boundary conditions. Constitutive relations for the fluid concerned, e.g., non-Newtonian fluids, can be stated exactly and varied more or less at will. One might say that computational studies of fluid motion are not constrained by reality! For many computational methods flow geometries can be changed quickly and with great flexibility. Theoretically motivated experiments such as turning certain terms on or off in the equations may be done. And with the detail available in most CFD investigations stunning images of flow quantities can be computed and displayed for later scrutiny and analysis. Indeed, the uncritical use of color graphics in CFD investigations has led some cynics to state that the "C" in CFD stands for "colorized" or "colorful."

Adopting a computational outlook on problems often reveals totally new and sometimes unexpected aspects. We shall see that the process of conducting numerical experiments on a computer has a lot in common with its laboratory counterpart, while the tools employed in setting up the calculation are those of theory. The laboratory experimentalist may be confident of manipulating the right quantities. The numerical experimentalist, on the other hand, is equally assured of manipulating the right equations! The results of laboratory and CFD experiments often complement one another in remarkable ways, as we shall have occasion to illustrate. This complementarity is part of the richness and allure of our subject.

Done correctly, CFD involves more than taking a code package off the shelf and running it to obtain a numerical answer to a pressing engineering or environmental problem. The capabilities of CFD in this respect are, of course, responsible for the leverage that the subject commands in the allocation of societal resources, and should in no way be underestimated. The true intellectual fascination of the subject lies, however, in the interplay between the physics and mathematics of fluid motion, and the numerical analysis and computer science to which these give rise. This multi-faceted nature of CFD ultimately is what gives the subject its attraction and from which future developments of the subject will spring. Hamming (1973) concludes with a philosophical chapter on such matters, the main thesis of which is: "The purpose of computing is insight, not numbers" – a very interesting notion for a monograph on numerical methods.

Because fluid mechanics is ruled by well-known partial differential equations, viz. the Navier–Stokes equations and specializations thereof, much more pragmatic visions of what CFD is abound. An applied mathematician or numerical analyst might see CFD simply as part of numerical partial differential equations (PDEs). An engineer working with one of the several available comprehensive,

commercial fluid dynamics packages might look upon CFD as a part of computer aided design. A meteorologist might classify CFD as simply another tool for weather and climate prediction. There are powerful arguments to support all these points of view. For example, in engineering the economics of conducting CFD-based design calculations versus full-scale construction and testing needs little in the way of general epistemology to promote the field! Similar arguments pertain to aspects of weather prediction, such as storm warnings. More fundamentally, numerical studies of climate dynamics, large-scale oceanography, convective and magnetohydrodynamic processes in Earth's interior, flow problems in astrophysics, etc., allow researchers to conduct *virtual experiments* in those fields, as they often are the only possible experiments. With the recent interest in global warming and the observation of the ozone hole, computer experiments on atmospheric dynamics are taking on an entirely new and societally urgent dimension.

Because so many different scientists with different outlooks and different objectives view CFD as everything from a tool to a basic discipline in its own right, texts on CFD vary widely. Most texts assume that the reader is already convinced of the merits of the CFD approach, and basically wants to improve technique. Hence, detailed discussions of computational methodologies are presented, often with code listings and comparisons in terms of accuracy and efficiency. The present volume starts at a considerably more elementary level, and attempts to build up a general philosophy of CFD as various technical issues are introduced. This philosophy is, we believe, important in establishing CFD solidly within the field as an intellectual equal to many of the profound physical and mathematical subjects that fluid mechanics is so well known for.

1.2 Overview of the Book

Before embarking on any detailed discussions it may be useful to give a brief outline of what follows. In this chapter we define some of the *basic concepts* of CFD and delineate the *intrinsic limitations* that arise when trying to simulate fluid motion with a large number of degrees of freedom on a finite machine. The evolution of computer hardware is of interest to this discussion and is briefly traced.

In Chapter 2 we discuss the general idea of a discrete mapping. Any computer representation of fluid motion with discrete steps in time can be viewed as such a mapping, and our study of mappings may be seen as a paradigm of CFD. Furthermore, the study of mappings, usually computer assisted, has revealed many fascinating features, some of which have a direct bearing on fluid flow. The coding required to compute mappings is usually very simple, and the topic thus allows us to get started on computer exercises with elementary code that yields interesting output. We encounter the wide-ranging topics of chaos and

fractals, which arise so naturally in fluid mechanics and have firm roots in the subject.

Chapters 3 and 5 are devoted to ordinary differential equations (ODEs). An ODE can be classified as *initial*, *boundary* and *eigenvalue problem*. The mathematical nature of these problems are so different that they warrant different numerical techniques. As we briefly review at the beginning of these chapters, techniques for solving these different types of ODEs are well in hand. Chapter 3 will cover initial value ODEs, where we will consider classical Adams–Bashforth, Adams–Moulton and Runge–Kutta family of time integration techniques. Chapter 5 will cover boundary and eigenvalue ODEs, where we will consider shooting and relaxation methods to these problems. As a preliminary, in Chapter 4 we will consider spatial discretization schemes. We will cover finite difference and compact difference schemes for spatial differentiation, and Simpson and quadrature rules for spatial integration. The concept of discrete derivative operators in the form of matrices will be introduced in this chapter.

A surprising number of interesting fluid mechanics problems present themselves in the format of a system of ODEs. We discuss by way of example the shape of capillary surfaces (which historically turns out to be the problem that motivated the development of what we today call the Adams–Bashforth method), the Blasius and Falkner–Skan boundary layer profiles, the Kármán solution for a rotating disk, the Graetz problem of thermal development in a pipe flow, onset of Rayleigh–Bénard convection from linear instability, the Orr–Sommerfeld equation for the stability of parallel shear flows, the Rayleigh–Plesset equation for collapse of a gas-filled cavity in a liquid, the Kelvin–Tait–Kirchhoff theory of motion of a solid body in irrotational, incompressible flow, the equations for interacting point vortices in two-dimensional flow, and the topic of chaotic advection. Meaningful computer exercises on all these can be completed with rather brief coding, and should help in developing an appreciation for the mode of work in CFD.

While finite difference methods work with spatially localized approximations for derivatives and field values, methods based on functional expansions use global representations in which the unknown field is expanded in a series of known modal shapes with variable coefficients. Methods based on such representations include finite element methods and, in particular, spectral methods. Chapter 6 is mainly concerned with spectral methods, although the theory of functional expansions is presented in rather general terms. The fast Fourier transform (FFT) algorithm is explained. The theory and utility of Chebyshev polynomials is discussed.

In Chapter 7 aspects of the theory of numerical solutions to partial differential equations are given. This chapter, in a sense, is the starting point of many advanced books on CFD. Thus, much more elaborate treatments of this material may be found in any of several CFD texts. Again, the intent is to provide an introduction that includes enough of the details to be interesting and to give you, the reader, an appreciation for some of the issues, yet avoid the comprehensive treat-

ment that the expert user of these techniques will need to acquire. The issue of stability, first mentioned in the context of mappings in Chapter 2, is re-examined, and plays an important role in structuring the discussion. Computer exercises for the simpler methods can be carried out with a nominal programming effort. In Chapter 8 we consider multi-dimensional PDEs. In this chapter we finally address direct and iterative numerical methods for the Navier–Stokes equations. In this chapter we introduce the powerful time-splitting or operator-splitting methodology, which reduces the Navier–Stokes equation into a set of Helmholtz and Poisson equations. We also briefly cover spectral methodology for PDEs. The Navier–Stokes equations are written out in spectral form. In principle, the material covered here should allow the reader to construct a spectral code for the three-dimensional Navier–Stokes equation in a periodic box. Finally, in Chapter 8 we also briefly cover CFD methodologies for solving fluid flow in complex geometries. The first approach considered is solving Navier–Stokes equations in a curvilinear body-fitted grid. As an alternative we also consider two Cartesian grid methods: sharp interface methods and immersed boundary techniques.

1.3 Algorithm, Numerical Method, Implementation and Simulation

We start by introducing some basic concepts. The first is **algorithm**. The word has the curious etymology of being derived from the name of the Arabic mathematician *Muhammed ibn-Musa al-Khowarizmi*, who flourished around 800 AD. Initially algorithm simply meant the art of calculating using Hindu–Arabic numerals. The modern meaning of the word is: *a systematic procedure for performing a calculation.* An algorithm is thus an independent construct – a set of rules – that gives the principle or strategy for carrying out a calculation. Algorithms exist quite independently of any implementation on a computer.

A well-known example of an algorithm is the sieve of Eratosthenes, dating from the third century BC, for finding all prime numbers between 1 and N. The algorithm consists of a series of steps. First we construct a table of all integers from 2 to N. Then:

(1) Cross out all multiples of 2 (except 2 itself).
(2) Find the first number after 2 that has not been crossed out (i.e., 3).
(3) Cross out all multiples of 3 (except 3 itself).
(4) Find the first number after 3 that has not been crossed out (i.e., 5).
(5) Cross out all multiples of 5 (except 5 itself).

And so on until the table is exhausted. We will be left with the primes $2, 3, 5, 7, \ldots$. Clearly the steps can be grouped together in a loop:

(n) Cross out all multiples of p (except p itself).
($n+1$) Find the first number after p that has not been crossed out, call it p'. Repeat the above with p' instead of p until $p \geq N$.

The algorithm can now be assessed with regard to efficiency, robustness, etc. For example, in this case an immediate improvement in speed is realized by stopping when the integer that is having its multiples eliminated exceeds $\sqrt{N}$, rather than N, since further testing accomplishes nothing.

On a modern computer system the lowest level algorithms for addition and multiplication are implemented in hardware. Some of the algorithms for often-required tasks, such as the computation of common functions (cos, sin, tan, log, etc.) or vector–matrix operations, are coded in highly optimized form and collected in libraries that are called as the default with the compiler/loader. Yet other algorithms are collected in libraries, for which the source code in a high-level language, such as FORTRAN or C, is often accessible. A good example is BLAS – a library of *Basic Linear Algebra Subprograms* (Dongarra *et al.*, 1990), which has become the *de facto* standard for elementary vector and matrix operations. Highly optimized BLAS modules that take full advantage of the machine architecture can be found on most parallel computers. Finally, you, the CFD programmer, will have to provide those algorithms that are particular to the problem you wish to compute.

The algorithms that perform standard calculus operations, such as differentiating or integrating a function, are treated in texts on numerical analysis. Many of them have keyword names, such as *trapezoidal rule*, *predictor–corrector method*, and *fast Fourier transform*. Others are named for their originators, such as *Gauss–Legendre quadrature*, *Simpson's rule*, and the *Newton–Raphson method*. From the above names it is quite clear that the terms *algorithms* and *methods* have traditionally been used interchangeably. CFD presumes and utilizes much of this material, and we shall only pause to treat some of it in detail. In most cases we shall refer to treatments in the numerical analysis literature.

It may be useful to point out that successful algorithms and elegant mathematics do not always go hand in hand. An elegant mathematical formulation may not lead to an efficient calculation procedure, or it may not have suitable stability properties, or it may require excessive storage. For example, in the relatively simple problem of solving systems of linear equations an elegant mathematical solution is given by *Cramer's algorithm*, where the solution is written in terms of ratios of certain determinants. However, the direct evaluation of a determinant of an $N \times N$ system requires some $N!$ operations. It is not difficult to see that this operation count can be beaten handily by the elementary process of successive elimination that one learns well before encountering determinants. In numerical analysis this method is called Gaussian elimination.

A collection of algorithms and basic numerical methods are often needed to perform more complex mathematical operations. For example, common problems in linear algebra, such as solving systems of equations, linear least square problem, singular value decomposition, and eigenvalue and eigenvector computation, require more basic vector–vector and matrix–vector operations. Transportable libraries in linear algebra, such as LAPACK, are built on top of BLAS routines. There have been a number of powerful numerical libraries, such as Portable

Extensible Toolkit for Scientific Computation (PETSc) that have been widely accepted and used in the CFD community. They provide ready-to-use algorithms and methods that are well designed for parallel computing.

We shall then think of a **numerical method** for CFD as a collection of many elementary algorithms and methods. This viewpoint has many potential advantages. First of all this provides a modular approach which greatly simplifies the programming and debugging of the numerical method. This approach also allows for easy replacement of an individual algorithm by another equivalent algorithm as and when necessary. Furthermore by using standard packages from libraries, such as PETSc and LAPACK, the user is relieved of having to code standard algorithms in common use. Full attention can therefore be focused on developing algorithms specific to the problem at hand.

What determines the merit of a given CFD method are typically the algorithms that transcribe the continuum PDEs describing fluid motion into a set of algebraic equations and solve them. This may be accomplished in a variety of ways, all of which, in general, incur some kind of approximation error. The most common approach is for the fluid domain to be fitted with a grid and the amplitudes of the fields of pressure, velocity, etc., at the nodes of the grid chosen as basic variables. This procedure is used in *finite difference methods*. Alternatively, one may transform the pressure and velocity fields to a set of amplitudes that enter in a series expansion in terms of modes of fixed shape. Such an expansion, of which the Fourier series may be the best-known example, is called a *Galerkin expansion*. Truncating the expansion and working with only a finite number of amplitudes leads to *spectral methods* (when the modes that we expand in are global orthogonal functions) and to *finite element methods* (when the modes are local low-order polynomials).

A less common but often very powerful approach is to abandon the Eulerian view altogether and use the computer to follow (many) individual fluid particles. In this way one arrives at any one of a variety of *particle methods*. Particularly useful are representations where the particles traced carry some important attribute of the fluid motion, such as vorticity, or delineate a physically significant boundary in the fluid.

In general, a set of ODEs involving grid values or Fourier amplitudes or particle positions results. Except for a few special cases, these provide only an approximate representation of the PDEs from which they came. We speak of having performed a **discretization** of the continuum equations in space. As seen above there are usually several choices here. Some may be arguably better than others, and considerations of the relative capabilities and merits of different methods make up much of CFD. The numerical method must further contain algorithms for handling the integration in time of the coupled set of ODEs. Again there are a variety of choices, and again the continuous variable time must be discretized in some fashion. For example, integration of the discretized equations can be done either *explicitly* or *implicitly*, and the time and space discretizations can be either coupled or decoupled. The success of a particular method usually depends

on a balance between purely numerical considerations and considerations arising from the underlying physics of the flow. The interplay of physics and numerics is often very important.

A numerical method turned into a computer program that will compile and execute on some machine and produce suitable output we shall call an **implementation** of the method. Implementations can vary considerably depending on the skill of the programmer. Implementations may be written in different computer languages. Implementations might depend on the machine architecture of the computer as well. For optimal performance, the choice of algorithms and methods needs to depend on the vector or parallel nature of the computer. The speed of the computation critically depends on these choices. The implementations may have many diagnostics built in that continually check the calculation for consistency and reliability. For example, if some flow quantity is known to be conserved in time by the PDE and/or the ODEs resulting from the discretization, it would be prudent of an implementation to monitor it, and a good implementation will have such a monitoring capability. There are many options available when it comes to the questions of how to interact with the program, i.e., how the program handles the inputting of variables and parameters and how the program presents output in specific numbers, tables and diagrams. Given an implementation of some numerical method we speak of an actual run of the program as a **numerical simulation** of the flow.

Most texts on CFD discuss many different numerical methods that have been used to solve the equations of fluid mechanics. Sometimes these methods are compared. More often they are not. Frequently, direct experimentation with implementations of different methods is the only real test of what works and what does not for a given type of flow. One can classify numerical codes broadly into general purpose and special purpose implementations. As the names indicate, implementations in the former category attempt to be applicable to a very wide variety of flow situations. Implementations in the latter category treat a more limited class of flows, hopefully and typically in greater detail than a general purpose implementation.

Many general purpose implementations are available commercially as codes. They are usually designed to be used as black boxes, i.e., the user is not supposed to change major elements of the source code (which may not be accessible). Such programs can have immense importance in engineering design situations, and many of them are used routinely in that way. However, for our present purposes they are less useful, since we want to start at a more primitive stage and trace the development from theory (PDEs) to executable code. Many human-years have been invested in commercial packages. Here we shall focus on smaller problems where the entire process of producing a numerical simulation is achievable in a limited time, and where one person can retain an overview of what has gone on in developing the program from start to finish.

Awareness of and experience with commercially available packages is certainly a marketable skill. However, neither an exhaustive discussion of techniques nor

the detailing of commercially available software will be our main goal in this book. Some general description of what is available and what has been and can be done is inevitable, but there are many other issues that we must explore, which involve the physics of fluid motions and illustrate the basic philosophy of computation. These include such topics as the computability of random processes (in our case a chaotic or turbulent flow), the adaptability of numerical methods to the flow physics (e.g., how to treat shocks or sharp interfaces, or how to limit diffusion), the manifestations of numerical errors as physical effects (e.g., as diffusion or surface tension), the retention in a discretization of integrals of the motion of a PDE, and so on. The general purpose codes invariably come with many switches and options. Good understanding of the anticipated general physics of the flow and the property of the numerical methods and algorithms employed is important to turn on the right switches in the general purpose code. Such understanding of the interplay between flow physics and numerical methods is the focus of this book. Issues of this kind will occupy us extensively.

The construction of numerical methods is an integral part of the field of CFD, and the attendant questions of stability, efficiency, robustness, etc., are all very important and interesting. Some of these come up in the context of very special problems, where the ability of an algorithm and attendant computer program to elucidate complicated analytical behavior of the flow solutions is in focus. General purpose methods typically have little to offer in addressing such questions. (A general purpose package will deal with the issues we raise in some way, but if you do not like the solution strategy adopted, you generally have little recourse, and you typically cannot experiment with methodological changes.) Our point of view is that although some of these issues might on occasion be dismissed as academic, they do, in general, provide a valuable avenue for looking behind the scenes and ultimately for assessing the quality of a given program. They constitute, in fact, the science of CFD.

1.4 Models and Methods

Issues of principle are often best elucidated by model problems where the mathematical mechanism under discussion is clearly displayed. It is probably not unfair to say that some of the most profound discoveries in CFD have come from the study of models using (quite obviously) special purpose numerical methods. In fact, most of these discoveries have arisen from programs that just barely qualify as implementations of numerical methods.

For example, calculations on a model called the *Lorenz equations* first showed evidence of the structure we now call a *strange attractor* (Figure 1.1). These computations changed our thinking about the transition from laminar to turbulent flow in profound ways. However, a program to solve the Lorenz equations and display graphs of the attractor certainly does not qualify as general purpose software for turbulent thermal convection – Lorenz equations are the simplest

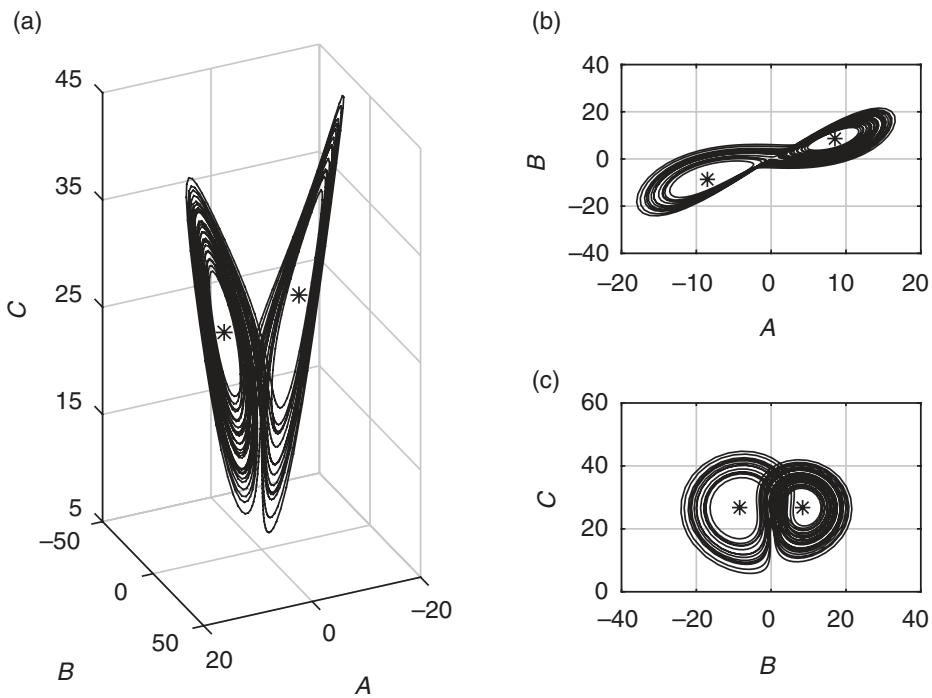

Figure 1.1 The initial computer generated evidence of a strange attractor (Lorenz, 1963). The equations that have been integrated numerically to produce these illustrations are $\frac{dA}{dt} = \Pr(B - A)$; $\frac{dB}{dt} = -AC + rA - B$; $\frac{dC}{dt} = AB - bC$. These equations result from expanding the equations governing thermal convection in a Fourier series and retaining the three lowest modes. In this expansion A is the amplitude of the stream-function mode, while B and C are modal amplitudes of the temperature field. The parameter $\Pr$ is the Prandtl number, which was chosen to be 10. The parameter b depends on the most unstable wavelength. To make the above Lorenz equations agree with the full linear stability calculation for the onset of thermal convection one needs to set $b = 8/3$. The final parameter, r, is the ratio of Rayleigh number to the critical Rayleigh number for the conduction-to-convection transition. Using the model equations the steady convection state that ensues for $r > 1$ becomes unstable for $r = 470/19 = 24.74$. The calculations were performed for $r = 28$. Panel (a) shows the trajectory over 30 time units. Panels (b) and (c) show the trajectory projected onto the AB- and BC-planes. The stars correspond to fixed points or steady convection.

model of thermal convection. Or think of the discovery of *solitons* by the numerical integration of certain PDEs (Figure 1.2), features that are not in fact all that easy to realize exactly in real flows.[1]

A system of equations abstracted from the general laws of fluid motion will be referred to as a **model**. On the other hand, the tradition in CFD has been to develop **methods**, and, indeed, to vary the resolution parameters to gain ever

[1] Closely related solitary waves are quite abundant, but solitons proper require a very special balance of steepening and dispersion in the governing equations.