Chapter 1
Introduction

1.1 Background

The field of computational fluid dynamics (CFD) is the subset of computational science concerned with the solution of the equations governing fluid flow. Although its birth date cannot be pinpointed precisely, it can be said to have begun in earnest in the 1960s. Not surprisingly, this coincides with the development of practical computers. However, the development of the pertinent theory began much earlier. A 1950 paper by Von Neumann and Richtmyer [1] contains a surprising number of the ideas of modern CFD. Furthermore, names such as Gauss, Richardson, and Courant, all of whom predate computers, crop up regularly in the CFD literature. Nevertheless, the development and application of CFD has paralleled that of computers. It is interesting to note that the concept of CFD was envisioned as soon as computers became a reality. For example, in 1946 Alan Turing remarked of the computer he was developing that it " . . . would be well adapted to deal with heat transfer problems, at any rate in solids or in fluids without turbulent motion" [2].

In addition to the emergence of viable computers, a second impetus for CFD comes from the inherent difficulty in obtaining general analytical solutions to the equations governing the flow of a fluid, the nonlinear partial differential equations known as the Navier-Stokes equations. At the core of CFD are algorithms for the numerical solution of these equations. Hence the theory associated with CFD algorithms is closely related to the more general theory of numerical methods for the solution of partial differential equations, appropriately specialized to the Navier-Stokes equations. Moreover, CFD in its broadest sense incorporates many other disciplines, from computational geometry to turbulence modeling.

The scientist or engineer often has a need for a quantitative knowledge of the flow of a fluid, such as the velocity, pressure, density, or temperature of the fluid at various locations in the flow domain, under a specific set of conditions. For the scientist, the purpose may be to gain an understanding of a particular phenomenon, such as turbulence or combustion. The engineer typically uses such information in the design process. In general, there are three means by which a quantitative understanding of

T. H. Pulliam and D. W. Zingg, *Fundamental Algorithms in Computational Fluid Dynamics*, Scientific Computation, DOI: 10.1007/978-3-319-05053-9_1,

the flow field of interest can be found: theory, experiment, and computation. Given that analytical solutions are rarely available, one is often left with two alternatives, experiment and CFD, each of which has its strengths and weaknesses. The primary advantages of CFD are typically cost and speed. The primary advantage of experiments is that they are in principle a true representation of reality. These generalizations are an oversimplification, since many CFD solutions can be time-consuming and require an expensive computer, and some experiments contain significant artifacts. Nevertheless, these characteristics of computation and experiment underly the decision process in choosing one or the other. In the aircraft industry, for example, the relatively low cost of CFD has led to a substantial reduction in the wind-tunnel testing performed when designing a new aircraft. However, final verification of the performance of an aircraft normally involves both wind-tunnel and flight testing.

Computing the solution to a specific flow problem using CFD involves a number of tasks, and for each task there exist numerous different approaches and methodologies. Despite this diversity, several common elements can be identified. Let us consider the following four basic steps involved in developing a useful numerical solution of a flow problem:

1. Problem and geometry specification
2. Mesh generation
3. Numerical solution of governing partial differential equations
4. Post-processing, assessment, and interpretation of results.

In order to discuss each of these tasks in more detail, we will consider a hypothetical problem, that of computing the forces generated on a specified aircraft wing in flight. In practice, one may be interested in knowing these forces for a wide range of flight conditions, but for our purpose here we will restrict our interest to one set of operating conditions.

In order to specify the problem, the operating conditions must be precisely defined. These include the speed of the aircraft, the orientation of the wing, and the state of the fluid through which the wing is flying, i.e. its pressure, density, and temperature. This information permits the calculation of key non-dimensional parameters such as the Reynolds number, Mach number, and Knudsen number. In addition, before embarking on a CFD adventure, one should have some qualitative idea of the answers to the following questions: How soon is the solution needed? What level of accuracy is needed in the forces? In other words, what level of error can be tolerated?

At this stage, based on the information described in the previous paragraph, several decisions must be made that will determine the success or failure of the venture. The following are examples of questions that must be asked: Is this a continuum flow? Will the flow be laminar or turbulent? If the latter, what is known about the onset of turbulence, the location of the transition from laminar to turbulent flow? Can compressibility effects be neglected? The answers to questions like these are needed to address the following critical question:
What governing equations will suffice to describe the expected flow phenomena to the desired level of accuracy?

For laminar flows, the correct answer is often the Navier-Stokes equations, as long as the continuum hypothesis holds, which depends on the Knudsen number. Nevertheless there are further questions to be answered: What is the equation of state of the fluid? Is the fluid Newtonian? If not, how are the viscous stresses defined? How does the viscosity vary with temperature? Are the expected flow phenomena time dependent?

When the flow is turbulent, the situation is much more complicated. The Navier-Stokes equations remain the appropriate governing equations. However, the physical time and space scales associated with high Reynolds number turbulent flows, especially wall-bounded flows, are usually much smaller than the scales associated with the geometry and flow speed. As a consequence, the numerical solution of such turbulent flows is extremely demanding computationally. Therefore, a hierarchy of approaches has been developed for tackling turbulent flow problems, ranging from a complete resolution of all relevant scales (known as direct numerical simulation or DNS [3]) to Reynolds averaging (or Favre averaging) in which the equations are time averaged and the resulting so-called Reynolds stresses are modeled. The time-averaged equations are known as the *Reynolds-averaged Navier-Stokes* (or RANS) equations. The models used for the Reynolds stresses are known as *turbulence models* and can be a significant source of error. In between the DNS and RANS approaches lie intermediate, hybrid approaches such as large-eddy simulation (LES) [4] or detached-eddy simulation (DES) [5]. These are intermediate in terms of both accuracy and computing cost.

The purpose of the above discussion is to demonstrate that a deep understanding of fluid dynamics is needed in order to properly formulate a problem for numerical solution. Next, we should discuss geometry specification and mesh generation, but before we can do that we need to define what we mean by a mesh, and we need at least a qualitative understanding of the errors that occur in CFD computations.

The methods we will describe in this book rely on a mesh, or grid–we will use the two terms interchangeably. A mesh is a collection of points that span the flow domain and are connected in some manner. There are two perspectives from which one may view the concept of a mesh. From a finite-difference perspective, the mesh supplies the points at which the solution is approximated, and the connectivity identifies the neighboring points to be used in constructing the finite-difference approximations. From a finite-volume perspective, the purpose of the mesh is to divide the flow domain into a large number of contiguous subdomains, or cells. Therefore, in two dimensions, the lines connecting the grid nodes are the edges of polygonal cells. In three dimensions, they are the edges of polyhedral cells.

The errors that occur in a computation of a fluid flow can be classified as numerical errors or physical model errors. Unless a suitable mesh is chosen for a specific application, numerical errors can be very large. They are typically reduced by adding additional mesh points to the flow domain. We call this *mesh refinement* and we describe the mesh with the added points as having an increased *density*. In principle, mesh dependent numerical error can be reduced to an arbitrarily small level by refining the mesh. With finite precision arithmetic, round-off error prevents the error from being reduced below some lower bound. In practice, this lower bound is rarely

approached due to the high computing costs associated with such a highly refined mesh. This has two important implications. First, it means that a certain degree of error is generally accepted. Thus it is important to have an understanding of this error and a means of measuring and controlling it. Second, it means that the properties of the mesh have a significant effect on the accuracy as well as the computational expense of the solution. Consequently a good understanding of both the flow and the algorithm is needed to generate an effective mesh. This motivates the idea of *solution-adaptive meshing*, in which the mesh is determined automatically as part of the solution process.

Numerical errors can be further subdivided into those that are dependent on the refinement of the computational mesh and those that are not. An example of the latter is the error arising as a result of performing an external flow computation on a finite computational domain, which implies enforcing boundary conditions at a finite distance from the body that theoretically should be applied an infinite distance from the body. We will not discuss such errors further at this stage except to remind the reader to be aware of them and to take the necessary steps to reduce them to appropriate levels. Another important type of error is round-off error, which results from finite precision arithmetic. For example, when calculating the difference between two numbers whose difference is many orders of magnitude smaller than either of the numbers, many significant digits are lost. Round-off error has implications for development and implementation of numerical algorithms. However, in practical CFD computations, it is rarely the predominant source of error.

Physical model errors are associated with the various models underlying the governing equations. The largest source of physical model error is often the turbulence model used together with the RANS equations, including the prediction of laminar-turbulent transition. However, it is important to be aware of other possible sources of physical model error, such as the perfect gas law and Sutherland's law, especially if they are used near the limits of their applicability. In comparisons with experiments, incorrect specification of boundary conditions related to the incoming flow can also be a source of error.

Physical model error is much more difficult to estimate and control than numerical error. Physical models must be validated through comparison with reliable experimental data. If the comparison is conducted properly, which means that the numerical errors are negligible compared to the physical model errors, the experimental errors are small, and the computation is an accurate representation of the experiment in terms of geometry, flow conditions, etc., then the physical model error for that particular flow is known. Through a number of such comparisons, a quantitative understanding of the physical model error for a range of flows is obtained. This can be used to estimate the physical model error for a given computation of a flow that lies within this range. If a model is applied outside the range of flows for which it has been validated, then the physical model error is not known and may be very large.

We are now ready to return to the tasks required to solve our hypothetical problem, computing the forces generated on an aircraft wing in flight. Having addressed all of the questions described above and thereby chosen a set of governing equations, let us say the compressible RANS equations, that will represent the physics of our problem

with sufficient accuracy, we are ready to generate a mesh. Therefore, we must have a suitable representation of the geometry. In the early days of CFD, the geometry was often represented simply by a surface mesh. Even though a more complete geometry representation may have existed, the mesh generator was provided with only the locations of a finite number of points lying on the surface of the geometry. This approach immediately becomes problematic when a more refined mesh is needed, especially in the context of solution-adaptive meshing. In order to add additional mesh points on the surface of the geometry, some sort of interpolation is needed. The interpolation used, which is dependent on the original surface mesh, then becomes the effective geometry representation. Different geometries can result from different initial meshes and different interpolation techniques. It is far preferable to separate the geometry representation from the mesh generation process and to have a complete and comprehensive geometry representation prior to generating the mesh.

The cost of a CFD computation (both processing time and memory) is dependent on the properties of the mesh and generally increases with the total number of nodes in the mesh. The accuracy of the computation is highly dependent on the properties of the mesh and generally improves as the total number of nodes is increased. Thus there is an inherent compromise between cost and accuracy, and it is important that the mesh points be judiciously distributed in a manner that leads to an efficient computation. For many flows, there are regions in the flow domain where the solution varies much more rapidly than in others. As a result, a uniform distribution of mesh nodes is rarely an efficient strategy. Determining the appropriate mesh density and an efficient placement of the mesh nodes requires an understanding of both the flow solution and the algorithm. This creates a dilemma, since the flow solution is not known prior to its computation. However, for many flows, qualitative features of the flow field can be identified a priori. For example, for our computation of the flow around a wing, since the Reynolds number is presumably large, we know that there will exist thin boundary layers near the surface of the wing. In these regions the flow velocity will change rapidly in the direction normal to the surface of the wing and much less rapidly in the directions parallel to the surface. In order to resolve such a flow field efficiently, the mesh should have a small spacing between mesh points normal to the wing surface and a larger spacing in the other two directions. Experience gained with other similar flows can also be used to guide the mesh generation process. Finally, automated solution-adaptive meshing can alleviate much of the difficulty associated with mesh generation.

The nature of the mesh has important implications for the solution algorithm and vice versa; each approach has advantages and disadvantages. A key distinction is between meshes that are fitted to the geometry and those that are not. Body-fitted meshes simplify the treatment of boundaries, while meshes that are not body-fitted, which are often Cartesian, can be easier to generate and can simplify the algorithm away from the boundaries. Body-fitted meshes can be classified as structured or unstructured; these terms are defined in Chap. 4. Different mesh types can also be combined to exploit their individual advantages. The choice of a specific meshing approach is a critical decision that depends on the geometric complexity and flow physics involved in the problem at hand.

Once a mesh has been generated, we are ready to solve the chosen governing equations. There exist many algorithms with various different properties, and the selection or development of an algorithm for a specific application depends on several factors. This we hope to clarify in this book.

After the solution has been computed, post-processing is required. For example, the forces on our wing must be calculated based on the computed flow solution. It is important that the post-processing calculations be performed in a manner that does not add significantly to the error. In addition, some form of error estimation should be performed, including both numerical and physical model error. If we are to make intelligent use of the calculated forces and moments, for example in the design of an aircraft, it is vital that we have a good understanding of the potential errors in those quantities. Moreover, it is worthwhile to investigate the computed flow solution, usually through flow visualization. This step may reveal that some sort of mundane error has occurred, such as an incorrect input, and, more importantly, it can serve as a check on the initial assumptions made in defining the problem and choosing the governing equations. Of course, a solution computed based on the assumption that the fluid is Newtonian will not provide evidence that it is actually non-Newtonian. However, if the solution has unexpected features, these may challenge some assumptions. If a flow field was computed on the assumption of steady flow, and a large region of flow separation is unexpectedly found, it may be worth recomputing the flow in a time dependent manner. Similarly, examination of the flow solution may reveal that the assumed location of laminar-turbulent transition is suspect or that the mesh is inadequately refined in some regions of the domain.

Finally, there are typically various uncertainties in a computation. Some parameters, such as the shape of the geometry, the angle to the flow, and some coefficients related to the fluid properties, are often known only to within some tolerance. It can be important to have a quantitative understanding of the sensitivity of important outputs of a computation, such as the forces generated by an aircraft wing, to variation in these uncertain input parameters. If the uncertainty in the inputs can be bounded or described in terms of a probability density function, this information can aid in finding a bound on or a probability density function for the outputs.

The goal of the CFD user is to generate a solution that is useful, trustworthy, and accurate; the goal of the CFD developer is to make this as likely as possible. The above discussion is intended to demonstrate that a great deal of knowledge and expertise is needed, of fluid dynamics as well as CFD, not only to develop algorithms and models, but also to apply them successfully. The purpose of this book is to provide the foundation needed to achieve the goals of both users and developers of CFD.

1.2 Overview and Roadmap

What are the foundations upon which the field of CFD is based? In other words, what are the basic topics that anyone intending to make use of CFD should understand? Our answer is as follows: finite-difference methods, finite-volume methods, as well as explicit and implicit time-marching methods. In addition, there are two further topics

that have become key ingredients of modern CFD: multigrid and high-resolution upwind schemes. These topics provide the foundations of CFD and are sufficiently mature that it is reasonable to cover them in a basic course in CFD.

Most algorithms used to solve the Euler and Navier-Stokes equations deal with the spatial and temporal aspects of the governing equations separately. Therefore, it is tempting to present them separately, and that is how the presentation of fundamentals proceeds in Chap. 2. However, we choose instead to present two complete algorithms, including both spatial and temporal discretization. One advantage of this approach is that it permits the reader to begin programming earlier. Having learned a complete algorithm in Chap. 4, the reader can immediately program it; this solidifies the understanding of the concepts and provides an opportunity to investigate the behavior of the algorithm. Moreover, we present the two specific algorithms in detail, rather than covering a broader range of different algorithms. This in-depth treatment of these two algorithms provides the reader with a strong basis for understanding other methodologies.

The fundamentals of CFD are covered in Chap. 2. This chapter summarizes much of the material in our previous book and can be omitted by those readers familiar with that book. It introduces the basic concepts of finite-difference methods, the semi-discrete approach, finite-volume methods, time-marching methods, stability analysis, and numerical dissipation in the context of two simple model equations, the linear convection equation and the diffusion equation. The approach is unified and general and provides the background needed to understand the subsequent chapters.

The Euler and Navier-Stokes equations are presented without derivation in Chap. 3 in a form suitable for numerical solution. They are given in both the partial differential equation form solved by finite-difference methods and the integral form solved by finite-volume methods. In addition this chapter introduces the quasi-one-dimensional Euler equations that form the basis of most of the programming assignments. This chapter's exercises require the development of the exact solutions to several one-dimensional problems to be used as a benchmark for the numerical solutions to be developed in the exercises of subsequent chapters.

Chapter 4 presents finite-difference methods and the implicit approximate-factorization algorithm. This classical algorithm forms the basis for many flow solvers, including the widely used NASA codes OVERFLOW [6] and CFL3D [7]. In addition, the generalized curvilinear coordinate transformation, artificial dissipation, and boundary conditions are covered. The exercises in this chapter provide an opportunity to write an implicit finite-difference solver and to apply it to some steady and unsteady problems. Some expected solutions and convergence histories are presented, so the reader can be certain that the algorithm has been properly understood and coded.

Chapter 5 presents a finite-volume method combined with explicit multi-stage time marching and multigrid. This classical algorithm was pioneered by Antony Jameson and his colleagues in various codes designated FLOxx [8–11] and is used in the NASA code TLNS3D [12]. The combination of explicit multi-stage time-marching and multigrid is particularly popular and is used in the NASA code CART3D [13], for example. This chapter's exercises involve coding multi-stage time

marching and multigrid. Again, the chapter includes plots of expected convergence behaviour to provide the reader with a reference for comparison.

Finally, Chap. 6 provides an introduction to high-resolution upwind schemes. These have been developed in order to improve the robustness and accuracy of numerical methods for the Euler and Navier-Stokes equations by maintaining some specific physical properties of the solution. For example, if a quantity, such as pressure, is physically required to be positive, then the introduction of a negative pressure as a result of numerical errors could cause significant problems. Calculation of the speed of sound for a perfect gas requires the square root of the pressure or temperature, presupposing that the pressure and temperature are nonnegative. High-resolution schemes are designed to prevent such unphysical occurrences, while maintaining accuracy, and are particularly relevant to flows with shock waves. As a result of their robustness, high-resolution upwind schemes have become quite prevalent in CFD for compressible flows. This chapter presents Godunov's method, which has been influential in the development of high-resolution upwind schemes. The popular approximate Riemann solver of Roe is also described. This leads into an introduction to the principles underlying high-resolution schemes and the presentation of some simple high-resolution upwind schemes with flux limiters. The exercise requires the programming of such a scheme to solve a shock-tube problem.

References

1. Von Neumann, J., and Richtmyer, R.D.: A method for the numerical calculation of hydrodynamic shocks. J. Appl. Phys. **21**, 232–237 (1950)
2. Hodges, A.: Alan Turing: The Enigma. Walker & Co., New York (2000)
3. Moin, P., Mahesh, K.: Direct numerical simulation: a tool in turbulence research. Annu. Rev. Fluid Mech. **30**(1), 539–578 (1998)
4. Sagaut, P.: Large Eddy Simulation for Turbulent Flows. Springer, Berlin (2005)
5. Spalart, P.R.: Strategies for turbulence modelling and simulations. Int. J. Heat Fluid Flow **21**(3), 252–263 (2000)
6. Jespersen, D.C., Pulliam, T.H., Buning, P.G.: Recent enhancements to OVERFLOW (Navier-Stokes code). AIAA Paper 97–0644 (1997)
7. Rumsey, C., Biedron, R., Thomas, J.: CFL3D: its history and some recent applications. NASA TM-112861 (1997)
8. Baker, T.J., Jameson, A., Schmidt, W.: A family of fast and robust Euler codes. Princeton University report MAE 1652 (1984)
9. Jameson, A.: Multigrid algorithms for compressible flow calculations. In: Proceedings of the 2nd European Conference on Multigrid Methods, Lecture Notes in Mathematics 1228. Springer, Heidelberg (1986)
10. Jameson, A., Baker, T.J.: Multigrid solution of the Euler equations for aircraft configurations. AIAA Paper 84–0093 (1984)
11. Swanson, R.C., Turkel, E.: Multistage schemes with multigrid for Euler and Navier-Stokes equations. NASA TP 3631 (1997)
12. Vatsa, V., Wedan, B.: Development of a multigrid code for 3-D Navier-Stokes equations and its application to a grid-refinement study. Comput. Fluids **18**(4), 391–403 (1990)
13. Aftosmis, M.J., Berger, M., Adomavicius, G.: A parallel multilevel method for adaptively refined Cartesian grids with embedded boundaries. AIAA Paper 2000–808 (2000)