2.2 What is CFD?

As we have seen in Chap. 1, flows and related phenomena can be described by partial differential (or integro-differential) equations, which cannot be solved analytically except in special cases. To obtain an approximate solution numerically, we have to use a *discretization method* which approximates the differential equations by a system of algebraic equations, which can then be solved on a computer. The approximations are applied to small domains in space and/or time so the numerical solution provides results at *discrete locations* in space and time. Much as the accuracy of experimental data depends on the quality of the tools used, the accuracy of numerical solutions is dependent on the quality of discretizations used.

Contained within the broad field of computational fluid dynamics are activities that cover the range from the automation of well-established engineering design methods to the use of detailed solutions of the Navier-Stokes equations as substitutes for experimental research into the nature of complex flows. At one end, one can purchase design packages for pipe systems that solve problems in a few seconds or minutes on personal computers or workstations. On the other, there are codes that may require hundreds of hours on the largest super-computers. The range is as large as the field of fluid mechanics itself, making it impossible to cover all of CFD in a single work. Also, the field is evolving so rapidly that we run the risk of becoming out of date in a short time.

We shall not deal with automated simple methods in this book. The basis for them is covered in elementary textbooks and undergraduate courses and the available program packages are relatively easy to understand and to use.

We shall be concerned with methods designed to solve the equations of fluid motion in two or three dimensions. These are the methods used in non-standard applications, by which we mean applications for which solutions (or, at least, good approximations) cannot be found in textbooks or handbooks. While these methods have been used in high-technology engineering (for example, aeronautics and astronautics) from the very beginning, they are being used more frequently in fields of engineering where the geometry is complicated or some important feature (such as the prediction of the concentration of a pollutant) cannot be dealt with by standard methods. CFD is finding its way into process, chemical, civil, and environmental engineering. Optimization in these areas can produce large savings in equipment and energy costs and in reduction of environmental pollution.

2.3 Possibilities and Limitations of Numerical Methods

We have already noted some problems associated with experimental work. Some of these problems are easily dealt with in CFD. For example, if we want to simulate the flow around a moving car in a wind tunnel, we need to fix

the car model and blow air at it – but the floor has to move at the air speed, which is difficult to do. It is not difficult to do in a numerical simulation. Other types of boundary conditions are easily prescribed in computations; for example, temperature or opaqueness of the fluid pose no problem. *If* we solve the unsteady three-dimensional Navier-Stokes equations accurately (as in direct simulation of turbulence), we obtain a complete data set from which any quantity of physical significance can be derived.

This sounds to good to be true. Indeed, these advantages of CFD are conditional on being able to solve the Navier-Stokes equations accurately, which is extremely difficult for most flows of engineering interest. We shall see in Chap. 9 why obtaining accurate numerical solutions of the Navier-Stokes equations for high Reynolds number flows is so difficult.

If we are unable to obtain *accurate* solutions for all flows, we have to determine what we can produce and learn to analyze and judge the results. First of all, we have to bear in mind that numerical results are always *approximate*. There are reasons for differences between computed results and 'reality' i.e. errors arise from each part of the process used to produce numerical solutions:

- The differential equations may contain approximations or idealizations, as discussed in Sect. 1.7;
- Approximations are made in the discretization process;
- In solving the discretized equations, iterative methods are used. Unless they are run for a very long time, the exact solution of the discretized equations is not produced.

When the governing equations are known accurately (e.g. the Navier-Stokes equations for incompressible Newtonian fluids), solutions of any desired accuracy can be achieved in principle. However, for many phenomena (e.g. turbulence, combustion, and multiphase flow) the exact equations are either not available or numerical solution is not feasible. This makes introduction of models a necessity. Even if we solve the equations exactly, the solution would not be a correct representation of reality. In order to *validate* the models, we have to rely on experimental data. Even when the exact treatment is possible, models are often needed to reduce the cost.

Discretization errors can be reduced by using more accurate interpolation or approximations or by applying the approximations to smaller regions but this usually increases the time and cost of obtaining the solution. Compromise is usually needed. We shall present some schemes in detail but shall also point out ways of creating more accurate approximations.

Compromises are also needed in solving the discretized equations. Direct solvers, which obtain accurate solutions, are seldom used, because they are too costly. Iterative methods are more common but the errors due to stopping the iteration process too soon need to be taken into account.

Errors and their estimation will be emphasized throughout this book. We shall present error estimates for many examples; the need to analyze and estimate numerical errors can not be overemphasized.

Visualization of numerical solutions using vector, contour or other kinds of plots or movies (videos) of unsteady flows is important for the interpretation of results. It is far and away the most effective means of interpreting the huge amount of data produced by a calculation. However, there is the danger that an erroneous solution may look good but may not correspond to the actual boundary conditions, fluid properties etc.! The authors have encountered incorrect numerically produced flow features that could be and have been interpreted as physical phenomena. Industrial users of commercial CFD codes should especially be careful, as the optimism of salesmen is legendary. Wonderful color pictures make a great impression but are of no value if they are not quantitatively correct. Results must be examined very critically before they are believed.

2.4 Components of a Numerical Solution Method

Since this book is meant not only for users of commercial codes but also for young researchers developing new codes, we shall present the important ingredients of a numerical solution method here. More details will be presented in the following chapters.

2.4.1 Mathematical Model

The starting point of any numerical method is the mathematical model, i.e. the set of partial differential or integro-differential equations and boundary conditions. Some sets of equations used for flow prediction were presented in Chap. 1. One chooses an appropriate model for the target application (incompressible, inviscid, turbulent; two- or three-dimensional, etc.). As already mentioned, this model may include simplifications of the exact conservation laws. A solution method is usually designed for a particular set of equations. Trying to produce a general purpose solution method, i.e. one which is applicable to all flows, is impractical, if not impossible and, as with most general purpose tools, they are usually not optimum for any one application.

2.4.2 Discretization Method

After selecting the mathematical model, one has to choose a suitable discretization method, i.e. a method of approximating the differential equations by a system of algebraic equations for the variables at some set of discrete locations in space and time. There are many approaches, but the most important of which are: finite difference (FD), finite volume (FV) and finite element (FE) methods. Important features of these three kinds of discretization methods are described at the end of this chapter. Other methods, like spectral schemes, boundary element methods, and cellular automata are used in CFD but their use is limited to special classes of problems.

Each type of method yields the same solution if the grid is very fine. However, some methods are more suitable to some classes of problems than others. The preference is often determined by the attitude of the developer. We shall discuss the pros and cons of the various methods later.

2.4.3 Coordinate and Basis Vector Systems

It was mentioned in Chap. 1 that the conservation equations can be written in many different forms, depending on the coordinate system and the basis vectors used. For example one can select Cartesian, cylindrical, spherical, curvilinear orthogonal or non-orthogonal coordinate systems, which may be fixed or moving. The choice depends on the target flow, and may influence the discretization method and grid type to be used.

One also has to select the basis in which vectors and tensors will be defined (fixed or variable, covariant or contravariant, etc.). Depending on this choice, the velocity vector and stress tensor can be expressed in terms of e.g. Cartesian, covariant or contravariant, physical or non-physical coordinate-oriented components. In this book we shall use Cartesian components exclusively for reasons explained in Chap. 8.

2.4.4 Numerical Grid

The discrete locations at which the variables are to be calculated are defined by the numerical grid which is essentially a discrete representation of the geometric domain on which the problem is to be solved. It divides the solution domain into a finite number of subdomains (elements, control volumes etc.). Some of the options available are the following:

- *Structured (regular) grid* — Regular or structured grids consist of families of grid lines with the property that members of a single family do not cross each other and cross each member of the other families only once. This allows the lines of a given set to be numbered consecutively. The position of any grid point (or control volume) within the domain is uniquely identified by a set of two (in 2D) or three (in 3D) indices, e.g. (i, j, k).
  This is the simplest grid structure, since it is logically equivalent to a Cartesian grid. Each point has four nearest neighbors in two dimensions and six in three dimensions; one of the indices of each neighbor of point P (indices i, j, k) differs by ± 1 from the corresponding index of P. An example of a structured 2D grid is shown in Fig. 2.1. This neighbor connectivity simplifies programming and the matrix of the algebraic equation system has a regular structure, which can be exploited in developing a solution technique. Indeed, there is a large number of efficient solvers applicable only to structured grids (see Chap. 5). The disadvantage of structured grids is that they can be used only for geometrically simple solution domains. Another disadvantage is that it may be difficult to control the distribution of the

grid points: concentration of points in one region for reasons of accuracy produces unnecessarily small spacing in other parts of the solution domain and a waste of resources. This problem is exaggerated in 3D problems. The long thin cells may also affect the convergence adversely.
Structured grids may be of H-, O-, or C-type; the names are derived from the shapes of the grid lines. Figure 2.1 shows an H-type grid which, when mapped onto a rectangle, has distinct east, west, north, and south boundaries. Figure 2.3 shows an O-type structured grid around a cylinder. In this type of grid, one set of grid lines is "endless"; if the grid lines are treated as coordinate lines and we follow the coordinate around the cylinder, it will continuously increase and, to avoid a problem, one must introduce an artificial "cut" at which the coordinate jumps from a finite value to zero. At the cut, the grid can be "unwrapped" but the neighboring points must be treated as interior grid points, in contrast to the treatment applied at the boundaries of an H-type grid. The outer grid in Fig. 2.3 is again of H-type. The block grid around the hydrofoil in Fig. bloknmgr is of C-type. In this type of grid, points on portions of one grid line coincide, requiring the introduction of a cut similar to the ones found in O-type grids. This type of grid is often used for bodies with sharp edges for which they are capable of good grid quality.

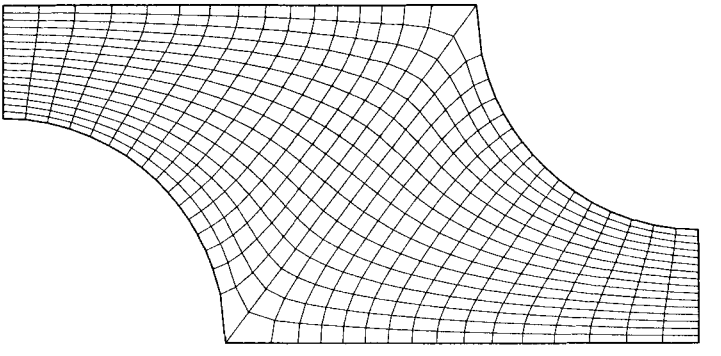

Fig. 2.1. Example of a 2D, structured, non-orthogonal grid, designed for calculation of flow in a symmetry segment of a staggered tube bank

- *Block-structured grid* — In a block structured grid, there is a two (or more) level subdivision of solution domain. On the coarse level, there are blocks which are relatively large segments of the domain; their structure may be irregular and they may or may not overlap. On the fine level (within each block) a structured grid is defined. Special treatment is necessary at block interfaces. Some methods of this kind are described in Chap. 8.
  In Fig. 2.2 a block-structured grid with matching at the interfaces is shown; it is designed for the calculation of 2D flow around a cylinder in a channel and contains three blocks.

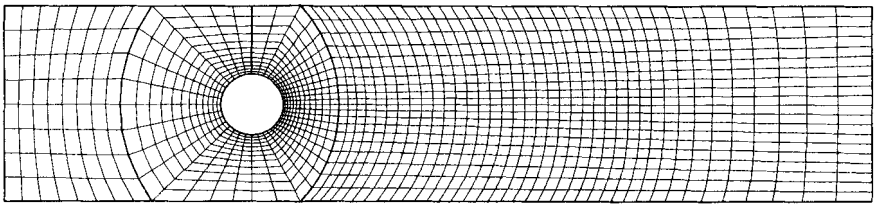

Fig. 2.2. Example of a 2D block-structured grid which matches at interfaces, used to calculate flow around a cylinder in a channel

In Fig. 2.3 a block-structured grid with non-matching interfaces is shown; it was used to calculate the flow around a submerged hydrofoil. It consists of five blocks of grids of different fineness. This kind of grid is more flexible than the previous ones, as it allows use of finer grids in regions, where greater resolution is required. The non-matching interface can be treated in a fully conservative manner, as will be discussed in Chap. 8. The programming is more difficult than for grid types described above. Solvers for structured grids can be applied block-wise, and complex flow domains can be treated with these grids. Local refinement is possible block-wise (i.e., the grid may be refined in some blocks).

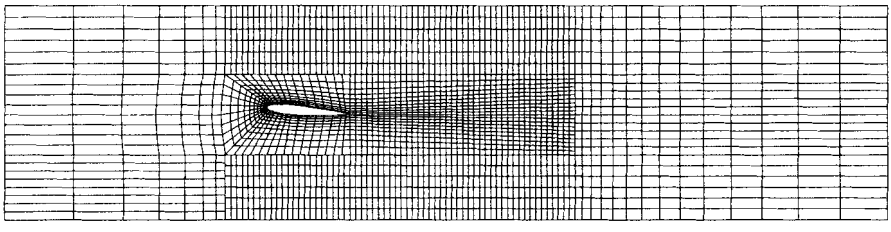

Fig. 2.3. Example of a 2D block-structured grid which does not match at interfaces, designed for calculation of flow around a hydrofoil under a water surface

Block-structured grids with overlapping blocks are sometimes called *composite* or *Chimera* grids. One such grid is shown in Fig. 2.4. In the overlap region, boundary conditions for one block are obtained by interpolating the solution from the other (overlapped) block. The disadvantage of these grids is that conservation is not easily enforced at block boundaries. The advantages of this approach are that complex domains are dealt with more easily and it can be used to follow moving bodies: one block is attached to the body and moves with it, while a stagnant grid covers the surroundings. This type of grid is not very often used, although it has strong supporters (Tu and Fuchs, 1992; Perng and Street, 1991; Hinatsu and Ferziger, 1991; Zang and Street, 1995; Hubbard and Chen, 1994, 1995).

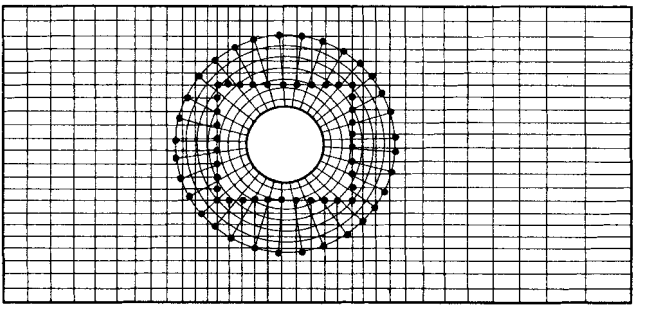

Fig. 2.4. A composite 2D grid, used to calculate flow around a cylinder in a channel

- *Unstructured grids* — For very complex geometries, the most flexible type of grid is one which can fit an arbitrary solution domain boundary. In principle, such grids could be used with any discretization scheme, but they are best adapted to the finite volume and finite element approaches. The elements or control volumes may have any shape; nor is there a restriction on the number of neighbor elements or nodes. In practice, grids made of triangles or quadrilaterals in 2D, and tetrahedra or hexahedra in 3D are most often used. Such grids can be generated automatically by existing algorithms. If desired, the grid can be made orthogonal, the aspect ratio is easily controlled, and the grid may be easily locally refined. The advantage of flexibility is offset by the disadvantage of the irregularity of the data structure. Node locations and neighbor connections need be specified explicitly. The matrix of the algebraic equation system no longer has regular, diagonal structure; the band width needs to be reduced by reordering of the points. The solvers for the algebraic equation systems are usually slower than those for regular grids.
  Unstructured grids are usually used with finite element methods and, increasingly, with finite volume methods. Computer codes for unstructured grids are more flexible. They need not be changed when the grid is locally refined, or when elements or control volumes of different shapes are used. However, grid generation and pre-processing are usually much more difficult. The finite volume method presented in this book is applicable to unstructured grids. An example of an unstructured grid is shown in Fig. 2.5.

Methods of grid generation will not be covered in detail in this book. Grid properties and some basic grid generation methods are discussed briefly in Chap. 8; there is a vast literature devoted to grid generation and interested reader is referred to books by Thompson et al. (1985) and Arcilla et al. (1991).

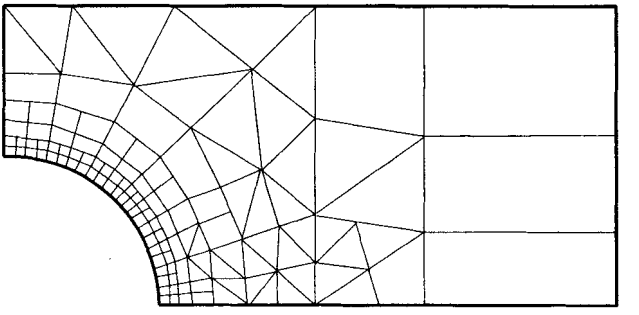

Fig. 2.5. Example of a 2D unstructured grid

2.4.5 Finite Approximations

Following the choice of grid type, one has to select the approximations to be used in the discretization process. In a finite difference method, approximations for the derivatives at the grid points have to be selected. In a finite volume method, one has to select the methods of approximating surface and volume integrals. In a finite element method, one has to choose the shape functions (elements) and weighting functions.

There are many possibilities to choose from; some of those most often used are presented in this book, some are simply mentioned and many more can be created. The choice influences the accuracy of the approximation. It also affects the difficulty of developing the solution method, coding it, debugging it, and the speed of the code. More accurate approximations involve more nodes and give fuller coefficient matrices. The increased memory requirement may require using coarser grids, partially offsetting the advantage of higher accuracy. A compromise between simplicity, ease of implementation, accuracy and computational efficiency has to be made. The second-order methods presented in this book were selected with this compromise in mind.

2.4.6 Solution Method

Discretization yields a large system of non-linear algebraic equations. The method of solution depends on the problem. For unsteady flows, methods based on those used for initial value problems for ordinary differential equations (marching in time) are used. At each time step an elliptic problem has to be solved. Steady flow problems are usually solved by pseudo-time marching or an equivalent iteration scheme. Since the equations are non-linear, an iteration scheme is used to solve them. These methods use successive linearization of the equations and the resulting linear systems are almost always solved by iterative techniques. The choice of solver depends on the grid type and the number of nodes involved in each algebraic equation. Some solvers will be presented in Chap. 5.

2.4.7 Convergence Criteria

Finally, one needs to set the convergence criteria for the iterative method. Usually, there are two levels of iterations: inner iterations, within which the linear equation are solved, and outer iterations, that deal with the non-linearity and coupling of the equations. Deciding when to stop the iterative process on each level is important, from both the accuracy and efficiency points of view. These issues are dealt with in Chaps. 5 and 11.

2.5 Properties of Numerical Solution Methods

The solution method should have certain properties. In most cases, it is not possible to analyze the complete solution method. One analyzes the components of the method; if the components do not possess the desired properties, neither will the complete method but the reverse is not necessarily true. The most important properties are summarized below.

2.5.1 Consistency

The discretization should become exact as the grid spacing tends to zero. The difference between the discretized equation and the exact one is called the *truncation error*. It is usually estimated by replacing all the nodal values in the discrete approximation by a Taylor series expansion about a single point. As a result one recovers the original differential equation plus a remainder, which represents the truncation error. For a method to be *consistent*, the truncation error must become zero when the mesh spacing $\Delta t \to 0$ and/or $\Delta x_i \to 0$. Truncation error is usually proportional to a power of the grid spacing Δx_i and/or the time step Δt. If the most important term is proportional to $(\Delta x)^n$ or $(\Delta t)^n$ we call the method an nth-order approximation; $n > 0$ is required for consistency. Ideally, all terms should be discretized with approximations of the same order of accuracy; however, some terms (e.g. convective terms in high Reynolds number flows or diffusive terms in low Reynolds number flows) may be dominant in a particular flow and it may be reasonable to treat them with more accuracy than the others.

Some discretization methods lead to truncation errors which are functions of the ratio of Δx_i to Δt or vice versa. In such a case the consistency requirement is only conditionally fulfilled: Δx_i and Δt must be reduced in a way that allows the appropriate ratio to go to zero. In the next two chapters we shall demonstrate consistency for several discretization schemes.

Even if the approximations are consistent, it does not necessarily mean that the solution of the discretized equation system will become the exact solution of the differential equation in the limit of small step size. For this to happen, the solution method has to be *stable*; this is defined below.

2.5.2 Stability

A numerical solution method is said to be stable if it does not magnify the errors that appear in the course of numerical solution process. For temporal problems, stability guarantees that the method produces a bounded solution whenever the solution of the exact equation is bounded. For iterative methods, a stable method is one that does not diverge. Stability can be difficult to investigate, especially when boundary conditions and non-linearities are present. For this reason, it is common to investigate the stability of a method for linear problems with constant coefficients without boundary conditions. Experience shows that the results obtained in this way can often be applied to more complex problems but there are notable exceptions.

The most widely used approach to studying stability of numerical schemes is the von Neumann's method. We shall describe it briefly for one scheme in Chap. 6. Most of the schemes to be described in this book have been analyzed for stability and we shall state the important result when describing each scheme. However, when solving complicated, non-linear and coupled equations with complicated boundary conditions, there are few stability results so we may have to rely on experience and intuition. Many solution schemes require that the time step be smaller than a certain limit or that under-relaxation be used. We shall discuss these issues and give guidelines for selecting time step size and values of under-relaxation parameters in Chaps. 6 and 7.

2.5.3 Convergence

A numerical method is said to be convergent if the solution of the discretized equations tends to the exact solution of the differential equation as the grid spacing tends to zero. For linear initial value problems, the *Lax equivalence theorem* (Richtmyer and Morton, 1967) states that "given a properly posed linear initial value problem and a finite difference approximation to it that satisfies the consistency condition, stability is the necessary and sufficient condition for convergence". Obviously, a consistent scheme is useless unless the solution method converges.

For non-linear problems which are strongly influenced by boundary conditions, the stability and convergence of a method are difficult to demonstrate. Therefore convergence is usually checked using numerical experiments, i.e. repeating the calculation on a series of successively refined grids. If the method is stable and if all approximations used in the discretization process are consistent, we usually find that the solution does converge to a *grid-independent solution.* For sufficiently small grid sizes, the rate of convergence is governed by the order of principal truncation error component. This allows us to estimate the error in the solution. We shall describe this in detail in Chaps. 3 and 5.

2.5.4 Conservation

Since the equations to be solved are conservation laws, the numerical scheme should also – on both a local and a global basis – respect these laws. This means that, at steady state and in the absence of sources, the amount of a conserved quantity leaving a closed volume is equal to the amount entering that volume. If the strong conservation form of equations and a finite volume method are used, this is guaranteed for each individual control volume and for the solution domain as a whole. Other discretization methods can be made conservative if care is taken in the choice of approximations. The treatment of sources or sink terms should be consistent so that the total source or sink in the domain is equal to the net flux of the conserved quantity through the boundaries.

This is an important property of the solution method, since it imposes a constraint on the solution error. If the conservation of mass, momentum and energy are insured, the error can only improperly distribute these quantities over the solution domain. Non-conservative schemes can produce artificial sources and sinks, changing the balance both locally and globally. However, non-conservative schemes can be consistent and stable and therefore lead to correct solutions in the limit of very fine grids. The errors due to non-conservation are in most cases appreciable only on relatively coarse grids. The problem is that it is difficult to know on which grid are these errors small enough. Conservative schemes are therefore preferred.

2.5.5 Boundedness

Numerical solutions should lie within proper bounds. Physically non-negative quantities (like density, kinetic energy of turbulence) must always be positive; other quantities, such as concentration, must lie between 0% and 100%. In the absence of sources, some equations (e.g. the heat equation for the temperature when no heat sources are present) require that the minimum and maximum values of the variable be found on the boundaries of the domain. These conditions should be inherited by the numerical approximation.

Boundedness is difficult to guarantee. We shall show later on that only some first order schemes guarantee this property. All higher-order schemes can produce unbounded solutions; fortunately, this usually happens only on grids that are too coarse, so a solution with undershoots and overshoots is usually an indication that the errors in the solution are large and the grid needs some refinement (at least locally). The problem is that schemes prone to producing unbounded solutions may have stability and convergence problems. These methods should be avoided, if possible.

2.5.6 Realizability

Models of phenomena which are too complex to treat directly (for example, turbulence, combustion, or multiphase flow) should be designed to guarantee

physically realistic solutions. This is not a numerical issue *per se* but models that are not realizable may result in unphysical solutions or cause numerical methods to diverge. We shall not deal with these issues in this book, but if one wants to implement a model in a CFD code, one has to be careful about this property.

2.5.7 Accuracy

Numerical solutions of fluid flow and heat transfer problems are only *approximate solutions*. In addition to the errors that might be introduced in the course of the development of the solution algorithm, in programming or setting up the boundary conditions, numerical solutions always include three kinds of systematic errors:

- *Modeling errors*, which are defined as the difference between the actual flow and the exact solution of the mathematical model;
- *Discretization errors*, defined as the difference between the exact solution of the conservation equations and the exact solution of the algebraic system of equations obtained by discretizing these equations, and
- *Iteration errors*, defined as the difference between the iterative and exact solutions of the algebraic equations systems.

Iteration errors are often called *convergence errors* (which was the case in the earlier editions of this book). However, the term *convergence* is used not only in conjunction with error reduction in iterative solution methods, but is also (quite appropriately) often associated with the convergence of numerical solutions towards a grid-independent solution, in which case it is closely linked to discretization error. To avoid confusion, we shall adhere to the above definition of errors and, when discussing issues of convergence, always indicate which type of convergence we are talking about.

It is important to be aware of the existence of these errors, and even more to try to distinguish one from another. Various errors may cancel each other, so that sometimes a solution obtained on a coarse grid may agree better with the experiment than a solution on a finer grid – which, by definition, should be more accurate.

Modeling errors depend on the assumptions made in deriving the transport equations for the variables. They may be considered negligible when laminar flows are investigated, since the Navier-Stokes equations represent a sufficiently accurate model of the flow. However, for turbulent flows, two-phase flows, combustion etc., the modeling errors may be very large – the exact solution of the model equations may be *qualitatively* wrong. Modeling errors are also introduced by simplifying the geometry of the solution domain, by simplifying boundary conditions etc. These errors are not known *a priori*; they can only be evaluated by comparing solutions in which the discretization and convergence errors are negligible with accurate experimental data or with data obtained by more accurate models (e.g. data from direct

simulation of turbulence, etc.). It is essential to control and estimate the convergence and discretization errors before the models of physical phenomena (like turbulence models) can be judged.

We mentioned above that discretization approximations introduce errors which decrease as the grid is refined, and that the order of the approximation is a measure of accuracy. However, on a given grid, methods of the same order may produce solution errors which differ by as much as an order of magnitude. This is because the order only tells us the *rate* at which the error decreases as the mesh spacing is reduced – it gives no information about the error on a single grid. We shall show how discretization errors can be estimated in the next chapter.

Errors due to iterative solution and round-off are easier to control; we shall see how this can be done in Chap. 5, where iterative solution methods are introduced.

There are many solution schemes and the developer of a CFD code may have a difficult time deciding which one to adopt. The ultimate goal is to obtain desired accuracy with least effort, or the maximum accuracy with the available resources. Each time we describe a particular scheme we shall point out its advantages or disadvantages with respect to these criteria.

2.6 Discretization Approaches

2.6.1 Finite Difference Method

This is the oldest method for numerical solution of PDE's, believed to have been introduced by Euler in the 18th century. It is also the easiest method to use for simple geometries.

The starting point is the conservation equation in differential form. The solution domain is covered by a grid. At each grid point, the differential equation is approximated by replacing the partial derivatives by approximations in terms of the nodal values of the functions. The result is one algebraic equation per grid node, in which the variable value at that and a certain number of neighbor nodes appear as unknowns.

In principle, the FD method can be applied to any grid type. However, in all applications of the FD method known to the authors, it has been applied to structured grids. The grid lines serve as local coordinate lines.

Taylor series expansion or polynomial fitting is used to obtain approximations to the first and second derivatives of the variables with respect to the coordinates. When necessary, these methods are also used to obtain variable values at locations other than grid nodes (interpolation). The most widely used methods of approximating derivatives by finite differences are described in Chap. 3.

On structured grids, the FD method is very simple and effective. It is especially easy to obtain higher-order schemes on regular grids; some will be

mentioned in Chap. 3. The disadvantage of FD methods is that the conservation is not enforced unless special care is taken. Also, the restriction to simple geometries is a significant disadvantage in complex flows.

2.6.2 Finite Volume Method

The FV method uses the integral form of the conservation equations as its starting point. The solution domain is subdivided into a finite number of contiguous control volumes (CVs), and the conservation equations are applied to each CV. At the centroid of each CV lies a computational node at which the variable values are to be calculated. Interpolation is used to express variable values at the CV surface in terms of the nodal (CV-center) values. Surface and volume integrals are approximated using suitable quadrature formulae. As a result, one obtains an algebraic equation for each CV, in which a number of neighbor nodal values appear.

The FV method can accommodate any type of grid, so it is suitable for complex geometries. The grid defines only the control volume boundaries and need not be related to a coordinate system. The method is conservative by construction, so long as surface integrals (which represent convective and diffusive fluxes) are the same for the CVs sharing the boundary.

The FV approach is perhaps the simplest to understand and to program. All terms that need be approximated have physical meaning which is why it is popular with engineers.

The disadvantage of FV methods compared to FD schemes is that methods of order higher than second are more difficult to develop in 3D. This is due to the fact that the FV approach requires three levels of approximation: interpolation, differentiation, and integration. We shall give a detailed description of the FV method in Chap. 4; it is the most used method in this book.

2.6.3 Finite Element Method

The FE method is similar to the FV method in many ways. The domain is broken into a set of discrete volumes or finite elements that are generally unstructured; in 2D, they are usually triangles or quadrilaterals, while in 3D tetrahedra or hexahedra are most often used. The distinguishing feature of FE methods is that the equations are multiplied by a *weight function* before they are integrated over the entire domain. In the simplest FE methods, the solution is approximated by a linear shape function within each element in a way that guarantees continuity of the solution across element boundaries. Such a function can be constructed from its values at the corners of the elements. The weight function is usually of the same form.

This approximation is then substituted into the weighted integral of the conservation law and the equations to be solved are derived by requiring the

derivative of the integral with respect to each nodal value to be zero; this corresponds to selecting the best solution within the set of allowed functions (the one with minimum residual). The result is a set of non-linear algebraic equations.

An important advantage of finite element methods is the ability to deal with arbitrary geometries; there is an extensive literature devoted to the construction of grids for finite element methods. The grids are easily refined; each element is simply subdivided. Finite element methods are relatively easy to analyze mathematically and can be shown to have optimality properties for certain types of equations. The principal drawback, which is shared by any method that uses unstructured grids, is that the matrices of the linearized equations are not as well structured as those for regular grids making it more difficult to find efficient solution methods. For more details on finite element methods and their application to the Navier-Stokes equations, see books by Oden (1972), Zinkiewicz (1977), Chung (1978), Baker (1983), Girault and Raviart (1986) or Fletcher (1991).

A hybrid method called *control-volume-based finite element method* (CV-FEM) should also be mentioned. In it, shape functions are used to describe the variation of the variables over an element. Control volumes are formed around each node by joining the centroids of the elements. The conservation equations in integral form are applied to these CVs in the same way as in the finite volume method. The fluxes through CV boundaries and the source terms are calculated element-wise. We shall give a short description of this approach in Chap. 8.