INTRODUCTION TO COMPUTATIONAL FLUID DYNAMICS

A brief introduction to computational fluid dynamics (CFD) is presented in this chapter. While any intelligent, computer-literate person can run a CFD code, the results he or she obtains may not be physically correct. In fact, if the grid is not properly generated, or if the boundary conditions or flow parameters are improperly applied, the results may even be completely erroneous. Therefore, the goal of this chapter is to present guidelines about how to generate a grid, how to specify boundary conditions, and how to determine if the computer output is meaningful. We stress the application of CFD to engineering problems, rather than details about grid generation techniques, discretization schemes, CFD algorithms, or numerical stability.

The examples presented here have been obtained with the commercial computational fluid dynamics code FLUENT. Other CFD codes would yield similar, but not identical results. Sample CFD solutions are shown for incompressible and compressible laminar and turbulent flows, flows with heat transfer, and flows with free surfaces. As always, one learns best by hands-on practice. For this reason, we provide several homework problems that utilize FLUENT FLOWLAB®, hereafter referred to as FlowLab, a student-friendly CFD code that is available with the purchase of this book.

CHAPTER 15

OBJECTIVES
When you finish reading this chapter, you should be able to:

- Understand the importance of a high-quality, good resolution mesh
- Apply appropriate boundary conditions to computational domains
- Understand how to apply CFD to basic engineering problems and how to determine whether the output is physically meaningful
- Realize that you need much further study and practice to use CFD successfully
INTRODUCTION AND FUNDAMENTALS

Motivation

There are two fundamental approaches to design and analysis of engineering systems that involve fluid flow: experimentation and calculation. The former typically involves construction of models that are tested in wind tunnels or other facilities (Chap. 7), while the latter involves solution of differential equations, either analytically (Chaps. 9 and 10) or computationally. In the present chapter, we provide a brief introduction to computational fluid dynamics (CFD), the field of study devoted to solution of the equations of fluid flow through use of a computer (or, more recently, several computers working in parallel). Modern engineers apply both experimental and CFD analyses, and the two complement each other. For example, engineers may obtain global properties, such as lift, drag, pressure drop, or power, experimentally, but use CFD to obtain details about the flow field, such as shear stresses, velocity and pressure profiles (Fig. 15–1), and flow streamlines. In addition, experimental data are often used to validate CFD solutions by matching the computationally and experimentally determined global quantities. CFD is then employed to shorten the design cycle through carefully controlled parametric studies, thereby reducing the required amount of experimental testing.

The current state of computational fluid dynamics is that CFD can handle laminar flows with ease, but turbulent flows of practical engineering interest are impossible to solve without invoking turbulence models. Unfortunately, no turbulence model is universal, and a turbulent CFD solution is only as good as the appropriateness of the turbulence model. In spite of this limitation, the standard turbulence models yield reasonable results for many practical engineering problems.

There are several aspects of CFD that are not covered in this chapter—grid generation techniques, numerical algorithms, finite difference and finite volume schemes, stability issues, turbulence modeling, etc. You need to study these topics in order to fully understand both the capabilities and limitations of computational fluid dynamics. In this chapter, we merely scratch the surface of this exciting field. Our goal is to present the fundamentals of CFD from a user’s point of view, providing guidelines about how to generate a grid, how to specify boundary conditions, and how to determine if the computer output is physically meaningful.

We begin this section by presenting the differential equations of fluid flow that are to be solved, and then we outline a solution procedure. Subsequent sections of this chapter are devoted to example CFD solutions for laminar flow, turbulent flow, flows with heat transfer, compressible flow, and open-channel flow.

Equations of Motion

For steady laminar flow of a viscous, incompressible, Newtonian fluid without free-surface effects, the equations of motion are the continuity equation

\[ \mathbf{\nabla} \cdot \mathbf{V} = 0 \quad (15-1) \]

and the Navier–Stokes equation

\[ (\mathbf{\nabla} \cdot \mathbf{V}) \mathbf{V} = -\frac{1}{\rho} \mathbf{\nabla} p + \nu \nabla^2 \mathbf{V} \quad (15-2) \]
Strictly speaking, Eq. 15–1 is a conservation equation, while Eq. 15–2 is a transport equation that represents transport of linear momentum throughout the computational domain. In Eqs. 15–1 and 15–2, \( \mathbf{V} \) is the velocity of the fluid, \( \rho \) is its density, and \( \nu \) is its kinematic viscosity (\( \nu = \mu/\rho \)). The lack of free-surface effects enables us to use the modified pressure \( P' \), thereby eliminating the gravity term from Eq. 15–2 (see Chap. 10). Note that Eq. 15–1 is a scalar equation, while Eq. 15–2 is a vector equation. Equations 15–1 and 15–2 apply only to incompressible flows in which we also assume that both \( \rho \) and \( \nu \) are constants. Thus, for three-dimensional flow in Cartesian coordinates, there are four coupled differential equations for four unknowns, \( u, v, w, \) and \( P' \) (Fig. 15–2). If the flow were compressible, Eqs. 15–1 and 15–2 would need to be modified appropriately, as discussed in Section 15–5. Liquid flows can almost always be treated as incompressible, and for many gas flows, the gas is at a low enough Mach number that it behaves as a nearly incompressible fluid.

**Solution Procedure**

To solve Eqs. 15–1 and 15–2 numerically, the following steps are performed. Note that the order of some of the steps (particularly steps 2 through 5) is interchangeable.

1. A computational domain is chosen, and a grid (also called a mesh) is generated; the domain is divided into many small elements called cells. For two-dimensional (2-D) domains, the cells are areas, while for three-dimensional (3-D) domains the cells are volumes (Fig. 15–3). You can think of each cell as a tiny control volume in which discretized versions of the conservation equations are solved. Note that we limit our discussion here to cell-centered finite volume CFD codes. The quality of a CFD solution is highly dependent on the quality of the grid. Therefore, you are advised to make sure that your grid is of high quality before proceeding to the next step (Fig. 15–4).

2. Boundary conditions are specified on each edge of the computational domain (2-D flows) or on each face of the domain (3-D flows).

3. The type of fluid (water, air, gasoline, etc.) is specified, along with fluid properties (temperature, density, viscosity, etc.). Many CFD codes

<table>
<thead>
<tr>
<th>Continuity:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x-momentum:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + w \frac{\partial u}{\partial z} = -\frac{1}{\rho} \frac{\partial P'}{\partial x} + \frac{\nu}{\rho} \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>y-momentum:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} + w \frac{\partial v}{\partial z} = -\frac{1}{\rho} \frac{\partial P'}{\partial y} + \frac{\nu}{\rho} \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2} \right) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>z-momentum:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u \frac{\partial w}{\partial x} + v \frac{\partial w}{\partial y} + w \frac{\partial w}{\partial z} = -\frac{1}{\rho} \frac{\partial P'}{\partial z} + \frac{\nu}{\rho} \left( \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} \right) )</td>
</tr>
</tbody>
</table>

**FIGURE 15–2**

The equations of motion to be solved by CFD for the case of steady, incompressible, laminar flow of a Newtonian fluid with constant properties and without free-surface effects. A Cartesian coordinate system is used. There are four equations and four unknowns: \( u, v, w, \) and \( P' \).

**FIGURE 15–3**

A computational domain is the region in space in which the equations of motion are solved by CFD. A cell is a small subset of the computational domain. Shown are (a) a two-dimensional domain and quadrilateral cell, and (b) a three-dimensional domain and hexahedral cell. The boundaries of a 2-D domain are called edges, while those of a 3-D domain are called faces.
have built-in property databases for common fluids, making this step relatively painless.

4. Numerical parameters and solution algorithms are selected. These are specific to each CFD code and are not discussed here. The default settings of most modern CFD codes are appropriate for the simple problems discussed in this chapter.

5. Starting values for all flow field variables are specified for each cell. These are initial conditions, which may or may not be correct, but are necessary as a starting point, so that the iteration process may proceed (step 6). We note that for proper unsteady-flow calculations, the initial conditions must be correct.

6. Beginning with the initial guesses, discretized forms of Eqs. 15–1 and 15–2 are solved iteratively, usually at the center of each cell. If one were to put all the terms of Eq. 15–2 on one side of the equation, the solution would be “exact” when the sum of these terms, defined as the residual, is zero for every cell in the domain. In a CFD solution, however, the sum is never identically zero, but (hopefully) decreases with progressive iterations. A residual can be thought of as a measure of how much the solution to a given transport equation deviates from exact, and one monitors the average residual associated with each transport equation to help determine when the solution has converged. Sometimes hundreds or even thousands of iterations are required to converge on a final solution, and the residuals may decrease by several orders of magnitude.

7. Once the solution has converged, flow field variables such as velocity and pressure are plotted and analyzed graphically. Users can also define and analyze additional custom functions that are formed by algebraic combinations of flow field variables. Most commercial CFD codes have built-in postprocessors, designed for quick graphical analysis of the flow field. There are also stand-alone postprocessor software packages available for this purpose. Since the graphics output is often displayed in vivid colors, CFD has earned the nickname colorful fluid dynamics.

8. Global properties of the flow field, such as pressure drop, and integral properties, such as forces (lift and drag) and moments acting on a body, are calculated from the converged solution (Fig. 15–5). With most CFD codes, this can also be done “on the fly” as the iterations proceed. In many cases, in fact, it is wise to monitor these quantities along with the residuals during the iteration process; when a solution has converged, the global and integral properties should settle down to constant values as well.

For unsteady flow, a physical time step is specified, appropriate initial conditions are assigned, and an iteration loop is carried out to solve the transport equations to simulate changes in the flow field over this small span of time. Since the changes between time steps are small, a relatively small number of iterations (on the order of tens) is usually required between each time step. Upon convergence of this “inner loop,” the code marches to the next time step. If a flow has a steady-state solution, that solution is often easier to find by marching in time—after enough time has past, the flow field variables settle down to their steady-state values. Most CFD codes take advantage of this fact by internally specifying a pseudo-time step (artificial time) and marching toward a steady-state solution. In such cases, the
pseudo-time step can even be different for different cells in the computational domain and can be tuned appropriately to decrease convergence time. Other “tricks” are often used to reduce computation time, such as multigridding, in which the flow field variables are updated first on a coarse grid so that gross features of the flow are quickly established. That solution is then interpolated to finer and finer grids, the final grid being the one specified by the user (Fig. 15–6). In some commercial CFD codes, several layers of multigridming may occur “behind the scenes” during the iteration process, without user input (or awareness). You can learn more about computational algorithms and other numerical techniques that improve convergence by reading books devoted to computational methods, such as Tannehill, Anderson, and Pletcher (1997).

**Additional Equations of Motion**

If energy conversion or heat transfer is important in the problem, another transport equation, the energy equation, must also be solved. If temperature differences lead to significant changes in density, an equation of state (such as the ideal-gas law) is used. If buoyancy is important, the effect of temperature on density is reflected in the gravity term (which must then be separated from the modified pressure term in Eq. 15–2).

For a given set of boundary conditions, a laminar flow CFD solution approaches an “exact” solution, limited only by the accuracy of the discretization scheme used for the equations of motion, the level of convergence, and the degree to which the grid is resolved. The same would be true of a turbulent flow simulation if the grid could be fine enough to resolve all the unsteady, three-dimensional turbulent eddies. Unfortunately, this kind of direct simulation of turbulent flow is usually not possible for practical engineering applications due to computer limitations. Instead, additional approximations are made in the form of turbulence models so that turbulent flow solutions are possible. The turbulence models generate additional transport equations that model the enhanced mixing and diffusion of turbulence; these additional transport equations must be solved along with those of mass and momentum. Turbulence modeling is discussed in more detail in Section 15–3.

Modern CFD codes include options for calculation of particle trajectories, species transport, heat transfer, and turbulence. The codes are easy to use, and solutions can be obtained without knowledge about the equations or their limitations. Herein lies the danger of CFD: When in the hands of someone without knowledge of fluid mechanics, erroneous results are likely to occur (Fig. 15–7). It is critical that users of CFD possess some fundamental knowledge of fluid mechanics so that they can discern whether a CFD solution makes physical sense or not.

**Grid Generation and Grid Independence**

The first step (and arguably the most important step) in a CFD solution is generation of a grid that defines the cells on which flow variables (velocity, pressure, etc.) are calculated throughout the computational domain. Modern commercial CFD codes come with their own grid generators, and third-party grid generation programs are also available. The grids used in this chapter are generated with FLUENT’s grid generation package, GAMBIT.
Many CFD codes can run with either structured or unstructured grids. A **structured grid** consists of planar cells with four edges (2-D) or volumetric cells with six faces (3-D). Although the cells can be distorted from rectangular, each cell is numbered according to indices \((i,j,k)\) that do not necessarily correspond to coordinates \(x, y,\) and \(z\). An illustration of a 2-D structured grid is shown in Fig. 15–8. To construct this grid, nine **nodes** are specified on the top and bottom edges; these nodes correspond to eight **intervals** along these edges. Similarly, five nodes are specified on the left and right edges, corresponding to four intervals along these edges. The intervals correspond to \(i = 1\) through 8 and \(j = 1\) through 4, and are numbered and marked in Fig. 15–8. An internal grid is then generated by connecting nodes one-for-one across the domain such that rows \((j = \text{constant})\) and columns \((i = \text{constant})\) are clearly defined, even though the cells themselves may be distorted (not necessarily rectangular). In a 2-D structured grid, each cell is uniquely specified by an index pair \((i,j)\). For example, the shaded cell in Fig. 15–8 is at \((i = 4, j = 3)\). You should be aware that some CFD codes number nodes rather than intervals.

An **unstructured grid** consists of cells of various shapes, but typically triangles or quadrilaterals (2-D) and tetrahedrons or hexahedrons (3-D) are used. Two unstructured grids for the same domain as that of Fig. 15–8 are generated, using the same interval distribution on the edges; these grids are shown in Fig. 15–9. Unlike the structured grid, one cannot uniquely identify cells in the unstructured grid by indices \(i\) and \(j\); instead, cells are numbered in some other fashion internally in the CFD code.

For complex geometries, an unstructured grid is usually much easier for the user of the grid generation code to create. However, there are some advantages to structured grids. For example, some (usually older) CFD codes are written specifically for structured grids; these codes converge more rapidly, and often more accurately, by utilizing the index feature of structured grids. For modern general-purpose CFD codes that can handle both structured and unstructured grids, however, this is no longer an issue. More importantly, fewer cells are usually generated with a structured grid than with an unstructured grid. In Fig. 15–8, for example, the structured grid has \(8 \times 4 = 32\) cells, while the unstructured triangular grid of Fig. 15–9a has 76 cells, and the unstructured quadrilateral grid has 38 cells, even though the identical node distribution is applied at the edges in all three cases. In bound-

---

**FIGURE 15–8**
Sample structured 2-D grid with nine nodes and eight intervals on the top and bottom edges, and five nodes and four intervals on the left and right edges. Indices \(i\) and \(j\) are shown. The shaded cell is at \((i = 4, j = 3)\).

**FIGURE 15–9**
Sample 2-D unstructured grids with nine nodes and eight intervals on the top and bottom edges, and five nodes and four intervals on the left and right edges. These grids use the same node distribution as that of Fig. 15–8: (a) unstructured triangular grid, and (b) unstructured quadrilateral grid. The shaded cell in (a) is moderately skewed.
ary layers, where flow variables change rapidly normal to the wall and highly resolved grids are required close to the wall, structured grids enable much finer resolution than do unstructured grids for the same number of cells. This can be seen by comparing the grids of Figs. 15–8 and 15–9 near the far right edge. The cells of the structured grid are thin and tightly packed near the right edge, while those of the unstructured grids are not.

We must emphasize that regardless of the type of grid you choose (structured or unstructured, quadrilateral or triangular, etc.), it is the quality of the grid that is most critical for reliable CFD solutions. In particular, you must always be careful that individual cells are not highly skewed, as this can lead to convergence difficulties and inaccuracies in the numerical solution. The shaded cell in Fig. 15–9a is an example of a cell with moderately high skewness, defined as the departure from symmetry. There are various kinds of skewness, for both two- and three-dimensional cells. Three-dimensional cell skewness is beyond the scope of the present textbook—the type of skewness most appropriate for two-dimensional cells is equiangle skewness, defined as

\[
Q_{EAS} = \max\left(\frac{\theta_{\text{max}} - \theta_{\text{equal}}}{180^\circ - \theta_{\text{equal}}}, \frac{\theta_{\text{equal}} - \theta_{\text{min}}}{\theta_{\text{equal}}}\right)
\]  

(15-3)

where \(\theta_{\text{min}}\) and \(\theta_{\text{max}}\) are the minimum and maximum angles (in degrees) between any two edges of the cell, and \(\theta_{\text{equal}}\) is the angle between any two edges of an ideal equilateral cell with the same number of edges. For triangular cells \(\theta_{\text{equal}} = 60^\circ\) and for quadrilateral cells \(\theta_{\text{equal}} = 90^\circ\). You can show by Eq. 15–3 that \(0 < Q_{EAS} < 1\) for any 2-D cell. By definition, an equilateral triangle has zero skewness. In the same way, a square or rectangle has zero skewness. A grossly distorted triangular or quadrilateral element may have unacceptably high skewness (Fig. 15–10). Some grid generation codes use numerical schemes to smooth the grid so as to minimize skewness.

Other factors affect the quality of the grid as well. For example, abrupt changes in cell size can lead to numerical or convergence difficulties in the CFD code. Also, cells with a very large aspect ratio can sometimes cause problems. While you can often minimize the cell count by using a structured grid instead of an unstructured grid, a structured grid is not always the best choice, depending on the shape of the computational domain. You must always be cognizant of grid quality. Keep in mind that a high-quality unstructured grid is better than a poor-quality structured grid. An example is shown in Fig. 15–11 for the case of a computational domain with a small acute angle at the upper-right corner. For this example we have adjusted the
node distribution so that the grid in any case contains between 60 and 70 cells for direct comparison. The structured grid (Fig. 15–11a) has $8 \times 8 = 64$ cells; but even after smoothing, the maximum equiangle skewness is 0.83—cells near the upper right corner are highly skewed. The unstructured triangular grid (Fig. 15–11b) has 70 cells, but the maximum skewness is reduced to 0.76. More importantly, the overall skewness is lower throughout the entire computational domain. The unstructured quad grid (Fig. 15–11c) has 67 cells. Although the overall skewness is better than that of the structured mesh, the maximum skewness is 0.87—higher than that of the structured mesh. The hybrid grid shown in Fig. 15–11d is discussed shortly.

Situations arise in which a structured grid is preferred (e.g., the CFD code requires structured grids, boundary layer zones need high resolution, or the simulation is pushing the limits of available computer memory). Generation of a structured grid is straightforward for geometries with straight edges. All we need to do is divide the computational domain into four-sided (2-D) or six-sided (3-D) blocks or zones. Inside each block, we generate a structured grid (Fig. 15–12a). Such an analysis is called multiblock analysis. For more complicated geometries with curved surfaces, we need to determine how the computational domain can be divided into individual blocks that may or may not have flat edges (2-D) or faces (3-D). A two-dimensional example involving circular arcs is shown in Fig. 15–12b. Most CFD codes require that the nodes match on the common edges and faces between blocks.

Many commercial CFD codes allow you to split the edges or faces of a block and assign different boundary conditions to each segment of the edge or face. In Fig. 15–12a for example, the left edge of block 2 is split about two-thirds of the way up to accommodate the junction with block 1. The lower segment of this edge is a wall, and the upper segment of this edge is an interior edge. (These and other boundary conditions are discussed shortly.) A similar situation occurs on the right edge of block 2 and on the top edge of block 3. Some CFD codes accept only elementary blocks, namely, blocks whose edges or faces cannot be split. For example, the four-block grid of Fig. 15–12a requires seven elementary blocks under this limitation (Fig. 15–13). The total number of cells is the same, which you can
verify. Finally, for CFD codes that allow blocks with split edges or faces, we can sometimes combine two or more blocks into one. For example, it is left as an exercise to show how the structured grid of Fig. 5–11b can be simplified to just three nonelementary blocks.

When developing the block topology with complicated geometries as in Fig. 15–12b, the goal is to create blocks in such a way that no cells in the grid are highly skewed. In addition, cell size should not change abruptly in any direction, and the blocking topology should lend itself to clustering cells near solid walls so that boundary layers can be resolved. With practice you can master the art of creating sophisticated multiblock structured grids. Multiblock grids are necessary for structured grids of complex geometry. They may also be used with unstructured grids, but are not necessary since the cells can accommodate complex geometries.

Finally, a **hybrid grid** is one that combines regions or blocks of structured and unstructured grids. For example, you can mate a structured grid block close to a wall with an unstructured grid block outside of the region of influence of the boundary layer. A hybrid grid is often used to enable high resolution near a wall without requiring high resolution away from the wall (Fig. 15–14). When generating any type of grid (structured, unstructured, or hybrid), you must always be careful that individual cells are not highly skewed. For example, none of the cells in Fig. 15–14 has any significant skewness. Another example of a hybrid grid is shown in Fig. 15–11d. Here we have split the computational domain into two blocks. The four-sided block on the left is meshed with a structured grid, while the three-sided block on the right is meshed with an unstructured triangular grid. The maximum skewness is 0.76, the same as that of the unstructured triangular grid of Fig. 15–11b, but the total number of cells is reduced from 70 to 62.

Computational domains with very small angles like the one shown in Fig. 15–11 are difficult to mesh at the sharp corner, regardless of the type of cells used. One way to avoid large values of skewness at a sharp corner is to simply chop off or round off the sharp corner. This can be done very close to the corner so that the geometric modification is imperceptible from an overall view and has little if any effect on the flow, yet greatly improves the performance of the CFD code by reducing the skewness. For example, the troublesome sharp corner of the computational domain of Fig. 15–11 is chopped off and replotted in Fig. 15–15. Through use of multiblocking and hybrid grids, the grid shown in Fig. 15–15 has 62 cells and a maximum skewness of only 0.53—a vast improvement over any of the grids in Fig. 15–11.

Generation of a good grid is often tedious and time consuming; engineers who use CFD on a regular basis will agree that grid generation usually takes more of their time than does the CFD solution itself (engineer’s time, not
CPU time). However, time spent generating a good grid is time well spent, since the CFD results will be more reliable and may converge more rapidly (Fig. 15–16). A high-quality grid is critical to an accurate CFD solution; a poorly resolved or low-quality grid may even lead to an incorrect solution. It is important, therefore, for users of CFD to test if their solution is grid independent. The standard method to test for grid independence is to increase the resolution (by a factor of 2 in all directions if feasible) and repeat the simulation. If the results do not change appreciably, the original grid is probably adequate. If, on the other hand, there are significant differences between the two solutions, the original grid is likely of inadequate resolution. In such a case, an even finer grid should be tried until the grid is adequately resolved. This method of testing for grid independence is time consuming, and unfortunately, not always feasible, especially for large engineering problems in which the solution pushes computer resources to their limits. In a 2-D simulation, if one doubles the number of intervals on each edge, the number of cells increases by a factor of $2^2 = 4$; the required computation time for the CFD solution also increases by approximately a factor of 4. For three-dimensional flows, doubling the number of intervals in each direction increases the cell count by a factor of $2^3 = 8$. You can see how grid independence studies can easily get beyond the range of a computer’s memory capacity and/or CPU availability. If you cannot double the number of intervals because of computer limitations, a good rule of thumb is that you should increase the number of intervals by at least 20 percent in all directions to test for grid independence.

On a final note about grid generation, the trend in CFD today is automated grid generation, coupled with automated grid refinement based on error estimates. Yet even in the face of these emerging trends, it is critical that you understand how the grid impacts the CFD solution.

**Boundary Conditions**

While the equations of motion, the computational domain, and even the grid may be the same for two CFD calculations, the type of flow that is modeled is determined by the imposed boundary conditions. Appropriate boundary conditions are required in order to obtain an accurate CFD solution (Fig. 15–17). There are several types of boundary conditions available; the most relevant ones are listed and briefly described in the following. The names are those used by FLUENT; other CFD codes may use somewhat different terminology, and the details of their boundary conditions may differ. In the descriptions given, the words face or plane are used, implying three-dimensional flow. For a two-dimensional flow, the word edge or line should be substituted for face or plane.

**Wall Boundary Conditions**

The simplest boundary condition is that of a wall. Since fluid cannot pass through a wall, the normal component of velocity is set to zero relative to the wall along a face on which the wall boundary condition is prescribed. In addition, because of the no-slip condition, we usually set the tangential component of velocity at a stationary wall to zero as well. In Fig. 15–17, for example, the upper and lower edges of this simple domain are specified as...
wall boundary conditions with no slip. If the energy equation is being solved, either wall temperature or wall heat flux must also be specified (but not both; see Section 15–4). If a turbulence model is being used, turbulence transport equations are solved, and wall roughness may need to be specified, since turbulent boundary layers are influenced greatly by the roughness of the wall. In addition, users must choose among various kinds of turbulence wall treatments (wall functions, etc.). These turbulence options are beyond the scope of the present text (see Wilcox, 1998); fortunately the default options of most modern CFD codes are sufficient for many applications involving turbulent flow.

Moving walls and walls with specified shear stresses can also be simulated in many CFD codes. There are situations where we desire to let the fluid slip along the wall (we call this an “inviscid wall”). For example, we can specify a zero-shear-stress wall boundary condition along the free surface of a swimming pool or hot tub when simulating such a flow (Fig. 15–18). Note that with this simplification, the fluid is allowed to “slip” along the surface, since the viscous shear stress caused by the air above it is negligibly small (Chap. 9). When making this approximation, however, surface waves and their associated pressure fluctuations cannot be taken into account.

**Inflow/Outflow Boundary Conditions**

There are several options at the boundaries through which fluid enters the computational domain (inflow) or leaves the domain (outflow). They are generally categorized as either velocity-specified conditions or pressure-specified conditions. At a velocity inlet, we specify the velocity of the incoming flow along the inlet face. If energy and/or turbulence equations are being solved, the temperature and/or turbulence properties of the incoming flow need to be specified as well.

At a pressure inlet, we specify the total pressure along the inlet face (for example, flow coming into the computational domain from a pressurized tank of known pressure or from the far field where the ambient pressure is known). At a pressure outlet, fluid flows out of the computational domain. We specify the static pressure along the outlet face; in many cases this is atmospheric pressure (zero gage pressure). For example, the pressure is atmospheric at the outlet of a subsonic exhaust pipe open to ambient air (Fig. 15–19). Flow properties, such as temperature, and turbulence properties are also specified at pressure inlets and pressure outlets. For the latter case, however, these properties are not used unless the solution demands reverse flow across the outlet. Reverse flow at a pressure outlet is usually an indication that the computational domain is not large enough. If reverse flow warnings persist as the CFD solution iterates, the computational domain should be extended.

Pressure is not specified at a velocity inlet, as this would lead to mathematical overspecification, since pressure and velocity are coupled in the equations of motion. Rather, pressure at a velocity inlet adjusts itself to match the rest of the flow field. In similar fashion, velocity is not specified at a pressure inlet or outlet, as this would also lead to mathematical overspecification. Rather, velocity at a pressure-specified boundary condition adjusts itself to match the rest of the flow field (Fig. 15–20).
Another option at an outlet of the computational domain is the outflow boundary condition. At an outflow boundary, no flow properties are specified; instead, flow properties such as velocity, turbulence quantities, and temperature are forced to have zero gradients normal to the outflow face (Fig. 15–21). For example, if a duct is sufficiently long so that the flow is fully developed at the outlet, the outflow boundary condition would be appropriate, since velocity does not change in the direction normal to the outlet face. Note that the flow direction is not constrained to be perpendicular to the outflow boundary, as also illustrated in Fig. 15–21. If the flow is still developing, but the pressure at the outlet is known, a pressure outlet boundary condition would be more appropriate than an outflow boundary condition. The outflow boundary condition is often preferred over the pressure outlet in rotating flows since the swirling motion leads to radial pressure gradients that are not easily handled by a pressure outlet.

A common situation in a simple CFD application is to specify one or more velocity inlets along portions of the boundary of the computational domain, and one or more pressure outlets or outflows at other portions of the boundary, with walls defining the geometry of the rest of the computational domain. For example, in our swimming pool (Fig. 15–18), we set the left-most face of the computational domain as a velocity inlet and the bottom-most face as a pressure outlet. The rest of the faces are walls, with the free surface modeled as a wall with zero shear stress.

Finally, for compressible flow simulations, the inlet and outlet boundary conditions are further complicated by introduction of Riemann invariants and characteristic variables related to incoming and outgoing waves, discussion of which is beyond the scope of the present text. Fortunately, many CFD codes have a pressure far field boundary condition for compressible flows. This boundary condition is used to specify the Mach number, pressure, and temperature at an inlet. The same boundary condition can be applied at an outlet; when flow exits the computational domain, flow variables at the outlet are extrapolated from the interior of the domain. Again you must ensure that there is no reverse flow at an outlet.

Miscellaneous Boundary Conditions

Some boundaries of a computational domain are neither walls nor inlets or outlets, but rather enforce some kind of symmetry or periodicity. For example, the periodic boundary condition is useful when the geometry involves repetition. Flow field variables along one face of a periodic boundary are
numerically linked to a second face of identical shape (and in most CFD codes, also identical face mesh). Thus, flow leaving (crossing) the first periodic boundary can be imagined as entering (crossing) the second periodic boundary with identical properties (velocity, pressure, temperature, etc.). Periodic boundary conditions always occur in pairs and are useful for flows with repetitive geometries, such as flow between the blades of a turbomachine or through an array of heat exchanger tubes (Fig. 15–22). The periodic boundary condition enables us to work with a computational domain that is much smaller than the full flow field, thereby conserving computer resources. In Fig. 15–22, you can imagine an infinite number of repeated domains (dashed lines) above and below the actual computational domain (the light blue shaded region). Periodic boundary conditions must be specified as either translational (periodicity applied to two parallel faces, as in Fig. 15–22) or rotational (periodicity applied to two radially oriented faces). The region of flow between two neighboring blades of a fan (a flow passage) is an example of a rotationally periodic domain (see Fig. 15–56).

The symmetry boundary condition forces flow field variables to be mirror-imaged across a symmetry plane. Mathematically, gradients of most flow field variables in the direction normal to the symmetry plane are set to zero across the plane of symmetry, although some variables are specified as even functions and some as odd functions across a symmetry boundary condition. For physical flows with one or more symmetry planes, this boundary condition enables us to model a portion of the physical flow domain, thereby conserving computer resources. The symmetry boundary differs from the periodic boundary in that no “partner” boundary is required for the symmetry case. In addition, fluid can flow parallel to a symmetry boundary, whereas flow can cross a periodic boundary. Consider, for example, flow across an array of heat exchanger tubes (Fig. 15–22). If we assume that no flow crosses the periodic boundaries of that computational domain, we can use symmetry boundary conditions instead. Alert readers will notice that we can even cut the size of the computational domain in half by wise choice of symmetry planes (Fig. 15–23).

For axisymmetric flows, the axis boundary condition is applied to a straight edge that represents the axis of symmetry (Fig. 15–24a). Fluid can flow parallel to the axis, but cannot flow through the axis. The axisymmetric option enables us to solve the flow in two dimensions, as sketched in Fig. 15–24b. The computational domain is simply a rectangle in the xy-plane; you can imagine rotating this plane about the x-axis to generate the axisymmetry. In the case of swirling axisymmetric flows, fluid can also flow tangentially in a circular path around the axis of symmetry. Swirling axisymmetric flows are sometimes called rotationally symmetric.

Internal Boundary Conditions

The final classification of boundary conditions is imposed on faces or edges that do not define a boundary of the computational domain, but rather exist inside the domain. When an interior boundary condition is specified on a face, flow crosses through the face without any user-forced changes, just as it would cross from one interior cell to another (Fig. 15–25). This boundary condition is necessary for situations in which the computational domain is divided into separate blocks or zones, and enables
communication between blocks. We have found this boundary condition to be useful for postprocessing as well, since a predefined face is present in the flow field, on whose surface one can plot velocity vectors, pressure contours, etc. In more sophisticated CFD applications in which there is a sliding or rotating mesh, the interface between the two blocks is called upon to smoothly transfer information from one block to another.

The fan boundary condition is specified on a plane across which a sudden pressure increase (or decrease) is to be assigned. This boundary condition is similar to an interior boundary condition except for the forced pressure rise. The CFD code does not solve the detailed, unsteady flow field through individual fan blades, but simply models the plane as an infinitesimally thin fan that changes the pressure across the plane. The fan boundary condition is useful, for example, as a simple model of a fan inside a duct (Fig. 15–25), a ceiling fan in a room, or the propeller or jet engine that provides thrust to an airplane. If the pressure rise across the fan is specified as zero, this boundary condition degenerates to an interior boundary condition.

Practice Makes Perfect

The best way to learn computational fluid dynamics is through examples and practice. You are encouraged to experiment with various grids, boundary conditions, numerical parameters, etc., in order to get a feel for CFD. Before tackling a complicated problem, it is best to solve simpler problems, especially ones for which analytical or empirical solutions are known (for comparison and verification). In the following sections, we solve several example problems of general engineering interest to illustrate many of the capabilities and limitations of CFD. We start with laminar flows, and then provide some introductory turbulent flow examples. Finally we provide examples of flows with heat transfer, compressible flows, and liquid flows with free surfaces. Color images of the results are available on the book’s website, including some animations.
15–2  LAMINAR CFD CALCULATIONS

Computational fluid dynamics does an excellent job at computing incompressible, steady or unsteady, laminar flow, provided that the grid is well resolved and the boundary conditions are properly specified. We show several simple examples of laminar flow solutions, paying particular attention to grid resolution and appropriate application of boundary conditions. In all examples in this section, the flows are incompressible and two-dimensional (or axisymmetric).

Pipe Flow Entrance Region at Re = 500

Consider flow of room-temperature water inside a smooth round pipe of length \( L = 40.0 \text{ cm} \) and diameter \( D = 1.00 \text{ cm} \). We assume that the water enters at a uniform speed equal to \( V = 0.05024 \text{ m/s} \). The kinematic viscosity of the water is \( \nu = 1.005 \times 10^{-6} \text{ m}^2/\text{s} \), producing a Reynolds number of \( Re = V D / \nu = 500 \). We assume incompressible, steady, laminar flow. We are interested in the entrance region in which the flow gradually becomes fully developed. Because of the axisymmetry, we set up a computational domain that is a two-dimensional slice from the axis to the wall, rather than a three-dimensional cylindrical volume (Fig. 15–26). We generate six structured grids for this computational domain: very coarse (40 intervals in the axial direction \( \times 8 \) intervals in the radial direction), coarse (80 \( \times 16 \)), medium (160 \( \times 32 \)), fine (320 \( \times 64 \)), very fine (640 \( \times 128 \)), and ultrafine (1280 \( \times 256 \)). (Note that the number of intervals is doubled in both directions for each successive grid; the number of computational cells increases by a factor of 4 for each grid.) In all cases the nodes are evenly distributed axially, but are concentrated near the wall radially, since we expect larger velocity gradients near the pipe wall. Close-up views of the first three of these grids are shown in Fig. 15–27.

FIGURE 15–26
Because of axisymmetry about the \( x \)-axis, flow through a round pipe can be solved computationally with a two-dimensional slice through the pipe from \( r = 0 \) to \( D/2 \). The computational domain is the light blue shaded region, and the drawing is not to scale. Boundary conditions are indicated.

FIGURE 15–27
Portions of the three coarsest structured grids generated for the laminar pipe flow example: (a) very coarse (40 \( \times 8 \)), (b) coarse (80 \( \times 16 \)), and (c) medium (160 \( \times 32 \)). The number of computational cells is 320, 1280, and 5120, respectively. In each view, the pipe wall is at the top and the pipe axis is at the bottom, as in Fig. 15–26.
We run the CFD program FLUENT in double precision for all six cases. (Double precision arithmetic is not always necessary for engineering calculations—we use it here to obtain the best possible precision in our comparisons.) Since the flow is laminar, incompressible, and axisymmetric, only three transport equations are solved—continuity, \( x \)-momentum, and \( y \)-momentum. Note that coordinate \( y \) is used in the CFD code instead of \( r \) as the distance from the axis of rotation (Fig. 15–24). The CFD code is run until convergence (all the residuals level off). Recall that a residual is a measure of how much the solution to a given transport equation deviates from exact; the lower the residual, the better the convergence. For the very coarse grid case, this occurs in about 500 iterations, and the residuals level off to less than \( 10^{-12} \) (relative to their initial values). The decay of the residuals is plotted in Fig. 15–28 for the very coarse case. Note that for more complicated flow problems with finer grids, you cannot always expect such low residuals; in some CFD solutions, the residuals level off at much higher values, like \( 10^{-3} \).

We define \( P_1 \) as the average pressure at an axial location one pipe diameter downstream of the inlet. Similarly we define \( P_{20} \) at 20 pipe diameters. The average axial pressure drop from 1 to 20 diameters is thus \( \Delta P = P_1 - P_{20} \), and is equal to 4.404 Pa (to four significant digits of precision) for the very coarse grid case. Centerline pressure and axial velocity are plotted in Fig. 15–29a as functions of downstream distance. The solution appears to be physically reasonable. We see the increase of centerline axial velocity to conserve mass as the boundary layer on the pipe wall grows downstream. We see a sharp drop in pressure near the pipe entrance where viscous shear stresses on the pipe wall are highest. The pressure drop approaches linear toward the end of the entrance region where the flow is nearly fully developed, as expected. Finally, we compare in Fig. 15–29b the axial velocity profile at the end of the pipe to the known analytical solution for fully developed laminar pipe flow (see Chap. 8). The agreement is excellent, especially considering that there are only eight intervals in the radial direction.

Is this CFD solution grid independent? To find out, we repeat the calculations using the coarse, medium, fine, very fine, and ultrafine grids. The
The convergence of the residuals is qualitatively similar to that of Fig. 15–28 for all cases, but CPU time increases significantly as grid resolution improves, and the levels of the final residuals are not as low as those of the coarse resolution case. The number of iterations required until convergence also increases with improved grid resolution. The pressure drop from $x/D = 1$ to 20 is listed in Table 15–1 for all six cases. $\Delta P$ is also plotted as a function of number of cells in Fig. 15–30. We see that even the very coarse grid does a reasonable job at predicting $\Delta P$. The difference in pressure drop from the very coarse grid to the ultrafine grid is less than 10 percent. Thus, the very coarse grid may be adequate for some engineering calculations. If greater precision is needed, however, we must use a finer grid. We see grid independence to three significant digits by the very fine case. The change in $\Delta P$ from the very fine grid to the ultrafine grid is less than 0.07 percent—a grid as finely resolved as the ultrafine grid is unnecessary in any practical engineering analysis.

The most significant differences between the six cases occur very close to the pipe entrance, where pressure gradients and velocity gradients are largest. In fact, there is a singularity at the inlet, where the axial velocity changes suddenly from $V$ to zero at the wall because of the no-slip condition. We plot in Fig. 15–31 contour plots of normalized axial velocity, $u/V$ near the pipe entrance. We see that although global properties of the flow field (like overall pressure drop) vary by only a few percent as the grid is refined, details of the flow field (like the velocity contours shown here) change considerably with grid resolution. You can see that as the grid is continually refined, the axial velocity contour shapes become smoother and more well defined. The greatest differences in the contour shapes occur near the pipe wall.

**Flow around a Circular Cylinder at Re = 150**

To illustrate that reliable CFD results require correct problem formulation, consider the seemingly simple problem of steady, incompressible, two-dimensional flow over a circular cylinder of diameter $D = 2.0$ cm (Fig. 15–32). The two-dimensional computational domain used for this simulation

<table>
<thead>
<tr>
<th>Case</th>
<th>Number of Cells</th>
<th>$\Delta P$, Pa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very coarse</td>
<td>320</td>
<td>4.404</td>
</tr>
<tr>
<td>Coarse</td>
<td>1,280</td>
<td>3.983</td>
</tr>
<tr>
<td>Medium</td>
<td>5,120</td>
<td>3.998</td>
</tr>
<tr>
<td>Fine</td>
<td>20,480</td>
<td>4.016</td>
</tr>
<tr>
<td>Very fine</td>
<td>81,920</td>
<td>4.033</td>
</tr>
<tr>
<td>Ultrafine</td>
<td>327,680</td>
<td>4.035</td>
</tr>
</tbody>
</table>

**FIGURE 15–30**
Calculated pressure drop from $x/D = 1$ to 20 in the entrance flow region of axisymmetric pipe flow as a function of number of cells.
is sketched in Fig. 15–33. Only the upper half of the flow field is solved, due to symmetry along the bottom edge of the computational domain; a symmetry boundary condition is specified along this edge to ensure that no flow crosses the plane of symmetry. With this boundary condition imposed, the required computational domain size is reduced by a factor of 2. A stationary, no-slip wall boundary condition is applied at the cylinder surface. The left half of the far field outer edge of the domain has a velocity inlet boundary condition, on which is specified the velocity components \( u/V \) and \( v/0 \). A pressure outlet boundary condition is specified along the right half of the outer edge of the domain. (The gage pressure there is set to zero, but since the velocity field in an incompressible CFD code depends only on pressure differences, not absolute value of pressure, the value of pressure specified for the pressure outlet boundary condition is irrelevant.)

Three two-dimensional structured grids are generated for comparison: coarse (30 radial intervals \( \times \) 60 intervals along the cylinder surface \( = 1800 \) cells), medium (60 \( \times \) 120 \( = 7200 \) cells), and fine (120 \( \times \) 240 \( = 28,800 \) cells), as seen in Fig. 15–34. Note that only a small portion of the computational domain is shown here; the full domain extends 15 cylinder diameters.
outward from the origin, and the cells get progressively larger further away from the cylinder.

We apply a free-stream flow of air at a temperature of 25°C, at standard atmospheric pressure, and at velocity \( V = 0.1096 \text{ m/s} \) from left to right around this circular cylinder. The Reynolds number of the flow, based on cylinder diameter (\( D = 2.0 \text{ cm} \)), is thus \( \text{Re} = \rho v D / \mu = 150 \). Experiments at this Reynolds number reveal that the boundary layer is laminar and separates almost 10° before the top of the cylinder, at \( \alpha \approx 82° \) from the front stagnation point. The wake also remains laminar. Experimentally measured values of drag coefficient at this Reynolds number show much discrepancy in the literature; the range is from \( C_D = 1.1 \) to 1.4, and the differences are most likely due to the quality of the free-stream and three-dimensional effects (oblique vortex shedding, etc.). (Recall that \( C_D = 2F_D / \rho V^2A \), where \( A \) is the frontal area of the cylinder, and \( A = D \times \text{span of the cylinder} \), taken as unit length in a two-dimensional CFD calculation.)

CFD solutions are obtained for each of the three grids shown in Fig. 15–34, assuming steady laminar flow. All three cases converge without problems, but the results do not necessarily agree with physical intuition or with experimental data. Streamlines are shown in Fig. 15–35 for the three grid resolutions. In all cases, the image is mirrored about the symmetry line so that even though only the top half of the flow field is solved, the full flow field is displayed.

For the coarse resolution case (Fig. 15–35a), the boundary layer separates at \( \alpha = 120° \), well past the top of the cylinder, and \( C_D = 1.00 \). The boundary layer is not well enough resolved to yield the proper boundary layer separation point, and the drag is somewhat smaller than it should be. Two large counter-rotating separation bubbles are seen in the wake; they stretch several cylinder diameters downstream. For the medium resolution case (Fig. 15–35b), the flow field is significantly different. The boundary layer separates a little further upstream at \( \alpha = 110° \), which is more in line with the experimental results, but \( C_D \) has decreased to about 0.982—further away from the experimental value. The separation bubbles in the cylinder’s wake have grown much longer than those of the coarse grid case. Does refining the grid even further improve the numerical results? Figure 15–35c shows streamlines for the fine resolution case. The results look qualitatively similar to those of the medium resolution case, with \( \alpha = 109° \), but the drag coefficient is even smaller (\( C_D = 0.977 \)), and the separation bubbles are even

---

**FIGURE 15–33**

Computational domain (light blue shaded region) used to simulate steady two-dimensional flow over a circular cylinder (not to scale). It is assumed that the flow is symmetric about the x-axis. Applied boundary conditions are shown for each edge in parentheses. We also define \( \alpha \), the angle measured along the cylinder surface from the front stagnation point.
FIGURE 15–34
Structured two-dimensional grids around the upper half of a circular cylinder: (a) coarse (30 × 60), (b) medium (60 × 120), and (c) fine (120 × 240). The bottom edge is a line of symmetry. Only a portion of each computational domain is shown—the domain extends well beyond the portion shown here.
longer. A fourth calculation (not shown) at even finer grid resolution shows the same trend—the separation bubbles stretch downstream and the drag coefficient decreases somewhat.

Shown in Fig. 15–36 is a contour plot of tangential velocity component ($u_t$) for the medium resolution case. We plot values of $u_t$ over a very small range around zero, so that we can clearly see where along the cylinder the flow changes direction. This is thus a clever way to locate the separation point along a cylinder wall. Note that this works only for a circular cylinder because of its unique geometry. A more general way to determine the separation point is to identify the point along the wall where the wall shear stress $\tau_w$ is zero; this technique works for bodies of any shape. From Fig. 15–36, we see that the boundary layer separates at an angle of $\alpha = 110^\circ$ from the front stagnation point, much further downstream than the experimentally obtained value of $82^\circ$. In fact, all our CFD results predict boundary layer separation on the rear side rather than the front side of the cylinder.

These CFD results are unphysical—such elongated separation bubbles could not remain stable in a real flow situation, the separation point is too far downstream, and the drag coefficient is too low compared to experimental data. Furthermore, repeated grid refinement does not lead to better results as we would hope; on the contrary, the results get worse with grid refinement. Why do these CFD simulations yield such poor agreement with experiment? The answer is twofold:

1. We have forced the CFD solution to be steady, when in fact flow over a circular cylinder at this Reynolds number is not steady. Experiments show that a periodic Kármán vortex street forms behind the cylinder (Tritton, 1977; see also Fig. 4–25 of this text).

2. All three cases in Fig. 15–35 are solved for the upper half-plane only, and symmetry is enforced about the x-axis. In reality, flow over a circular cylinder is highly nonsymmetric; vortices are shed alternately from the top and the bottom of the cylinder, forming the Kármán vortex street.

To correct both of these problems, we need to run an unsteady CFD simulation with a full grid (top and bottom)—without imposing the symmetry condition. We run the simulation as an unsteady two-dimensional laminar flow, using the computational domain sketched in Fig. 15–37. The top and bottom (far field) edges are specified as a periodic boundary condition pair so that nonsymmetric oscillations in the wake are not suppressed (flow can cross these boundaries as necessary). The far field boundaries are also very far away (75 to 200 cylinder diameters), so that their effect on the calculations is insignificant.

The mesh is very fine near the cylinder to resolve the boundary layer. The grid is also fine in the wake region to resolve the shed vortices as they travel downstream. For this particular simulation, we use a hybrid grid somewhat like that shown in Fig. 15–14. The fluid is air, the cylinder diameter is 1.0 m, and the free-stream air speed is set to 0.00219 m/s. These values produce a Reynolds number of 150 based on cylinder diameter. Note that the Reynolds number is the important parameter in this problem—the choices of D, V, and type of fluid are not critical, so long as they produce the desired Reynolds number (Fig. 15–38).
As we march in time, small nonuniformities in the flow field amplify, and the flow becomes unsteady and antisymmetric with respect to the $x$-axis. A Kármán vortex street forms naturally. After sufficient CPU time, the simulated flow settles into a periodic vortex shedding pattern, much like the real flow. A contour plot of vorticity at one instant of time is shown in Fig. 15–39, along with a photograph showing streaklines of the same flow obtained experimentally in a wind tunnel. It is clear from the CFD simulation that the Kármán vortices decay downstream, since the magnitude of vorticity in the vortices decreases with downstream distance. This decay is partly physical (viscous), and partly artificial (numerical dissipation). Nevertheless, physical experiments verify the decay of the Kármán vortices. The decay is not so obvious in the streakline photograph (Fig. 15–39); this is due to the time-integrating property of streaklines, as was pointed out in Chap. 4. A close-up view of vortices shedding from the cylinder at a particular instant in time is shown in Fig. 15–40, again with a comparison between CFD results and experimental results—this time from experiments in a water channel. A full-color animated version of Fig. 15–40 is provided on the book’s website so that you can watch the dynamic process of vortex shedding.

We compare the CFD results to experimental results in Table 15–2. The calculated time-averaged drag coefficient on the cylinder is 1.14. As mentioned previously, experimental values of $C_D$ at this Reynolds number vary from about 1.1 to 1.4, so the agreement is within the experimental scatter. Note that the present simulation is two-dimensional, inhibiting any kind of oblique vortex shedding or other three-dimensional nonuniformities. This may be why our calculated drag coefficient is on the lower end of the reported experimental range. The Strouhal number of the Kármán vortex street is defined as

$$\text{Strouhal number: } St = \frac{f_{\text{shedding}}D}{V}$$  \hspace{1cm} (15–4)
FIGURE 15–39
Laminar flow in the wake of a circular cylinder at Re = 150: (a) an instantaneous snapshot of vorticity contours produced by CFD, and (b) time-integrated streaklines produced by a smoke wire located at x/D = 5. The vorticity contours show that Kármán vortices decay rapidly in the wake, whereas the streaklines retain a “memory” of their history from upstream, making it appear that the vortices continue for a great distance downstream.
Photo from Cimbala et al., 1988.

FIGURE 15–40
Close-up view of vortices shedding from a circular cylinder: (a) instantaneous vorticity contour plot produced by CFD at Re = 150, and (b) dye streaklines produced by dye introduced at the cylinder surface at Re = 140. An animated version of this CFD picture is available on the book’s website.
Photo (b) reprinted by permission of Sadatoshi Taneda.
where \( f_{\text{shedding}} \) is the shedding frequency of the vortex street. From our CFD simulation, we calculate \( St = 0.16 \). The experimentally obtained value of Strouhal number at this Reynolds number is about 0.18 (Williamson, 1989), so again the agreement is reasonable, although the CFD results are a bit low compared to experiment. Perhaps a finer grid would help somewhat, but the major reason for the discrepancy is more likely due to unavoidable three-dimensional effects in the experiments, which are not present in these two-dimensional simulations. Overall this CFD simulation is a success, as it captures all the major physical phenomena in the flow field.

This exercise with “simple” laminar flow over a circular cylinder has demonstrated some of the capabilities of CFD, but has also revealed several aspects of CFD about which one must be cautious. Poor grid resolution can lead to incorrect solutions, particularly with respect to boundary layer separation, but continued refinement of the grid does not lead to more physically correct results if the boundary conditions are not set appropriately (Fig. 15–41). For example, forced numerical flow symmetry is not always wise, even for cases in which the physical geometry is entirely symmetric.

Symmetric geometry does not guarantee symmetric flow.

In addition, forced steady flow may yield incorrect results when the flow is inherently unstable and/or oscillatory. Likewise, forced two-dimensionality may yield incorrect results when the flow is inherently three-dimensional.

How then can we ensure that a laminar CFD calculation is correct? Only by systematic study of the effects of computational domain size, grid resolution, boundary conditions, flow regime (steady or unsteady, 2-D or 3-D, etc.), along with experimental validation. As with most other areas of engineering, experience is of paramount importance.

### 15–3 = TURBULENT CFD CALCULATIONS

CFD simulations of turbulent flow are much more difficult than those of laminar flow, even for cases in which the flow field is steady in the mean (statisticians refer to this condition as stationary). The reason is that the finer features of the turbulent flow field are always unsteady and three-dimensional—random, swirling, vortical structures called turbulent eddies of all orientations arise in a turbulent flow (Fig. 15–42). Some CFD calculations use a technique called direct numerical simulation (DNS), in which an attempt is made to resolve the unsteady motion of all the scales of the turbulent flow. However, the size difference and the time scale difference between the largest and smallest eddies can be several orders of magnitude (\( L >> \eta \) in Fig. 15–42). Furthermore, these differences increase with the Reynolds number (Tennekes and Lumley, 1972), making DNS calculations of turbulent flows even more difficult as the Reynolds number increases.

DNS solutions require extremely fine, fully three-dimensional grids, large computers, and an enormous amount of CPU time. With today’s computers, DNS results are not yet feasible for practical high Reynolds number turbulent flows of engineering interest such as flow over a full-scale airplane. This situation is not expected to change for several more decades, even if the fantastic rate of computer improvement continues at today’s pace.

---

**TABLE 15–2**

Comparison of CFD results and experimental results for unsteady laminar flow over a circular cylinder at \( Re = 150^* \)

<table>
<thead>
<tr>
<th></th>
<th>( C_D )</th>
<th>( St )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment</td>
<td>1.1 to 1.4</td>
<td>0.18</td>
</tr>
<tr>
<td>CFD</td>
<td>1.14</td>
<td>0.16</td>
</tr>
</tbody>
</table>

* The main cause of the disagreement is most likely due to three-dimensional effects rather than grid resolution or numerical issues.
Thus, we find it necessary to make some simplifying assumptions in order to simulate complex, high Reynolds number, turbulent flow fields. The next level below DNS is large eddy simulation (LES). With this technique, large unsteady features of the turbulent eddies are resolved, while small-scale dissipative turbulent eddies are modeled (Fig. 15–43). The basic assumption is that the smaller turbulent eddies are isotropic; i.e., it is assumed that the small eddies are independent of coordinate system orientation and always behave in a statistically similar and predictable way, regardless of the turbulent flow field. LES requires significantly less computer resources than does DNS, because we eliminate the need to resolve the smallest eddies in the flow field. In spite of this, the computer requirements for practical engineering analysis and design are nevertheless still formidable using today’s technology. Further discussion about DNS and LES is beyond the scope of the present text, but these are areas of much current research.

The next lower level of sophistication is to model all the unsteady turbulent eddies with some kind of turbulence model. No attempt is made to resolve the unsteady features of any of the turbulent eddies, not even the largest ones (Fig. 15–44). Instead, mathematical models are employed to take into account the enhanced mixing and diffusion caused by turbulent eddies. For simplicity, we consider only steady (that is, stationary), incompressible flow. When using a turbulence model, the steady Navier–Stokes equation (Eq. 15–2) is replaced by what is called the Reynolds-averaged Navier–Stokes (RANS) equation, shown here for steady (stationary), incompressible, turbulent flow,

\[
\overline{\nabla \cdot \overline{V} - \nabla \overline{V}} = \frac{1}{\rho} \frac{\partial p}{\partial t} + \nabla \cdot \overline{(\tau_{ij, \text{turbulent}})} \tag{15-5}
\]

Compared to Eq. 15–2, there is an additional term on the right side of Eq. 15–5 that accounts for the turbulent fluctuations. \(\tau_{ij, \text{turbulent}}\) is a tensor known as the specific Reynolds stress tensor, so named because it acts in a similar fashion as the viscous stress tensor \(\tau_{ij}\) (Chap. 9). In Cartesian coordinates, \(\tau_{ij, \text{turbulent}}\) is

\[
\tau_{ij, \text{turbulent}} = \begin{pmatrix} u'u' & u'u' & u'u' \\ u'u' & u'u' & u'u' \\ u'u' & u'u' & u'u' \end{pmatrix} \tag{15-6}
\]

where the overbar indicates the time average of the product of two fluctuating velocity components and primes denote fluctuating velocity components. Since the Reynolds stress is symmetric, six additional unknowns are introduced into the problem. These new unknowns are modeled in various ways by turbulence models. A detailed description of turbulence models is beyond the scope of this text; you are referred to Wilcox (1998) or Chen and Jaw (1998) for further discussion.

There are many turbulence models in use today, including algebraic, one-equation, two-equation, and Reynolds stress models. Three of the most popular turbulence models are the \(k-E\) model, the \(k-\omega\) model, and the \(q-\omega\) model. These so-called two-equation turbulence models add two more transport equations, which must be solved simultaneously with the equations of mass and linear momentum (and also energy if this equation is
When a turbulence model is used in a CFD calculation, all the turbulent eddies are modeled, and only Reynolds-averaged flow properties are calculated. Shown is the average velocity profile. There are no resolved turbulent eddies. A useful rule of thumb for turbulence properties at a pressure inlet or velocity inlet boundary condition is to specify a turbulence intensity of 10 percent and a turbulent length scale of one-half of some characteristic length scale in the problem ($\ell = D/2$).

We emphasize that turbulence models are approximations that rely heavily on empirical constants for mathematical closure of the equations. The models are calibrated with the aid of direct numerical simulation and experimental data obtained from simple flow fields like flat plate boundary layers, shear layers, and isotropic decaying turbulence downstream of screens. Unfortunately, no turbulence model is universal, meaning that although the model works well for flows similar to those used for calibration, it is not guaranteed to yield a physically correct solution when applied to general turbulent flow fields, especially those involving flow separation and reattachment and/or large-scale unsteadiness.

Turbulent flow CFD solutions are only as good as the appropriateness and validity of the turbulence model used in the calculations. We emphasize also that this statement remains true regardless of how fine we make the computational grid. When applying CFD to laminar flows, we can often improve the physical accuracy of the simulation by refining the grid. This is not always the case for turbulent flow CFD analyses using turbulence models. While a refined grid produces better numerical accuracy, the physical accuracy of the solution is always limited by the physical accuracy of the turbulence model itself.

With these cautions in mind, we now present some practical examples of CFD calculations of turbulent flow fields. In all the turbulent flow examples discussed in this chapter, we employ the $k$-$\varepsilon$ turbulence model with wall functions. This model is the default turbulence model in many commercial CFD codes such as FLUENT. In all cases we assume stationary flow; no attempt is made to model unsteady features of the flow, such as vortex shedding in the wake of a bluff body. It is assumed that the turbulence model accounts for all the inherent unsteadiness due to turbulent eddies in the flow field. Note that unsteady (nonstationary) turbulent flows are also solvable with turbulence models, through the use of time-marching schemes (unsteady RANS calculations), but only when the time scale of the unsteadiness is much longer than that of individual turbulent eddies. For example, suppose you are calculating the forces and moments on a blimp during a gust of wind (Fig. 15–46). At the inlet boundary, you would impose the time-varying wind velocity and turbulence levels, and an unsteady turbulent flow solution could then be calculated using turbulence models. The large-
scale, overall features of the flow (flow separation, forces and moments on the body, etc.) would be unsteady, but the fine-scale features of the turbulent boundary layer, for example, would be modeled by the quasi-steady turbulence model.

### Flow around a Circular Cylinder at Re = 10,000

As our first example of a turbulent flow CFD solution, we calculate flow over a circular cylinder at Re = 10,000. For illustration, we use the same two-dimensional computational domain that was used for the laminar cylinder flow calculations, as sketched in Fig. 15–33. As with the laminar flow calculation, only the upper half of the flow field is solved here, due to symmetry along the bottom edge of the computational domain. We use the same three grids used for the laminar flow case as well—coarse, medium, and fine resolution (Fig. 15–34). We point out, however, that grids designed for turbulent flow calculations (especially those employing turbulence models with wall functions) are generally not the same as those designed for laminar flow of the same geometry, particularly near walls.

We apply a free-stream flow of air at 25°C and at velocity $V = 7.304$ m/s from left to right around this circular cylinder. The Reynolds number of the flow, based on cylinder diameter ($D = 2.0$ cm), is approximately 10,000. Experiments at this Reynolds number reveal that the boundary layer is laminar and separates several degrees upstream of the top of the cylinder (at $\alpha \approx 82^\circ$). The wake, however, is turbulent; such a mixture of laminar and turbulent flow is particularly difficult for CFD codes. The measured drag coefficient at this Reynolds number is $C_D = 1.15$ (Tritton, 1977). CFD solutions are obtained for each of the three grids, assuming stationary (steady in the mean) turbulent flow. We employ the $k$-$\varepsilon$ turbulence model with wall functions. The inlet turbulence level is set to 10 percent with a length scale of 0.01 m (half of the cylinder diameter). All three cases converge nicely.

Streamlines are plotted in Fig. 15–47 for the three grid resolution cases. In each plot, the image is mirrored about the symmetry line so that even though only the top half of the flow field is solved, the full flow field is visualized.

For the coarse resolution case (Fig. 15–47a), the boundary layer separates well past the top of the cylinder, at $\alpha \approx 140^\circ$. Furthermore, the drag coefficient $C_D$ is only 0.647, almost a factor of 2 smaller than it should be. Let’s see if a finer mesh improves the agreement with experimental data. For the medium resolution case (Fig. 15–47b), the flow field is significantly different. The boundary layer separates nearer to the top of the cylinder, at $\alpha = 104^\circ$, and $C_D$ has increased to about 0.742—closer, but still significantly less than the experimental value. We also notice that the recirculating eddies in the cylinder’s wake have grown in length by nearly a factor of 2 compared to those of the coarse grid case. Figure 15–47c shows streamlines for the fine resolution case. The results look very similar to those of the medium resolution case, and the drag coefficient has increased only slightly ($C_D = 0.753$). The boundary layer separation point for this case is at $\alpha = 102^\circ$.

Further grid refinement (not shown) does not change the results significantly from those of the fine grid case. In other words, the fine grid appears to be sufficiently resolved, yet the results do not agree with experiment.
Why? There are several problems with our calculations: we are modeling a steady flow, even though the actual physical flow is unsteady; we are enforcing symmetry about the x-axis, even though the physical flow is unsymmetric (a Kármán vortex street can be observed in experiments at this Reynolds number); and we are using a turbulence model instead of resolving all the small eddies of the turbulent flow. Another significant source of error in our calculations is that the CFD code is run with turbulence turned on in order to reasonably model the wake region, which is turbulent; however, the boundary layer on the cylinder surface is actually still laminar. The predicted location of the separation point downstream of the top of the cylinder is more in line with turbulent boundary layer separation, which does not occur until much higher values of Reynolds number (after the “drag crisis” at Re greater than $2 \times 10^5$).

The bottom line is that CFD codes have a hard time in the transitional regime between laminar and turbulent flow, and when there is a mixture of laminar and turbulent flow in the same computational domain. In fact, most commercial CFD codes give the user a choice between laminar and turbulent—there is no “middle ground.” In the present calculations, we model a turbulent boundary layer, even though the physical boundary layer is laminar; it is not surprising, then, that the results of our calculations do not agree well with experiment. If we would have instead specified laminar flow over the entire computational domain, the CFD results would have been even worse (less physical).

Is there any way around this problem of poor physical accuracy for the case of mixed laminar and turbulent flow? Perhaps. In some CFD codes you can specify the flow to be laminar or turbulent in different regions of the flow. But even then, the transitional process from laminar to turbulent flow is somewhat abrupt, again not physically correct. Furthermore, you would need to know where the transition takes place in advance—this defeats the purpose of a stand-alone CFD calculation for fluid flow prediction. Advanced wall treatment models are being generated that may some day do a better job in the transitional region. In addition, some new turbulence models are being developed that are better tuned to low Reynolds number turbulence.

In summary, we cannot accurately model the mixed laminar/turbulent flow problem of flow over a circular cylinder at Re $\sim$ 10,000 using standard turbulence models and the steady Reynolds-averaged Navier–Stokes (RANS) equation. It appears that accurate results can be obtained only through use of time-accurate (unsteady RANS), LES, or DNS solutions that are orders of magnitude more computationally demanding.

**Flow around a Circular Cylinder at Re $= 10^7$**

As a final cylinder example, we use CFD to calculate flow over a circular cylinder at Re $= 10^7$—well beyond the drag crisis. The cylinder for this case is of 1.0 m diameter, and the fluid is water. The free-stream velocity is 10.05 m/s. At this value of Reynolds number the experimentally measured value of drag coefficient is around 0.7 (Tritton, 1977). The boundary layer is turbulent at the separation point, which occurs at around 120°. Thus we do not have the mixed laminar/turbulent boundary layer problem as in the
lower Reynolds number example—the boundary layer is turbulent everywhere except near the nose of the cylinder, and we should expect better results from the CFD predictions. We use a two-dimensional half-grid similar to that of the fine resolution case of the previous examples, but the mesh near the cylinder wall is adapted appropriately for this high Reynolds number. As previously, we use the $k-\varepsilon$ turbulence model with wall functions. The inlet turbulence level is set to 10 percent with a length scale of 0.5 m. Unfortunately, the drag coefficient is calculated to be 0.262—less than half of the experimental value at this Reynolds number. Streamlines are shown in Fig. 15-48. The boundary layer separates a bit too far downstream, at $\alpha = 129^\circ$. There are several possible reasons for the discrepancy. We are forcing the simulated flow to be steady and symmetric, whereas the actual flow is neither, due to vortex shedding. (Vortices are shed even at high Reynolds numbers.) In addition, the turbulence model and its near wall treatment (wall functions) may not be capturing the proper physics of the flow field. Again we must conclude that accurate results for flow over a circular cylinder can be obtained only through use of a full grid rather than a half grid, and with time-accurate (unsteady RANS), LES, or DNS solutions that are orders of magnitude more computationally demanding.

Design of the Stator for a Vane-Axial Flow Fan
The next turbulent flow CFD example involves design of the stator for a vane-axial flow fan that is to be used to drive a wind tunnel. The overall fan diameter is $D = 1.0$ m, and the design point of the fan is at an axial-flow speed of $V = 50$ m/s. The stator vanes span from radius $r = r_{\text{hub}} = 0.25$ m at the hub to $r = r_{\text{tip}} = 0.50$ m at the tip. The stator vanes are upstream of the rotor blades in this design (Fig. 15-49). A preliminary stator vane shape is chosen that has a trailing edge angle of $\beta_{\text{st}} = 63^\circ$ and a chord length of 20 cm. At any value of radius $r$, the actual amount of turning depends on the number of stator vanes—we expect that the fewer the number of vanes, the smaller the average angle at which the flow is turned by the stator vanes because of the greater spacing between vanes. It is our goal to determine the minimum number of stator vanes required so that the flow impinging on the leading edges of the rotor blades (located one chord length downstream of the stator vane trailing edges) is turned at an average angle of at least 45°. We also require there to be no significant flow separation from the stator vane surface.

As a first approximation, we model the stator vanes at any desired value of $r$ as a two-dimensional cascade of vanes (see Chap. 14). Each vane is separated by blade spacing $s$ at this radius, as defined in Fig. 15-50. We use CFD to predict the maximum allowable value of $s$, from which we estimate the minimum number of stator vanes that meet the given requirements of the design.

Since the flow through the two-dimensional cascade of stator vanes is infinitely periodic in the $y$-direction, we need to model only one flow passage through the vanes, specifying two pairs of periodic boundary conditions on the top and bottom edges of the computational domain (Fig. 15-51). We run six cases, each with a different value of blade spacing. We choose $s = 10, 20, 30, 40, 50,$ and $60$ cm, and generate a structured grid for

**FIGURE 15–48**
Streamlines produced by CFD calculations of stationary turbulent flow over a circular cylinder at $Re = 10^7$. Unfortunately, the predicted drag coefficient is still not accurate for this case.

**FIGURE 15–49**
Schematic diagram of the vane-axial flow fan being designed. The stator precedes the rotor, and the flow through the stator vanes is to be modeled with CFD.
each of these values of blade spacing. The grid for the case with $s = 10$ cm is shown in Fig. 15–52; the other grids are similar, but more intervals are specified in the $y$-direction as $s$ increases. Notice how we have made the grid spacing fine near the pressure and suction surfaces so that the boundary layer on these surfaces can be better resolved. We specify $V = 50$ m/s at the velocity inlet, zero gage pressure at the pressure outlet, and a smooth wall boundary condition with no slip at both the pressure and suction surfaces. Since we are modeling the flow with a turbulence model ($k$-$e$ with wall functions), we must also specify turbulence properties at the velocity inlet. For these simulations we specify a turbulence intensity of 10 percent and a turbulence length scale of 0.01 m (1.0 cm).

We run the CFD calculations long enough to converge as far as possible for all six cases, and we plot streamlines in Fig. 15–53 for six blade spacings: $s = 10, 20, 30, 40, 50,$ and 60 cm. Although we solve for flow through only one flow passage, we draw several duplicate flow passages, stacked one on top of the other, in order to visualize the flow field as a periodic cascade. The streamlines for the first three cases look very similar at first glance, but closer inspection reveals that the average angle of flow downstream of the trailing edge of the stator vane decreases with $s$. (We define flow angle $\beta$ relative to horizontal as sketched in Fig. 15–53a.) Also, the gap (white space) between the wall and the closest streamline to the suction surface increases in size as $s$ increases, indicating that the flow speed in that region decreases. In fact, it turns out that the boundary layer on the suction surface of the stator vane must resist an ever-increasingly adverse pressure gradient (decelerating flow speed and positive pressure gradient) as blade spacing is increased. At large enough $s$, the boundary layer on the suction surface cannot withstand the severely adverse pressure gradient and separates off the wall. For $s = 40, 50,$ and 60 cm (Fig. 15–53d through f), flow separation off the suction surface is clearly seen in these streamline plots. Furthermore, the severity of the flow separation increases with $s$. This is not
FIGURE 15–52
Structured grid for the two-dimensional stator vane cascade at blade spacing $s = 10$ cm. The outflow region in the wake of the vanes is intentionally longer than that at the inlet to avoid backflow at the pressure outlet in case of flow separation on the suction surface of the stator vane. The outlet is one chord length downstream of the stator vane trailing edges; the outlet is also the location of the leading edges of the rotor blades (not shown).

FIGURE 15–53
Streamlines produced by CFD calculations of stationary turbulent flow through a stator vane flow passage: (a) blade spacing $s = 10$, (b) 20, (c) 30, (d) 40, (e) 50, and (f) 60 cm. The CFD calculations are performed using the $k-\epsilon$ turbulence model with wall functions. Flow angle $\beta$ is defined in image (a) as the average angle of flow, relative to horizontal, just downstream of the trailing edge of the stator vane.
unexpected if we imagine the limit as $s \to \infty$. In that case, the stator vane is isolated from its neighbors, and we surely expect massive flow separation since the vane has such a high degree of camber.

We list average outlet flow angle $\beta_{\text{avg}}$, average outlet flow speed $V_{\text{avg}}$ and predicted drag force per unit depth $F_D/b$ in Table 15-3 as functions of blade spacing $s$. (Depth $b$ is into the page of Fig. 15-53 and is assumed to be 1 m in two-dimensional calculations such as these.) While $\beta_{\text{avg}}$ and $V_{\text{avg}}$ decrease continuously with $s$, $F_D/b$ first rises to a maximum for the $s = 30$ cm case, and then decreases from there on.

You may recall from the previously stated design criteria that the average outlet flow angle must be greater than $45^\circ$, and there must be no significant flow separation. From our CFD results, it appears that both of these criteria break down somewhere between $s = 30$ and 40 cm. We obtain a better picture of flow separation by plotting vorticity contours (Fig. 15-54). In these gray-scale contour plots, black represents large negative vorticity (clockwise rotation), white represents large positive vorticity (counterclockwise rotation), and middle gray is zero vorticity. If the boundary layer remains attached, we expect the vorticity to be concentrated within thin boundary layers along the

### TABLE 15–3

<table>
<thead>
<tr>
<th>$s$, cm</th>
<th>$\beta_{\text{avg}}$, degrees</th>
<th>$V_{\text{avg}}$, m/s</th>
<th>$F_D/b$, N/m</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>60.8</td>
<td>103</td>
<td>554</td>
</tr>
<tr>
<td>20</td>
<td>56.1</td>
<td>89.6</td>
<td>722</td>
</tr>
<tr>
<td>30</td>
<td>49.7</td>
<td>77.4</td>
<td>694</td>
</tr>
<tr>
<td>40</td>
<td>43.2</td>
<td>68.6</td>
<td>612</td>
</tr>
<tr>
<td>50</td>
<td>37.2</td>
<td>62.7</td>
<td>538</td>
</tr>
<tr>
<td>60</td>
<td>32.3</td>
<td>59.1</td>
<td>489</td>
</tr>
</tbody>
</table>

* All calculated values are reported to three significant digits. The CFD calculations are performed using the $k$-$\varepsilon$ turbulence model with wall functions.

### FIGURE 15–54

Vorticity contour plots produced by CFD calculations of stationary turbulent flow through a stator vane flow passage: blade spacing (a) $s = 30$ cm and (b) $s = 40$ cm. The flow field is largely irrotational (zero vorticity) except in the thin boundary layer along the walls and in the wake region. However, when the boundary layer separates, as in case (b), the vorticity spreads throughout the separated flow region.
stator vane surfaces, as is the case in Fig. 15-54a for $s = 30$ cm. If the boundary layer separates, however, the vorticity suddenly spreads out away from the suction surface, as seen in Fig. 15-54b for $s = 40$ cm. These results verify that significant flow separation occurs somewhere between $s = 30$ and $40$ cm. As a side note, notice how the vorticity is concentrated not only in the boundary layer, but also in the wake for both cases shown in Fig. 15-54.

Finally, we compare velocity vector plots in Fig. 15-55 for three cases: $s = 20$, 40, and 60 cm. We generate several equally spaced parallel lines in the computational domain; each line is tilted at 45° from horizontal. Velocity vectors are then plotted along each of these parallel lines. When $s = 20$ cm (Fig. 15-55a), the boundary layer remains attached on both the suction and pressure surfaces of the stator vane all the way to its trailing edge. When $s = 40$ cm (Fig. 15-55b), flow separation and reverse flow along the suction surface appears. When $s = 60$ cm (Fig. 15-55c), the separation bubble and the reverse flow region have grown—this is a “dead” flow region, in which the air speeds are very small. In all cases, the flow on the pressure surface of the stator vane remains attached.

How many vanes ($N$) does a blade spacing of $s = 30$ cm represent? We can easily calculate $N$ by noting that at the vane tip ($r = r_{tip} = D/2 = 50$ cm), where $s$ is largest, the total available circumference ($C$) is

Available circumference: $C = 2\pi r_{tip} = \pi D$ \hspace{1cm} (15-7)

The number of vanes that can be placed within this circumference with a blade spacing of $s = 30$ cm is thus

Maximum number of vanes: $N = \frac{C}{s} = \frac{\pi D}{s} = \frac{\pi(100 \text{ cm})}{30 \text{ cm}} = 10.5$ \hspace{1cm} (15-8)

Obviously we can have only an integer value of $N$, so we conclude from our preliminary analysis that we should have at least 10 or 11 stator vanes.

How good is our approximation of the stator as a two-dimensional cascade of vanes? To answer this question, we perform a full three-dimensional CFD analysis of the stator. Again we take advantage of the periodicity by modeling only one flow passage—a three-dimensional passage between two radial stator vanes (Fig. 15-56). We choose $N = 10$ stator vanes by specifying an angle of periodicity of $360/10 = 36°$. From Eq. 15-8, this represents $s = 31.4$ at the vane tips and $s = 15.7$ at the hub, for an average value of $s_{avg} = 23.6$. We generate a hexagonal structured grid in a computational domain bounded by a velocity inlet, an outflow outlet, a section of cylindrical wall at the hub and another at the tip, the pressure surface of the vane, the suction surface of the vane, and two pairs of periodic boundary conditions. In this three-dimensional case, the periodic boundaries are rotationally periodic instead of translationally periodic. Note that we use an outflow boundary condition rather than a pressure outlet boundary condition, because we expect the swirling motion to produce a radial pressure distribution on the outlet face. The grid is finer near the walls than elsewhere (as usual), to better resolve the boundary layer. The incoming velocity, turbulence level, turbulence model, etc., are all the same as those used for the two-dimensional approximation. The total number of computational cells is almost 800,000.
Pressure contours on the stator vane surfaces and on the inner cylindrical wall are plotted in Fig. 15–57. This view is from the same angle as that of Fig. 15–56, but we have zoomed out and duplicated the computational domain nine times circumferentially about the axis of rotation (the x-axis) for a total of 10 flow passages to aid in visualization of the flow field. You can see that the pressure is higher (lighter shade of gray) on the pressure surface than on the suction surface. You can also see an overall drop in pressure along the hub surface from upstream to downstream of the stator. The change in average pressure from the inlet to the outlet is calculated to be 3.29 kPa.

To compare our three-dimensional results directly with the two-dimensional approximation, we run one additional two-dimensional case at the average blade spacing, \( s = s_{avg} = 23.6 \) cm. A comparison between the two- and three-
dimensional cases is shown in Table 15–4. From the three-dimensional calculation, the net axial force on one stator vane is $F_D = 183\ N$. We compare this to the two-dimensional value by converting to force per unit depth (force per unit span of the stator vane). Since the stator vane spans 0.25 m, $F_D/b = (183\ N)/(0.25\ m) = 732\ N/m$. The corresponding two-dimensional value from Table 15–4 is $F_D/b = 724\ N/m$, so the agreement is very good (≈ 1 percent difference). The average speed at the outlet of the three-dimensional domain is $V_{avg} = 84.7\ m/s$, almost identical to the two-dimensional value of 84.8 m/s in Table 15–4. The two-dimensional approximation differs by less than 1 percent. Finally, the average outlet flow angle $\beta_{avg}$ obtained from our full three-dimensional calculation is 53.3°, which easily meets the design criterion of 45°. We compare this to the two-dimensional approximation of 53.9° in Table 15–4; the agreement is again around 1 percent.

Contours of tangential velocity component at the outlet of the computational domain are plotted in Fig. 15–58. We see that the tangential velocity distribution is not uniform; it decreases as we move radially outward from hub to tip as we should expect, since blade spacing $s$ increases from hub to tip. We also find (not shown here) that the outflow pressure increases radially from hub to tip. This also agrees with our intuition, since we know that a radial pressure gradient is required to sustain a tangential flow—the pressure rise with increasing radius provides the centripetal acceleration necessary to turn the flow about the $x$-axis.

A nother comparison can be made between the three-dimensional and two-dimensional calculations by plotting vorticity contours in a slice through the computational domain within the flow passage between vanes. Two such

![Grayscale tangential velocity contour plot produced by three-dimensional CFD calculations of stationary turbulent flow through a stator vane flow passage. The tangential velocity component is shown in m/s at the outlet of the computational domain (and also on the vane surfaces, where the velocity is zero). An outline of the inlet to the computational domain is also shown for clarity. Although only one flow passage is modeled, we duplicate the image circumferentially around the $x$-axis nine times to visualize the entire stator flow field. In this grayscale image, the tangential velocity ranges from 0 (black) to 90 m/s (white).](image)

### TABLE 15–4

<table>
<thead>
<tr>
<th>2-D, $s = 23.6\ cm$</th>
<th>Full 3-D</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta_{avg}$</td>
<td>53.9°</td>
</tr>
<tr>
<td>$V_{avg}$, m/s</td>
<td>84.8</td>
</tr>
<tr>
<td>$F_D/b$, N/m</td>
<td>724</td>
</tr>
</tbody>
</table>

* Values are shown to three significant digits.
slices are created—a slice close to the hub and a slice close to the tip, and vorticity contours are plotted in Fig. 15–59. In both slices, the vorticity is confined to the thin boundary layer and wake. There is no flow separation near the hub, but we see that near the tip, the flow has just begun to separate on the suction surface near the trailing edge of the stator vane. Notice that the air leaves the trailing edge of the vane at a steeper angle at the hub than at the tip. This also agrees with our two-dimensional approximation (and our intuition), since blade spacing $s$ at the hub (15.7 cm) is smaller than $s$ at the tip (31.4 cm).

In conclusion, the approximation of this three-dimensional stator as a two-dimensional cascade of stator vanes turns out to be quite good overall, particularly for preliminary analysis. The discrepancy between the two- and three-dimensional calculations for gross flow features, such as force on the vane, outlet flow angle, etc., is around 1 percent or less for all reported quantities. It is therefore no wonder that the two-dimensional cascade approach is such a popular approximation in turbomachinery design. The more detailed three-dimensional analysis gives us confidence that a stator
with 10 vanes is sufficient to meet the imposed design criteria for this axial-flow fan. However, our three-dimensional calculations have revealed a small separated region near the tip of the stator vane. It may be wise to apply some twist to the stator vanes (reduce the pitch angle or angle of attack toward the tip) in order to avoid this separation. (Twist is discussed in more detail in Chap. 14.) Alternatively, we can increase the number of stator vanes to 11 or 12 to hopefully eliminate flow separation at the vane tips.

As a final comment on this example flow field, all the calculations were performed in a fixed coordinate system. Modern CFD codes contain options for modeling zones in the flow field with rotating coordinate systems so that similar analyses can be performed on rotor blades as well as on stator vanes.

15–4 = CFD WITH HEAT TRANSFER

By coupling the differential form of the energy equation with the equations of fluid motion, we can use a computational fluid dynamics code to calculate properties associated with heat transfer (e.g., temperature distributions or rate of heat transfer from a solid surface to a fluid). Since the energy equation is a scalar equation, only one extra transport equation (typically for either temperature or enthalpy) is required, and the computational expense (CPU time and RAM requirements) is not increased significantly. Heat transfer capability is built into most commercially available CFD codes, since many practical problems in engineering involve both fluid flow and heat transfer. As mentioned previously, additional boundary conditions related to heat transfer need to be specified. At solid wall boundaries, we may specify either wall temperature \( T_{\text{wall}} \) (K) or the wall heat flux \( q_{\text{wall}} \) (W/m\(^2\)), defined as the rate of heat transfer per unit area from the wall to the fluid (but not both at the same time, as illustrated in Fig. 15–60). When we model a zone in a computational domain as a solid body that involves the generation of thermal energy via electric heating (as in electronic components) or chemical or nuclear reactions (as in nuclear fuel rods), we may instead specify the heat generation rate per unit volume \( g \) (W/m\(^3\)) within the solid since the ratio of the total heat generation rate to the exposed surface area must equal the average wall heat flux. In that case, neither \( T_{\text{wall}} \) nor \( q_{\text{wall}} \) are specified; both converge to values that match the specified heat generation rate. In addition, the temperature distribution inside the solid object itself can then be calculated. Other boundary conditions (such as those associated with radiation heat transfer) may also be applied in CFD codes.

In this section we do not go into details about the equations of motion or the numerical techniques used to solve them. Rather, we show some basic examples that illustrate the capability of CFD to calculate practical flows of engineering interest that involve heat transfer.

Temperature Rise through a Cross-Flow Heat Exchanger

Consider flow of cool air through a series of hot tubes as sketched in Fig. 15–61. In heat exchanger terminology, this geometrical configuration is called a cross-flow heat exchanger. If the airflow were to enter horizontally \( (\alpha = 0) \) at all times, we could cut the computational domain in half and

**FIGURE 15–60**

At a wall boundary, we may specify either (a) the wall temperature or (b) the wall heat flux, but not both, as this would be mathematically overspecified.

**FIGURE 15–61**

The computational domain (light blue shaded region) used to model turbulent flow through a cross-flow heat exchanger. Flow enters from the left at angle \( \alpha \) from the horizontal.
apply symmetry boundary conditions on the top and bottom edges of the domain (see Fig. 15–23). In the case under consideration, however, we allow the airflow to enter the computational domain at some angle \( \alpha \neq 0 \). Thus, we impose translationally periodic boundary conditions on the top and bottom edges of the domain as sketched in Fig. 15–61. We set the inlet air temperature to 300 K and the surface temperature of each tube to 500 K. The diameter of the tubes and the speed of the air are chosen such that the Reynolds number is approximately \( 1 \times 10^5 \) based on tube diameter. The tube surfaces are assumed to be hydrodynamically smooth (zero roughness) in this first set of calculations. The hot tubes are staggered as sketched in Fig. 15–61 and are spaced three diameters apart both horizontally and vertically. We assume two-dimensional stationary turbulent flow without gravity effects and set the turbulence intensity of the inlet air to 10 percent. We run two cases for comparison: \( \alpha = 0 \) and \( 10^\circ \). Our goal is to see whether the heat transfer to the air is enhanced or inhibited by a nonzero value of \( \alpha \). Which case do you think will provide greater heat transfer?

We generate a two-dimensional, multiblock, structured grid with very fine resolution near the tube walls as shown in Fig. 15–62, and we run the CFD code to convergence for both cases. Temperature contours are shown for the \( \alpha = 0^\circ \) case in Fig. 15–63, and for the \( \alpha = 10^\circ \) case in Fig. 15–64. The average rise of air temperature leaving the outlet of the control volume for the case with \( \alpha = 0^\circ \) is 5.51 K, while that for \( \alpha = 10^\circ \) is 5.65 K. Thus we conclude that the off-axis inlet flow leads to more effective heating of the air, although the improvement is only about 2.5 percent. We compute a third case (not shown) in which \( \alpha = 0^\circ \) but the turbulence intensity of the incoming air is increased to 25 percent. This leads to improved mixing, and the average air temperature rise from inlet to outlet increases by about 6.5 percent to 5.87 K.

Finally, we study the effect of rough tube surfaces. We model the tube walls as rough surfaces with a characteristic roughness height of 0.01 m (1 percent of cylinder diameter). Note that we had to coarsen the grid somewhat near each tube so that the distance from the center of the closest computational cell to the wall is greater than the roughness height; otherwise the
The roughness model in the CFD code is unphysical. The flow inlet angle is set to $\alpha = 0^\circ$ for this case, and flow conditions are identical to those of Fig. 15–63. Temperature contours are plotted in Fig. 15–65. Pure white regions in the contour plot represent locations where the air temperature is greater than 315 K. The average air temperature rise from inlet to outlet is 14.48 K, a 163 percent increase over the smooth wall case at $\alpha = 0^\circ$. Thus we see that wall roughness is a critical parameter in turbulent flows. This example provides some insight as to why the tubes in heat exchangers are often purposely roughened.

**Cooling of an Array of Integrated Circuit Chips**

In electronics equipment, instrumentation, and computers, electronic components, such as integrated circuits (ICs or “chips”), resistors, transistors, diodes, and capacitors, are soldered onto printed circuit boards (PCBs).
The PCBs are often stacked in rows as sketched in Fig. 15–66. Because many of these electronic components must dissipate heat, cooling air is often blown through the air gap between each pair of PCBs to keep the components from getting too hot. Consider the design of a PCB for an outer space application. Several identical PCBs are to be stacked as in Fig. 15–66. Each PCB is 10 cm high and 30 cm long, and the spacing between boards is 2.0 cm. Cooling air enters the gap between the PCBs at a speed of 2.60 m/s and a temperature of 30°C. The electrical engineers must fit eight identical ICs on a 10 cm × 15 cm portion of each board. Each of the ICs dissipates 6.24 W of heat: 5.40 W from its top surface and 0.84 W from its sides. (There is assumed to be no heat transfer from the bottom of the chip to the PCB.) The rest of the components on the board have negligible heat transfer compared to that from the eight ICs. To ensure adequate performance, the average temperature on the chip surface should not exceed 150°C, and the maximum temperature anywhere on the surface of the chip should not exceed 180°C. Each chip is 2.5 cm wide, 4.5 cm long, and 0.50 cm thick.

The electrical engineers come up with two possible configurations of the eight chips on the PCB as sketched in Fig. 15–67: in the long configuration, the chips are aligned with their long dimension parallel to the flow, and in the short configuration, the chips are aligned with their short dimension parallel to the flow. The chips are staggered in both cases to enhance cooling. We are to determine which arrangement leads to the lower maximum surface temperature on the chips, and whether the electrical engineers will meet the surface temperature requirements.

For each configuration, we define a three-dimensional computational domain consisting of a single flow passage through the air gap between two PCBs (Fig. 15–68). We generate a structured hexagonal grid with 267,520 cells for each configuration. The Reynolds number based on the 2.0-cm gap between boards is about 3600. If this were a simple two-dimensional channel flow, this Reynolds number would be barely high enough to establish turbulent flow. However, since the surfaces leading up to the velocity inlet are very rough, the flow is most likely turbulent. We note that low Reynolds number turbulent flows are challenging for most turbulence models, since the models are calibrated at high Reynolds numbers. Nevertheless, we...
assume stationary turbulent flow and employ the $k$-$\epsilon$ turbulence model with wall functions. While the absolute accuracy of these calculations may be suspect because of the low Reynolds number, comparisons between the long and short configurations should be reasonable. We ignore buoyancy effects in the calculations since this is a space application. The inlet is specified as a velocity inlet (air) with $V = 2.60$ m/s and $T_w = 30^\circ$C; we set the inlet turbulence intensity to 20 percent and the turbulent length scale to 1.0 mm. The outlet is a pressure outlet at zero gage pressure. The PCB is modeled as a smooth adiabatic wall (zero heat transfer from the wall to the air). The top and sides of the computational domain are also approximated as smooth adiabatic walls.

Based on the given chip dimensions, the surface area of the top of a chip is $4.5 \text{ cm} \times 2.5 \text{ cm} = 11.25 \text{ cm}^2$. The total surface area of the four sides of the chip is $7.0 \text{ cm}^2$. From the given heat transfer rates, we calculate the rate of heat transfer per unit area from the top surface of each chip,

$$q_{\text{top}} = \frac{5.4 \text{ W}}{11.25 \text{ cm}^2} = 0.48 \text{ W/cm}^2$$

So, we model the top surface of each chip as a smooth wall with a surface heat flux of 4800 W/m$^2$ from the wall to the air. Similarly, the rate of heat transfer per unit area from the sides of each chip is

$$q_{\text{sides}} = \frac{0.84 \text{ W}}{7.0 \text{ cm}^2} = 0.12 \text{ W/cm}^2$$

Since the sides of the chip have electrical leads, we model each side surface of each chip as a rough wall with an equivalent roughness height of 0.50 mm and a surface heat flux of 1200 W/m$^2$ from the wall to the air.

The CFD code FLUENT is run for each case to convergence. Results are summarized in Table 15-5, and temperature contours are plotted in Figs. 15-69 and 15-70. The average temperature on the top surfaces of the chips is about the same for either configuration (144.4°C for the long case and 144.7°C for the short case) and is below the recommended limit of 150°C.
There is more of a difference in average temperature on the side surfaces of the chips, however (84.2°C for the long case and 91.4°C for the short case), although these values are well below the limit. Of greatest concern are the maximum temperatures. For the long configuration, $T_{\text{max}} = 187.5°C$ and occurs on the top surface of chip 7 (the middle chip of the last row). For the short configuration, $T_{\text{max}} = 182.1°C$ and occurs close to midboard on the top surfaces of chips 7 and 8 (the two chips in the last row). For both configurations these values exceed the recommended limit of 180°C, although not by much. The short configuration does a better job at cooling the top surfaces of the chips, but at the expense of a slightly larger pressure drop and poorer cooling along the side surfaces of the chips.

Notice from Table 15–5 that the average change in air temperature from inlet to outlet is identical for both configurations (7.83°C). This should not be

### TABLE 15–5
Comparison of CFD results for the chip cooling example, long and short configurations

<table>
<thead>
<tr>
<th></th>
<th>Long</th>
<th>Short</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{\text{max}}$, top surfaces of chips</td>
<td>187.5°C</td>
<td>182.1°C</td>
</tr>
<tr>
<td>$T_{\text{avg}}$, top surfaces of chips</td>
<td>144.5°C</td>
<td>144.7°C</td>
</tr>
<tr>
<td>$T_{\text{max}}$, side surfaces of chips</td>
<td>154.0°C</td>
<td>170.6°C</td>
</tr>
<tr>
<td>$T_{\text{avg}}$, side surfaces of chips</td>
<td>84.2°C</td>
<td>91.4°C</td>
</tr>
<tr>
<td>Average $\Delta T$, inlet to outlet</td>
<td>7.83°C</td>
<td>7.83°C</td>
</tr>
<tr>
<td>Average $\Delta P$, inlet to outlet</td>
<td>−5.14 Pa</td>
<td>−5.58 Pa</td>
</tr>
</tbody>
</table>

**FIGURE 15–69**
CFD results for the chip cooling example, long configuration: grayscale temperature contours as viewed from directly above the chip surfaces, with $T$ values in K on the legend. The location of maximum surface temperature is indicated, it occurs near the end of chip 7. Light regions near the leading edges of chips 1, 2, and 3 are also seen, indicating high surface temperatures at those locations.
surprising, because the total rate of heat transferred from the chips to the air is the same regardless of chip configuration. In fact, in a CFD analysis it is wise to check values like this— if average ΔT were not the same between the two configurations, we would suspect some kind of error in our calculations.

There are many other interesting features of these flow fields that we can point out. For either configuration, the average surface temperature on the downstream chips is greater than that on the upstream chips. This makes sense physically, since the first chips receive the coolest air, while those downstream are cooled by air that has already been warmed up somewhat. We notice that the front chips (1, 2, and 3 in the long configuration and 1 and 2 in the short configuration) have regions of high temperature just downstream of their leading edges. A close-up view of the temperature distribution on one of these chips is shown in Fig. 15–71a. Why is the temperature so high there? It turns out that the flow separates off the sharp corner at the front of the chip and forms a recirculating eddy called a separation bubble on the top of the chip (Fig. 15–71b). The air speed is slow in that region, especially along the reattachment line where the flow reattaches to the surface. The slow air speed leads to a local "hot spot" in that region of the chip surface since convective cooling is minimal there. Finally, we notice in Fig. 15–71a that downstream of the separation bubble, T increases down the chip surface. There are two reasons for this: (1) the air warms up as it travels down the chip, and (2) the boundary layer on the chip surface grows downstream. The larger the boundary layer thickness, the lower the air speed near the surface, and thus the lower the amount of convective cooling at the surface.

**FIGURE 15–70**

CFD results for the chip cooling example, short configuration: grayscale temperature contours as viewed from directly above the chip surfaces, with T values in K on the legend. The same temperature scale is used here as in Fig. 15–69. The locations of maximum surface temperature are indicated; they occur near the end of chips 7 and 8 near the center of the PCB. Light regions near the leading edges of chips 1 and 2 are also seen, indicating high surface temperatures at those locations.
In summary, our CFD calculations have predicted that the short configuration leads to a lower value of maximum temperature on the chip surfaces and appears at first glance to be the preferred configuration for heat transfer. However, the short configuration demands a higher pressure drop at the same volume flow rate (Table 15–5). For a given cooling fan, this additional pressure drop would shift the operating point of the fan to a lower volume flow rate (Chap. 14), decreasing the cooling effect. It is not known whether this shift would be enough to favor the long configuration—more information about the fan and more analysis would be required. The bottom line in either case is that there is not sufficient cooling to keep the chip surface temperature below 180°C everywhere on every chip. To rectify the situation, we recommend that the designers spread the eight hot chips over the entire PCB rather than in the limited 10 cm × 15 cm area of the board. The increased space between chips should result in sufficient cooling for the given flow rate. Another option is to install a more powerful fan that would increase the speed of the inlet air.

15–5 = COMPRESSIBLE FLOW CFD CALCULATIONS

All the examples discussed in this chapter so far have been for incompressible flow (\( \rho = \text{constant} \)). When the flow is compressible, density is no longer a constant, but becomes an additional variable in the equation set. We limit our discussion here to ideal gases. When we apply the ideal-gas law, we introduce yet another unknown, namely, temperature \( T \). Hence, the energy equation must be solved along with the compressible forms of the equations of conservation of mass and conservation of momentum (Fig. 15–72). In addition, fluid properties, such as viscosity and thermal conductivity, are no longer necessarily treated as constants, since they are functions of temperature; thus, they appear inside the derivative operators in the differential equations of Fig. 15–72. While the equation set looks ominous, many commercially available CFD codes are able to handle compressible flow problems, including shock waves.

When solving compressible flow problems with CFD, the boundary conditions are somewhat different than those of incompressible flow. For example, at a pressure inlet we need to specify both stagnation pressure and static pressure, along with stagnation temperature. A special boundary condition (called pressure far field in FLUENT) is also available for compressible flows. With this boundary condition, we specify the Mach number, the static pressure, and the temperature; it can be applied to both inlets and outlets and is well-suited for supersonic external flows.

The equations of Fig. 15–72 are for laminar flow, whereas many compressible flow problems occur at high flow speeds in which the flow is turbulent. Therefore, the equations of Fig. 15–72 must be modified accordingly (into the RANS equation set) to include a turbulence model, and more transport equations must be added, as discussed previously. The equations then get quite long and complicated and are not included here. Fortunately, in many situations, we can approximate the flow as inviscid, eliminating the viscous terms from the equations of Fig. 15–72 (the Navier–Stokes equation reduces to the Euler equation). As we shall see, the inviscid flow approximation turns out to be quite good for many practical high-speed flows, since the boundary layers along walls are very thin at high Reynolds numbers. In
The equations of motion for the case of steady, compressible, laminar flow of a Newtonian fluid in Cartesian coordinates are:

\[
\begin{align*}
\text{Continuity:} & \quad \frac{\partial (\rho u)}{\partial x} + \frac{\partial (\rho v)}{\partial y} + \frac{\partial (\rho w)}{\partial z} = 0 \\
\text{Ideal gas law:} & \quad P = \rho RT \\
\text{x-momentum:} & \quad \rho u \left( \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + w \frac{\partial u}{\partial z} \right) = \rho g_x + \frac{\partial P}{\partial x} + \frac{\partial}{\partial x} \left( \mu \frac{\partial u}{\partial x} + \frac{\partial P}{\partial x} + \lambda \nabla^2 u \right) + \frac{\partial}{\partial y} \left( \mu \frac{\partial u}{\partial y} + \frac{\partial P}{\partial y} + \lambda \nabla^2 u \right) + \frac{\partial}{\partial z} \left( \mu \frac{\partial u}{\partial z} + \frac{\partial P}{\partial z} + \lambda \nabla^2 u \right) \\
\text{y-momentum:} & \quad \rho v \left( \frac{\partial v}{\partial x} + u \frac{\partial v}{\partial y} + w \frac{\partial v}{\partial z} \right) = \rho g_y + \frac{\partial P}{\partial y} + \frac{\partial}{\partial x} \left( \mu \frac{\partial v}{\partial x} + \frac{\partial P}{\partial x} + \lambda \nabla^2 v \right) + \frac{\partial}{\partial y} \left( \mu \frac{\partial v}{\partial y} + \frac{\partial P}{\partial y} + \lambda \nabla^2 v \right) + \frac{\partial}{\partial z} \left( \mu \frac{\partial v}{\partial z} + \frac{\partial P}{\partial z} + \lambda \nabla^2 v \right) \\
\text{z-momentum:} & \quad \rho w \left( \frac{\partial w}{\partial x} + u \frac{\partial w}{\partial y} + v \frac{\partial w}{\partial z} \right) = \rho g_z + \frac{\partial P}{\partial z} + \frac{\partial}{\partial x} \left( \mu \frac{\partial w}{\partial x} + \frac{\partial P}{\partial x} + \lambda \nabla^2 w \right) + \frac{\partial}{\partial y} \left( \mu \frac{\partial w}{\partial y} + \frac{\partial P}{\partial y} + \lambda \nabla^2 w \right) + \frac{\partial}{\partial z} \left( \mu \frac{\partial w}{\partial z} + \frac{\partial P}{\partial z} + \lambda \nabla^2 w \right) \\
\text{Energy:} & \quad \rho c_p \left( \frac{\partial T}{\partial x} + u \frac{\partial T}{\partial y} + v \frac{\partial T}{\partial z} \right) = \beta \left( \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + w \frac{\partial u}{\partial z} \right) + \nabla \cdot (k \nabla T) + \Phi
\end{align*}
\]

The equations of motion are the basic form used in CFD, but they are often extended to include the effect of turbulence. In this case, the dissipation function \( \Phi \) is given by White (1991) as:

\[
\Phi = 2\mu \left( \frac{\partial u}{\partial x} \right)^2 + 2\mu \left( \frac{\partial v}{\partial y} \right)^2 + 2\mu \left( \frac{\partial w}{\partial z} \right)^2 + \mu \left( \frac{\partial w}{\partial x} + \frac{\partial w}{\partial y} \right)^2 + \mu \left( \frac{\partial w}{\partial x} + \frac{\partial w}{\partial z} \right)^2 + \mu \left( \frac{\partial w}{\partial y} + \frac{\partial w}{\partial z} \right)^2 + \lambda \left( \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} + \frac{\partial w}{\partial z} \right)^2
\]

In fact, compressible CFD calculations can predict flow features that are often quite difficult to obtain experimentally. For example, many experimental measurement techniques require optical access, which is limited in three-dimensional flows, and even in some axisymmetric flows. CFD is not limited in this way.

**Compressible Flow through a Converging–Diverging Nozzle**

For our first example, we consider compressible flow of air through an axisymmetric converging–diverging nozzle. The computational domain is shown in Fig. 15–73. The inlet radius is 0.10 m, the throat radius is 0.075 m, and the outlet radius is 0.12 m. The axial distance from the inlet to the throat is 0.30 m—the same as the axial distance from the throat to the outlet. A structured grid with approximately 12,000 quadrilateral cells is used in the calculations. At the pressure inlet boundary, the stagnation pressure \( P_{0,\text{inlet}} \) is set to 220 kPa (absolute), the static pressure \( P_{\text{inlet}} \) is set to 210 kPa, and the stagnation temperature \( T_{0,\text{inlet}} \) is set to 300 K. For the first case, we set the static pressure \( P_0 \) at the pressure outlet boundary to 50.0 kPa \( (P_0/P_{0,\text{inlet}} = 0.227) \)—low enough that the flow is supersonic through the entire diverging section of the nozzle, without any normal shocks in the nozzle throat.
nozzle. This back pressure ratio corresponds to a value between cases E and F in Fig. 12–27, in which a complex shock pattern occurs downstream of the nozzle exit; these shock waves do not influence the flow in the nozzle itself, since the flow exiting the nozzle is supersonic. We do not attempt to model the flow downstream of the nozzle exit.

The CFD code is run to convergence in its steady, inviscid, compressible flow mode. The average values of the Mach number $M_a$ and pressure ratio $P/P_{0,\text{inlet}}$ are calculated at 25 axial locations along the converging-diverging nozzle (every 0.025 m) and are plotted in Fig. 15–74a. The results match almost perfectly with the predictions of one-dimensional isentropic flow (Chap. 12). At the throat ($x = 0.30$ m), the average Mach number is 0.997, and the average value of $P/P_{0,\text{inlet}}$ is 0.530. One-dimensional isentropic flow theory predicts $M_a = 1$ and $P/P_{0,\text{inlet}} = 0.528$ at the throat. The small discrepancies between CFD and theory are due to the fact that the computed flow is not one-dimensional, since there is a radial velocity component and, therefore, a radial variation of the Mach number and static pressure. Careful examination of the Mach number contour lines of Fig. 15–74b reveal that they are curved, not straight as would be predicted by one-dimensional isentropic

![FIGURE 15–74](image)

CFD results for steady, adiabatic, inviscid compressible flow through an axisymmetric converging-diverging nozzle: (a) calculated average Mach number and pressure ratio at 25 axial locations (circles), compared to predictions from isentropic, one-dimensional compressible flow theory (solid lines); (b) grayscale Mach number contours, ranging from $M_a = 0.3$ (darkest) to 2.7 (lightest). Although only the top half is calculated, a mirror image about the x-axis is shown for clarity. The sonic line ($M_a = 1$) is also highlighted. It is parabolic rather than straight in this axisymmetric flow due to the radial component of velocity, as discussed in Schreier (1982).
theory. The sonic line (Ma = 1) is identified for clarity in the figure. Although Ma = 1 right at the wall of the throat, sonic conditions along the axis of the nozzle are not reached until somewhat downstream of the throat.

Next, we run a series of cases in which back pressure \( P_b \) is varied, while keeping all other boundary conditions fixed. Results for three cases are shown in Fig. 15–75: \( P_b = (a) \) 100, (b) 150, and (c) 200 kPa, i.e., \( P_b / P_{0, \text{inlet}} = (a) \) 0.455, (b) 0.682, and (c) 0.909, respectively. For all three cases, a normal shock occurs in the diverging portion of the nozzle. Furthermore, as back pressure increases, the shock moves upstream toward the throat, and decreases in strength. Since the flow is choked at the throat, the mass flow rate is identical in all three cases (and also in the previous case shown in Fig. 15–74). We notice that the normal shock is not straight, but rather is curved due to the radial component of velocity, as previously mentioned.

For case (b), in which \( P_b / P_{0, \text{inlet}} = 0.682 \), the average values of the Mach number and pressure ratio \( P / P_{0, \text{inlet}} \) are calculated at 25 axial locations along the converging-diverging nozzle (every 0.025 m), and are plotted in Fig. 15–76. For comparison with theory, the one-dimensional isentropic flow relations are used upstream and downstream of the shock, and the normal
shock relations are used to calculate the pressure jump across the shock (Chap. 12). To match the specified back pressure, one-dimensional analysis requires that the normal shock be located at $x = 0.4436$ m, accounting for the change in both $P_0$ and $A^*$ across the shock. The agreement between CFD calculations and one-dimensional theory is again excellent. The small discrepancy in both the pressure and the Mach number just downstream of the shock is attributed to the curved shape of the shock (Fig. 15–75b), as discussed previously. In addition, the shock in the CFD calculations is not infinitesimally thin, as predicted by one-dimensional theory, but is spread out over a few computational cells. The latter inaccuracy can be reduced somewhat by refining the grid in the area of the shock wave (not shown).

The previous CFD calculations are for steady, inviscid, adiabatic flow. When there are no shock waves (Fig. 15–74), the flow is also isentropic, since it is both adiabatic and reversible (no irreversible losses). However, when a shock wave exists in the flow field (Fig. 15–75), the flow is no longer isentropic since there are irreversible losses across the shock, although it is still adiabatic.

One final CFD case is run in which two additional irreversibilities are included, namely, friction and turbulence. We modify case (b) of Fig. 15–75 by running a steady, adiabatic, turbulent case using the $k$-$\varepsilon$ turbulence model with wall functions. The turbulence intensity at the inlet is set to 10 percent with a turbulence length scale of 0.050 m. A contour plot of $P/P_0$, inlet is shown in Fig. 15–77, using the same grayscale range as in Fig. 15–75. Comparison of Figs. 15–75b and 15–77 reveals that the shock wave for the turbulent case occurs further upstream and is therefore somewhat weaker. In addition, the stagnation pressure is small in a very thin region along the channel walls. This is due to frictional losses in the thin boundary layer. Turbulent and viscous irreversibilities in the boundary layer region are responsible for this decrease in stagnation pressure. Furthermore, the boundary layer separates just downstream of the shock, leading to more

**FIGURE 15–76**
Mach number and pressure ratio as functions of axial distance along a converging-diverging nozzle for the case in which $P_b/P_0$, inlet = 0.682. Averaged CFD results at 25 axial locations (circles) for steady, inviscid, adiabatic, compressible flow are compared to predictions from one-dimensional compressible flow theory (solid lines).
irreversibilities. A close-up view of velocity vectors in the vicinity of the separation point along the wall is shown in Fig. 15–78. We note that this case does not converge well and is inherently unsteady; the interaction between shock waves and boundary layers is a very difficult task for CFD. Because we use wall functions, flow details within the turbulent boundary layer are not resolved in this CFD calculation. Experiments reveal, however, that the shock wave interacts much more significantly with the boundary layer, producing "l-feet," as discussed in the Application Spotlight of Chap. 12.

Finally, we compare the mass flow rate for this viscous, turbulent case to that of the inviscid case, and find that \( m \) has decreased by about 0.7 percent. Why? As discussed in Chap. 10, a boundary layer along a wall impacts the outer flow such that the wall appears to be thicker by an amount equal to the displacement thickness \( \delta^* \). The effective throat area is thus reduced somewhat by the presence of the boundary layer, leading to a reduction in mass flow rate through the converging-diverging nozzle. The effect is small in this example since the boundary layers are so thin relative to the dimensions of the nozzle, and it turns out that the inviscid approximation is quite good (less than one percent error).

**Oblique Shocks over a Wedge**

For our final compressible flow example, we model steady, adiabatic, two-dimensional, inviscid, compressible flow of air over a wedge of half-angle \( \theta \) (Fig. 15–79). Since the flow has top-bottom symmetry, we model only the upper half of the flow and use a symmetry boundary condition along the bottom edge. We run three cases: \( \theta = 10, 20, \) and \( 30^\circ \), at an inlet Mach number of 2.0. CFD results are shown in Fig. 15–80 for all three cases. In the CFD plots, a mirror image of the computational domain is projected across the line of symmetry for clarity.

For the 10° case (Fig. 15–80a), a straight oblique shock originating at the apex of the wedge is observed, as also predicted by inviscid theory. The flow turns across the oblique shock by 10° so that it is parallel to the wedge wall. The shock angle \( \beta \) predicted by inviscid theory is 39.31°, and the predicted Mach number downstream of the shock is 1.64. Measurements with a protractor on Fig. 15–80a yield \( \beta = 40^\circ \), and the CFD calculation of the Mach number downstream of the shock is 1.64; the agreement with theory is thus excellent.
For the 20° case (Fig. 15–80b), the CFD calculations yield a Mach number of 1.21 downstream of the shock. The shock angle measured from the CFD calculations is about 54°. Inviscid theory predicts a Mach number of 1.21 and a shock angle of 53.4°, so again the agreement between theory and CFD is excellent. Since the shock for the 20° case is at a steeper angle (closer to a normal shock), it is stronger than the shock for the 10° case, as indicated by the darker coloring in the Mach contours downstream of the shock as $\theta$ increases.

At Mach number 2.0 in air, inviscid theory predicts that a straight oblique shock can form up to a maximum wedge half-angle of about 23° (Chap. 12). At wedge half-angles greater than this, the shock must move upstream of the wedge (become detached), forming a detached shock, which takes the shape of a bow wave (Chap. 12). The CFD results at $\theta = 30^\circ$ (Fig. 15–80c) show that this is indeed the case. The portion of the detached shock just upstream of the leading edge is a normal shock, and thus the flow downstream of that portion of the shock is subsonic. As the shock number increases, it becomes progressively weaker, and the Mach number downstream of the shock increases, as indicated by the lighter shade of gray.

**FIGURE 15–80**

CFD results (grayscale Mach number contours) for steady, adiabatic, inviscid, compressible flow at $M_{a_1} = 2.0$ over a wedge of half-angle $\theta = (a) 10^\circ$, (b) 20°, and (c) 30°. The Mach number contours range from $M_a = 0.2$ (darkest) to 2.0 (lightest) in all cases. For the two smaller wedge half-angles, an attached weak oblique shock forms at the leading edge of the wedge, but for the 30° case, a detached shock (bow wave) forms ahead of the wedge. Shock strength increases with $\theta$, as indicated by the darker shade of gray downstream of the shock as $\theta$ increases.

For the 20° case (Fig. 15-80b), the CFD calculations yield a Mach number of 1.21 downstream of the shock. The shock angle measured from the CFD calculations is about 54°. Inviscid theory predicts a Mach number of 1.21 and a shock angle of 53.4°, so again the agreement between theory and CFD is excellent. Since the shock for the 20° case is at a steeper angle (closer to a normal shock), it is stronger than the shock for the 10° case, as indicated by the darker coloring in the Mach contours downstream of the shock for the 20° case.

At Mach number 2.0 in air, inviscid theory predicts that a straight oblique shock can form up to a maximum wedge half-angle of about 23° (Chap. 12). At wedge half-angles greater than this, the shock must move upstream of the wedge (become detached), forming a detached shock, which takes the shape of a bow wave (Chap. 12). The CFD results at $\theta = 30^\circ$ (Fig. 15–80c) show that this is indeed the case. The portion of the detached shock just upstream of the leading edge is a normal shock, and thus the flow downstream of that portion of the shock is subsonic. As the shock number increases, it becomes progressively weaker, and the Mach number downstream of the shock increases, as indicated by the lighter shade of gray.

**15–6 = OPEN-CHANNEL FLOW CFD CALCULATIONS**

So far, all our examples have been for one single-phase fluid (air or water). However, many commercially available CFD codes can handle flow of a mixture of gases (e.g., carbon monoxide in air), flow with two phases of the same fluid (e.g., steam and liquid water), and even flow of two fluids of different phase (e.g., liquid water and gaseous air). The latter case is of interest in this section, namely, the flow of water with a free surface, above which is gaseous air, i.e., open-channel flow. We present here some simple examples of CFD solutions of open-channel flows.
Flow over a Bump on the Bottom of a Channel

Consider a two-dimensional channel with a flat, horizontal bottom. At a certain location along the bottom of the channel, there is a smooth bump, 1.0 m long and 0.10 m high at its center (Fig. 15–81). The velocity inlet is split into two parts—the lower part for liquid water and the upper part for air. In the CFD calculations, the inlet velocity of both the air and the water is specified as $V_{\text{inlet}}$. The water depth at the inlet of the computational domain is specified as $y_{\text{inlet}}$, but the location of the water surface in the rest of the domain is calculated. The flow is modeled as inviscid.

We consider cases with both subcritical and supercritical inlets (Chap. 13). Results from the CFD calculations are shown in Fig. 15–82 for three cases for comparison. For the first case (Fig. 15–82a), $y_{\text{inlet}}$ is specified as 0.30 m, and $V_{\text{inlet}}$ is specified as 0.50 m/s. The corresponding Froude number is calculated to be $F_{\text{r}} = \sqrt{\frac{g y_{\text{inlet}}}{V_{\text{inlet}}^2}} = \sqrt{\frac{(9.81 \text{ m/s}^2)(0.30 \text{ m})}{0.50 \text{ m/s}^2}} = 0.291$.

Since $F_r < 1$, the flow at the inlet is subcritical, and the liquid surface dips slightly above the bump (Fig. 15–82a). The flow remains subcritical downstream of the bump, and the liquid surface height slowly rises back to its prebump level. The flow is thus subcritical everywhere.

For the second case (Fig. 15–82b), $y_{\text{inlet}}$ is specified as 0.50 m, and $V_{\text{inlet}}$ is specified as 4.0 m/s. The corresponding Froude number is calculated to be $F_{\text{r}} = 1.81$. Since $F_r > 1$, the flow at the inlet is supercritical, and the liquid surface rises above the bump (Fig. 15–82b). Far downstream, the liquid depth returns to 0.50 m, and the average velocity returns to 4.0 m/s, yielding $F_r = 1.81$—the same as at the inlet. Thus, this flow is supercritical everywhere.

Finally, we show results for a third case (Fig. 15–82c) in which the flow entering the channel is subcritical ($y_{\text{inlet}} = 0.50$ m, $V_{\text{inlet}} = 1.0$ m/s, and $F_r = 0.452$). In this case, the water surface dips downward over the bump, as expected for subcritical flow. However, on the downstream side of the bump, $y_{\text{outlet}} = 0.25$ m, $V_{\text{outlet}} = 2.0$ m/s, and $F_r = 1.28$. Thus, this flow starts subcritical, but changes to supercritical downstream of the bump. If the domain had extended further downstream, we would likely see a hydraulic jump that would bring the Froude number back below unity (subcritical).
Flow through a Sluice Gate (Hydraulic Jump)

As a final example, we consider a two-dimensional channel with a flat, horizontal bottom, but this time with a sluice gate (Fig. 15–83). The water depth at the inlet of the computational domain is specified as $y_{\text{inlet}}$, and the inlet flow velocity is specified as $V_{\text{inlet}}$. The bottom of the sluice gate is a distance $a$ from the channel bottom. The flow is modeled as inviscid.

We run the CFD code with $y_{\text{inlet}} = 12.0$ m and $V_{\text{inlet}} = 0.833$ m/s, yielding an inlet Froude number of $Fr_{\text{inlet}} = 0.0768$ (subcritical). The bottom of the sluice gate is at $a = 0.125$ m from the channel bottom. Results from the CFD calculations are shown in Fig. 15–84. After the water passes under the sluice gate, its average velocity increases to $12.8$ m/s, and its depth decreases to $y = 0.78$ m. Thus, $Fr = 4.63$ (supercritical) downstream of the sluice gate and upstream of the hydraulic jump. Some distance downstream, we see a hydraulic jump in which the average water depth increases to $y = 3.54$ m, and the average water velocity decreases to $2.82$ m/s. The Froude number downstream of the hydraulic jump is thus $Fr = 0.478$ (subcritical).

We notice that the downstream water depth is significantly lower than that upstream of the sluice gate, indicating relatively large dissipation through the hydraulic jump and a corresponding decrease in the specific energy of the flow (Chap. 13). The analogy between specific energy loss through a hydraulic jump in open-channel flow and stagnation pressure loss through a shock wave in compressible flow is reinforced.

**FIGURE 15–83**

Computational domain for steady, incompressible, two-dimensional flow of water through a sluice gate, with boundary conditions identified. Two fluids are modeled in the flow field—liquid water, and air above the free surface of the water. Liquid depth $y_{\text{inlet}}$ and inlet speed $V_{\text{inlet}}$ are specified.

**FIGURE 15–84**

CFD results for incompressible, two-dimensional flow of water through a sluice gate in an open channel. Phase contours are plotted, where blue indicates liquid water and white indicates gaseous air: (a) overall view of the sluice gate and hydraulic jump, and (b) close-up view of the hydraulic jump. The flow is highly unsteady, and these are instantaneous snapshots at an arbitrary time.
CHAPTER 15

Guest Authors: James G. Brasseur and Anupam Pal, The Pennsylvania State University

The mechanical function of the stomach (called gastric “motility”) is central to proper nutrition, reliable drug delivery, and many gastric dysfunctions such as gastroparesis. Figure 15–85 shows a magnetic resonance image (MRI) of the stomach. The stomach is a mixer, a grinder, a storage chamber, and a sophisticated pump that controls the release of liquid and solid gastric content into the small intestines where nutrient uptake occurs. Nutrient release is controlled by the opening and closing of a valve at the end of the stomach (the pylorus) and the time variations in pressure difference between the stomach and duodenum. Gastric pressure is controlled by muscle tension over the stomach wall and peristaltic contraction waves that pass through the antrum (Fig. 15–85). These antral peristaltic contraction waves also break down food particles and mix material within the stomach, both food and drugs. It is currently impossible, however, to measure the mixing fluid motions in the human stomach. The MRI, for example, gives only an outline of special magnetized fluid within the stomach. In order to study these invisible fluid motions and their effects, we have developed a computer model of the stomach using computational fluid dynamics.

The mathematics underlying our computational model is derived from the laws of fluid mechanics. The model is a way of extending MRI measurements of time-evolving stomach geometry to the fluid motions within. Whereas computer models cannot describe the full complexity of gastric physiology, they have the great advantage of allowing controlled systematic variation of parameters, so sensitivities that cannot be measured experimentally can be studied computationally. Our virtual stomach applies a numerical method called the “lattice Boltzmann” algorithm that is well suited to fluid flows in complex geometries, and the boundary conditions are obtained from MRI data. In Fig. 15–86 we predict the motions, breakdown, and mixing of 1-cm-size extended-release drug tablets in the stomach. In this numerical experiment the drug tablet is denser than the surrounding highly viscous meal. We predict that the antral peristaltic waves generate recirculating eddies and retropulsive “jets” within the stomach, which in turn generate high shear stresses that wear away the tablet surface and release the drug. The drug then mixes by the same fluid motions that release the drug. We find that gastric fluid motions and mixing depend on the details of the time variations in stomach geometry and pylorus.

References

APPLICATION SPOTLIGHT ■ A Virtual Stomach

Figure 15–85
Magnetic resonance image of the human stomach in vivo at one instant in time showing peristaltic (i.e., propagating) contraction waves (CW) in the end region of the stomach (the antrum). The pylorus is a sphincter, or valve, that allows nutrients into the duodenum (small intestines).

Figure 15–86
Computer simulation of fluid motions within the stomach (velocity vectors) from peristaltic antral contraction waves (Fig. 15–85), and the release of a drug (blue trail) from an extended release tablet (blue circle).
SUMMARY

Although neither as ubiquitous as spreadsheets, nor as easy to use as mathematical solvers, computational fluid dynamics codes are continually improving and are becoming more commonplace. Once the realm of specialized scientists who wrote their own codes and used supercomputers, commercial CFD codes with numerous features and user-friendly interfaces can now be obtained for personal computers at a reasonable cost and are available to engineers of all disciplines. As shown in this chapter, however, a poor grid, improper choice of laminar versus turbulent flow, inappropriate boundary conditions, and/or any of a number of other miscues can lead to CFD solutions that are physically incorrect, even though the colorful graphical output always looks pretty. Therefore, it is imperative that CFD users be well grounded in the fundamentals of fluid mechanics in order to avoid erroneous answers from a CFD simulation. In addition, appropriate comparisons should be made to experimental data whenever possible to validate CFD predictions. Bearing these cautions in mind, CFD has enormous potential for diverse applications involving fluid flows.

We show examples of both laminar and turbulent CFD solutions. For incompressible laminar flow, computational fluid dynamics does an excellent job, even for unsteady flows with separation. In fact, laminar CFD solutions are “exact” to the extent that they are limited by grid resolution and boundary conditions. Unfortunately, many flows of practical engineering interest are turbulent, not laminar. Direct numerical simulation (DNS) has great potential for simulation of complex turbulent flow fields, and algorithms for solving the equations of motion (the three-dimensional continuity and Navier–Stokes equations) are well established. However, resolution of all the fine scales of a high Reynolds number complex turbulent flow requires computers that are orders of magnitude faster than today’s fastest machines. It will be decades before computers advance to the point where DNS is useful for practical engineering problems. In the meantime, the best we can do is to employ turbulence models, which are semi-empirical transport equations that model (rather than solve) the increased mixing and diffusion caused by turbulent eddies. When running CFD codes that utilize turbulence models, we must be careful that we have a fine-enough mesh and that all boundary conditions are properly applied. In the end, however, regardless of how fine the mesh, or how valid the boundary conditions, turbulent CFD results are only as good as the turbulence model used. Nevertheless, while no turbulence model is universal (applicable to all turbulent flows), we obtain reasonable performance for many practical flow simulations.

We also demonstrate in this chapter that CFD can yield useful results for flows with heat transfer, compressible flows, and open-channel flows. In all cases, however, users of CFD must be careful that they choose an appropriate computational domain, apply proper boundary conditions, generate a good grid, and use the proper models and approximations. As computers continue to become faster and more powerful, CFD will take on an ever-increasing role in design and analysis of complex engineering systems.

We have only scratched the surface of computational fluid dynamics in this brief chapter. In order to become proficient and competent at CFD, you must take advanced courses of study in numerical methods, fluid mechanics, turbulence, and heat transfer. We hope that, if nothing else, this chapter has spurred you on to further study of this exciting topic.

REFERENCES AND SUGGESTED READING

PROBLEMS*

Fundamentals, Grid Generation, and Boundary Conditions

15–1C A CFD code is used to solve a two-dimensional (x and y), incompressible, laminar flow without free surfaces. The fluid is Newtonian. Appropriate boundary conditions are used. List the variables (unknowns) in the problem, and list the corresponding equations to be solved by the computer.

15–2C Write a brief (a few sentences) definition and description of each of the following, and provide example(s) if helpful: (a) computational domain, (b) mesh, (c) transport equation, (d) coupled equations.

15–3C What is the difference between a node and an interval and how are they related to cells? In Fig. P15–3C, how many nodes and how many intervals are on each edge?

15–4C For the two-dimensional computational domain of Fig. P15–3C, with the given node distribution, sketch a simple structured grid using four-sided cells and sketch a simple unstructured grid using three-sided cells. How many cells are in each? Discuss.

15–5C Summarize the eight steps involved in a typical CFD analysis of a steady, laminar flow field.

15–6C Suppose you are using CFD to simulate flow through a duct in which there is a circular cylinder as in Fig. P15–6C. The duct is long, but to save computer resources you choose a computational domain in the vicinity of the cylinder only. Explain why the downstream edge of the computational domain should be further from the cylinder than the upstream edge.

15–7C Write a brief (a few sentences) discussion about the significance of each of the following in regards to an iterative CFD solution: (a) initial conditions, (b) residual, (c) iteration, (d) postprocessing.

15–8C Briefly discuss how each of the following is used by CFD codes to speed up the iteration process: (a) multigrid and (b) artificial time.

15–9C Of the boundary conditions discussed in this chapter, list all the boundary conditions that may be applied to the right edge of the two-dimensional computational domain sketched in Fig. P15–9C. Why can't the other boundary conditions be applied to this edge?

15–10C What is the standard method to test for adequate grid resolution when using CFD?

15–11C What is the difference between a pressure inlet and a velocity inlet boundary condition? Explain why you cannot specify both pressure and velocity at a velocity inlet boundary condition or at a pressure inlet boundary condition.

15–12C An incompressible CFD code is used to simulate the flow of air through a two-dimensional rectangular channel (Fig. P15–12C). The computational domain consists of four blocks, as indicated. Flow enters block 4 from the upper right and exits block 1 to the left as shown. Inlet velocity \( V \) is known and outlet pressure \( P_{\text{out}} \) is also known. Label the boundary conditions that should be applied to every edge of every block of this computational domain.

* Problems designated by a “C” are concept questions, and students are encouraged to answer them all. Problems designated by an “E” are in English units, and the SI users can ignore them.
15–13C Consider Prob. 15–12C again, except let the boundary condition on the common edge between blocks 1 and 2 be a fan with a specified pressure rise from right to left across the fan. Suppose an incompressible CFD code is run for both cases (with and without the fan). All else being equal, will the pressure at the inlet increase or decrease? Why? What will happen to the velocity at the outlet? Explain.

15–14C List six boundary conditions that are used with CFD to solve incompressible fluid flow problems. For each one, provide a brief description and give an example of how that boundary condition is used.

15–15 A CFD code is used to simulate flow over a two-dimensional airfoil at an angle of attack. A portion of the computational domain near the airfoil is outlined in Fig. P15–15 (the computational domain extends well beyond the region outlined by the dashed line). Sketch a coarse structured grid using four-sided cells and sketch a coarse unstructured grid using three-sided cells in the region shown. Be sure to cluster the cells where appropriate. Discuss the advantages and disadvantages of each grid type.

15–16 For the airfoil of Prob. 15–15, sketch a coarse hybrid grid and explain the advantages of such a grid.

15–17 An incompressible CFD code is used to simulate the flow of water through a two-dimensional rectangular channel in which there is a circular cylinder (Fig. P15–17). Flow enters from the left and exits to the right as shown. Inlet velocity $V$ is known, and outlet pressure $P_{out}$ is also known. Generate the blocking for a structured grid using four-sided blocks, and sketch a coarse grid using four-sided cells, being sure to cluster cells near walls. Also be careful to avoid highly skewed cells. Label the boundary conditions that should be applied to every edge of every block of your computational domain. (Hint: Six to seven blocks are sufficient.)

15–18 An incompressible CFD code is used to simulate the flow of gasoline through a two-dimensional rectangular channel in which there is a large circular settling chamber (Fig. P15–18). Flow enters from the left and exits to the right as shown. A time-averaged turbulent flow solution is generated using a turbulence model. Top–bottom symmetry is assumed. Inlet velocity $V$ is known, and outlet pressure $P_{out}$ is also known. Generate the blocking for a structured grid using four-sided blocks, and sketch a coarse grid using four-sided cells, being sure to cluster cells near walls. Also be careful to avoid highly skewed cells. Label the boundary conditions that should be applied to every edge of every block of your computational domain.

15–19 Redraw the structured multiblock grid of Fig. 15–12b for the case in which your CFD code can handle only elementary blocks. Renumber all the blocks and indicate how many i- and j-intervals are contained in each block. How many elementary blocks do you end up with? Add up all the cells, and verify that the total number of cells does not change.

15–20 Suppose your CFD code can handle nonelementary blocks. Combine as many blocks of Fig. 15–12b as you can. The only restriction is that in any one block, the number of i-intervals and the number of j-intervals must be constants. Show that you can create a structured grid with only three nonelementary blocks. Renumber all the blocks and indicate how many i- and j-intervals are contained in each block. Add up all the cells and verify that the total number of cells does not change.

15–21 A new heat exchanger is being designed with the goal of mixing the fluid downstream of each stage as thoroughly as possible. Anita comes up with a design whose cross section for one stage is sketched in Fig. P15–21. The geometry extends periodically up and down beyond the region shown here. She uses several dozen rectangular tubes inclined at a high angle of attack to ensure that the flow separates and
mixes in the wakes. The performance of this geometry is to be tested using two-dimensional time-averaged CFD simulations with a turbulence model, and the results will be compared to those of competing geometries. Sketch the simplest possible computational domain that can be used to simulate this flow. Label and indicate all boundary conditions on your diagram. Discuss.

15–22 Sketch a coarse structured multiblock grid with four-sided elementary blocks and four-sided cells for the computational domain of Prob. 15–21.

15–23 Anita runs a CFD code using the computational domain and grid developed in Probs. 15–21 and 15–22. Unfortunately, the CFD code has a difficult time converging and Anita realizes that there is reverse flow at the outlet (far right edge of the computational domain). Explain why there is reverse flow, and discuss what Anita should do to correct the problem.

15–24 As a follow-up to the heat exchanger design of Prob. 15–21, suppose Anita’s design is chosen based on the results of a preliminary single-stage CFD analysis. Now she is asked to simulate two stages of the heat exchanger. The second row of rectangular tubes is staggered and inclined oppositely to that of the first row to promote mixing (Fig. P15–24). The geometry extends periodically up and down beyond the region shown here. Sketch a computational domain that can be used to simulate this flow. Label and indicate all boundary conditions on your diagram. Discuss.

15–25 Sketch a structured multiblock grid with four-sided elementary blocks for the computational domain of Prob. 15–24. Each block is to have four-sided structured cells, but you do not have to sketch the grid, just the block topology. Try to make all the blocks as rectangular as possible to avoid highly skewed cells in the corners. Assume that the CFD code requires that the node distribution on periodic pairs of edges be identical (the two edges of a periodic pair are “linked” in the grid generation process). Also assume that the CFD code does not allow a block’s edges to be split for application of boundary conditions.

FlowLab Problems*

15–26 In this exercise, we examine how far away the boundary of the computational domain needs to be when simulating external flow around a body in a free stream. We choose a two-dimensional case for simplicity—flow at speed $V$ over a rectangular block whose length $L$ is 1.5 times its height $D$ (Fig. P15–26a). We assume the flow to be symmetric about the centerline ($x$-axis), so that we need to model only the upper half of the flow. We set up a semicircular computational domain for the CFD solution, as sketched in Fig. P15–26b. Boundary conditions are shown on all edges. We run several values of outer edge radius $R$ ($5 < R/D < 500$) to determine when the far field boundary is “far enough” away. Run FlowLab, and start template Block_domain.

(a) Calculate the Reynolds number based on the block height $D$. What is the experimentally measured value of the drag coefficient for this two-dimensional block at this Reynolds number (see Chap. 11)?

(b) Generate CFD solutions for various values of $R/D$. For each case, calculate and record drag coefficient $C_D$. Plot $C_D$ as a function of $R/D$. At what value of $R/D$ does $C_D$ become independent of computational extent to three significant digits of precision? Report a final value of $C_D$, and discuss your results.

* These problems require the CFD software program FlowLab, provided with this textbook by FLUENT, Inc. Templates for these problems are available on the book’s website. In each case, a brief statement of the problem is provided here, whereas additional details about the geometry, boundary conditions, and computational parameters are provided within the template.
(c) Discuss some reasons for the discrepancy between the experimental value of $C_D$ and the value obtained here using CFD.

(d) Plot streamlines for two cases: $R/D = 5$ and 500. Compare and discuss.

15–27 Using the geometry of Prob. 15–26, and the case with $R/D = 500$, the goal of this exercise is to check for grid independence. Run FlowLab, and start template Block, mesh. Run various values of grid resolution, and tabulate drag coefficient $C_D$ as a function of the number of cells. Has grid independence been achieved? Report a final value of $C_D$ to three significant digits of precision. Does the final value of drag coefficient agree better with that of this experiment? Discuss.

15–28 In Probs. 15–26 and 15–27, we used air as the fluid in our calculations. In this exercise, we repeat the calculation of drag coefficient, except we use different fluids. We adjust the inlet velocity appropriately such that the calculations are always at the same Reynolds number. Run FlowLab, and start template Block, fluid. Compare the value of $C_D$ for all three cases (air, water, and kerosene) and discuss.

15–29 Experiments on two-dimensional rectangular blocks in an incompressible free-stream flow reveal that the drag coefficient is independent of Reynolds number for Re greater than about $10^4$. In this exercise, we examine if CFD calculations are able to predict the same independence of $C_D$ on Re. Run FlowLab, and start template Block, Reynolds. Calculate and record $C_D$ for several values of Re. Discuss.

15–30 In Probs. 15–26 through 15–29, the k-ε turbulence model is used. The goal of this exercise is to see how sensitive the drag coefficient is to our choice of turbulence model and to see if a different turbulence model yields better agreement with experiment. Run FlowLab, and start template Block, turbulence_model. Run the simulation with all the available turbulence models. For each case, record $C_D$. Which one gives the best agreement with experiment? Discuss.

15–31 Experimental drag coefficient data are available for two-dimensional blocks of various shapes in external flow. In this exercise, we use CFD to compare the drag coefficient of rectangular blocks with $L/D$ ranging from 0.1 to 3.0 (Fig. P15–31). The computational domain is a semicircle similar to that sketched in Fig. P15–26b; we assume steady, incompressible, turbulent flow with symmetry about the x-axis. Run FlowLab, and start template Block, length.

(a) Run the CFD simulation for various values of $L/D$ between 0.1 and 3.0. Record the drag coefficient for each case, and plot $C_D$ as a function of $L/D$. Compare to experimentally obtained data on the same plot. Discuss.

(b) For each case, plot streamlines near the block and in its wake region. Use these streamlines to help explain the trend in the plot of $C_D$ versus $L/D$.

(c) Discuss possible reasons for the discrepancy between CFD calculations and experimental data and suggest a remedy.

15–32 Repeat Prob. 15–26 for the case of axisymmetric flow over a blunt-faced cylinder (Fig. P15–32), using FlowLab
template Block_axisymmetric. The grids and all the parameters are the same as those in Prob. 15–26, except the symmetry boundary condition is changed to “axis,” and the flow solver is axisymmetric about the x-axis. In addition to the questions listed in Prob. 15–26, compare the two-dimensional and axisymmetric cases. Which one requires a greater extent of the far field boundary? Which one has better agreement with experiment? Discuss. (Note: The reference area for the far field boundary? Which one has better agreement with axisymmetric cases. Which one requires a greater extent of the computational domain? The overall length of the computational domain is 9.50 m in all cases.)

15-33 Air flows through a conical diffuser in an axisymmetric wind tunnel (Fig. P15–33a—drawing not to scale). \( \theta \) is the diffuser half-angle (the total angle of the diffuser is equal to \( 2\theta \)). The inlet and outlet diameters are \( D_1 = 0.50 \) m and \( D_2 = 1.0 \) m, respectively, and \( \theta = 20^\circ \). The inlet velocity is nearly uniform at \( V = 10.0 \) m/s. The axial distance upstream of the diffuser is \( L_1 = 1.50 \) m, and the axial distance from the start of the diffuser to the outlet is \( L_2 = 8.00 \) m. We set up a computational domain for a CFD solution, as sketched in Fig. P15–33b. Since the flow is axisymmetric and steady in the mean, we model only one two-dimensional slice as shown, with the bottom edge of the domain specified as an axis. The goal of this exercise is to test for grid independence. Run FlowLab, and start template Diffuser_mesh.

(a) Generate CFD solutions for several grid resolutions. Plot streamlines in the diffuser section for each case. At what grid resolution does the streamline pattern appear to be grid independent? Describe the flow field for each case and discuss.

(b) For each case, calculate and record pressure difference \( \Delta P = P_{in} - P_{out} \). At what grid resolution is the \( \Delta P \) grid independent (to three significant digits of precision)? Plot \( \Delta P \) as a function of number of cells. Discuss your results.

15-34 Repeat Prob. 15–33 for the finest resolution case, but with the “pressure outlet” boundary condition changed to an “outflow” boundary condition instead, using FlowLab template Diffuser_outflow. Record \( \Delta P \) and compare with the result of Prob. 15–33 for the same grid resolution. Also compare the pressure distribution at the outlet for the case with the pressure outlet boundary condition and the case with the outflow boundary condition. Discuss.

15-35 Barbara is designing a conical diffuser for the axisymmetric wind tunnel of Prob. 15–33. She needs to achieve at least 40 Pa of pressure recovery through the diffuser, while keeping the diffuser length as small as possible. Barbara decides to use CFD to compare the performance of diffusers of various half-angles (\( 5^\circ \leq \theta \leq 90^\circ \)) (see Fig. P15–33 for the definition of \( \theta \) and other parameters in the problem). In all cases, the diameter doubles through the diffuser—the inlet and outlet diameters are \( D_1 = 0.50 \) m and \( D_2 = 1.0 \) m, respectively. The inlet velocity is nearly uniform at \( V = 10.0 \) m/s. The axial distance upstream of the diffuser is \( L_1 = 1.50 \) m, and the axial distance from the start of the diffuser to the outlet is \( L_2 = 8.00 \) m. (The overall length of the computational domain is 9.50 m in all cases.)

Run FlowLab with template Diffuser_angle. In addition to the axis and wall boundary conditions labeled in Fig. P15–33, the inlet is specified as a velocity inlet and the outlet is specified as a pressure outlet with \( P_{out} = 0 \) gage pressure for all cases. The fluid is air at default conditions, and turbulent flow is assumed.

(a) Generate CFD solutions for half-angle \( \theta = 5, 7.5, 10, 12.5, 15, 17.5, 20, 25, 30, 45, 60, \) and \( 90^\circ \). Plot streamlines for each case. Describe how the flow field changes with the diffuser half-angle, paying particular attention to flow separation on the diffuser wall. How small must \( \theta \) be to avoid flow separation?

(b) For each case, calculate and record \( \Delta P = P_{in} - P_{out} \). Plot \( \Delta P \) as a function of \( \theta \) and discuss your results. What is the maximum value of \( \theta \) that achieves Barbara’s design objectives?

15-36 Consider the diffuser of Prob. 15-35 with \( \theta = 90^\circ \) (sudden expansion). In this exercise, we test whether the grid is fine enough by performing a grid independence check. Run FlowLab, and start template Expansion_mesh. Run the CFD code for several levels of grid refinement. Calculate and record \( \Delta P \) for each case. Discuss.

15-37 Water flows through a sudden contraction in a small round tube (Fig. P15–37a). The tube diameters are \( D_1 = 8.0 \) mm and \( D_2 = 2.0 \) mm. The inlet velocity is nearly uniform at \( V = 0.050 \) m/s, and the flow is laminar. Shane wants to predict the pressure difference from the inlet (\( x = -L_2 \)) to
Contraction_pressure

Since the flow is axisymmetric and steady, Shane models only one slice, as shown, with the bottom edge of the domain specified as an axis. In addition to the boundary conditions labeled in Fig. P15–37b, the inlet is specified as a velocity inlet, and the outlet is specified as a pressure outlet with $P_{\text{out}} = 0$ gage. What Shane does not know is how far he needs to extend the domain downstream of the contraction in order for the flow field to be simulated accurately upstream of the contraction. (He has no interest in the flow downstream of the contraction.) In other words, he does not know how long to make $L_{\text{extend}}$. Run FlowLab, and start template Contraction_domain.

(a) Generate solutions for $L_{\text{extend}}/D_2 = 0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 2.0, 2.5$, and $3.0$. How big must $L_{\text{extend}}/D_2$ be in order to avoid reverse flow at the pressure outlet? Explain. Plot streamlines near the sudden contraction to help explain your results.

(b) For each case, record gage pressures $P_{\text{in}}$ and $P_1$, and calculate $\Delta P = P_{\text{in}} - P_1$. How big must $L_{\text{extend}}/D_2$ be in order for $\Delta P$ to become independent of $L_{\text{extend}}$ (to three significant digits of precision)?

(c) Plot inlet gage pressure $P_{\text{in}}$ as a function of $L_{\text{extend}}/D_2$. Discuss and explain the trend. Based on all your results taken collectively, which value of $L_{\text{extend}}/D_2$ would you recommend to Shane?

15-38 Consider the sudden contraction of Prob. 15-37 (Fig. P15-37). Suppose Shane were to disregard the downstream extension entirely ($L_{\text{extend}}/D_2 = 0$). Run FlowLab, and start template Contraction_zerolength. Iterate to convergence. Is there reverse flow? Explain. Plot streamlines near the outlet, and compare with those of Prob. 15-37. Discuss. Calculate $\Delta P = P_{\text{in}} - P_{\text{out}}$ and calculate the percentage error in $\Delta P$ under these conditions, compared to the converged value of Prob. 15-37. Discuss.

15-39 In this exercise, we apply different back pressures to the sudden contraction of Prob. 15-37 (Fig. P15-37), for the case with $L_{\text{extend}}/D_2 = 2.0$. Run FlowLab, and start template Contraction_pressure. Set the pressure boundary condition at the outlet to $P_{\text{out}} = -50,000$ Pa gage (about 1/2 atm below atmospheric pressure). Record $P_{\text{in}}$ and $P_1$, and calculate $\Delta P = P_{\text{in}} - P_1$. Repeat for $P_{\text{out}} = 0$ Pa gage and $P_{\text{out}} = 50,000$ Pa gage. Discuss your results.

15-40 Consider the sudden contraction of Prob. 15-37, but this time with turbulent rather than laminar flow. The dimensions shown in Fig. P15-37 are scaled proportionally by a factor of 100 everywhere so that $D_1 = 0.80$ m and $D_2 = 0.20$ m. The inlet velocity is also increased to $V = 1.0$ m/s. A 10 percent turbulence intensity is specified at the inlet. The outlet pressure is fixed at zero gage pressure for all cases. Run FlowLab, and start template Contraction_turbulent.

(a) Calculate the Reynolds numbers of flow through the large tube and the small tube for Prob. 15-37 and also for this problem. Are our assumptions of laminar versus turbulent flow reasonable for these problems?

(b) Generate CFD solutions for $L_{\text{extend}}/D_2 = 0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 2.0$. How big must $L_{\text{extend}}/D_2$ be in order to avoid reverse flow at the pressure outlet? Plot streamlines for the case in which $L_{\text{extend}}/D_2 = 0.75$ and compare to the corresponding streamlines of Prob. 15-37 (laminar flow). Discuss.

(c) For each case, record gage pressures $P_{\text{in}}$ and $P_1$, and calculate $\Delta P = P_{\text{in}} - P_1$. How big must $L_{\text{extend}}/D_2$ be in order for $\Delta P$ to become independent of $L_{\text{extend}}$ (to three significant digits of precision)?

15-41 Run FlowLab, and start template Contraction_outflow. The conditions are identical to Prob. 15-40 for the case with $L_{\text{extend}}/D_2 = 0.75$, but with the “pressure outlet” boundary condition changed to an “outflow” boundary condition instead. Record $P_{\text{in}}$ and $P_1$, calculate $\Delta P = P_{\text{in}} - P_1$, and compare with the result of Prob. 15-40 for the same geometry. Discuss.

15-42 Run FlowLab with template Contraction_2d. This is identical to the sudden contraction of Prob. 15-40, but the flow is two-dimensional instead of axisymmetric. (Note that the “axis” boundary condition is replaced by “symmetry.”) As previously, the outlet pressure is set to zero gage pressure.

(a) Generate CFD solutions for $L_{\text{extend}}/D_2 = 0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 2.0, 3.0$, and $4.0$. How big must $L_{\text{extend}}/D_2$ be in order to avoid reverse flow in the outlet? Plot streamlines near the outlet, and explain your results.

For each case, record gage pressures $P_{\text{in}}$ and $P_1$, and calculate $\Delta P = P_{\text{in}} - P_1$. How big must $L_{\text{extend}}/D_2$ be in order for $\Delta P$ to become independent of $L_{\text{extend}}$ (to three significant digits of precision)?
order to avoid reverse flow at the pressure outlet? Plot streamlines for the case in which \( L_{\text{extend}}/D_1 = 0.75 \), and compare to the corresponding streamlines of Prob. 15–40 (axisymmetric flow). Discuss.

(b) For each case, record gage pressures \( P_{\text{in}} \) and \( P_1 \), and calculate \( \Delta P = P_{\text{in}} - P_1 \). How big must \( L_{\text{extend}}/D_1 \) be in order for \( \Delta P \) to become independent of \( L_{\text{extend}} \) (to three significant digits of precision)?

15–43 Air flows through a “jog” in a rectangular channel (Fig. P15–43a, not to scale). The channel dimension is \( D_1 = 1.0 \text{ m} \) everywhere, and it is wide enough (into the page of Fig. P15–43) that the flow can be considered two-dimensional. The inlet velocity is nearly uniform at \( V = 1.0 \text{ m/s} \). The distance upstream of the jog is \( L_1 = 5.0 \text{ m} \), the overall jog length is \( L_2 = 3.0 \text{ m} \), and the distance from the end of the jog to the outlet is \( L_3 = 10.0 \text{ m} \). We set up a computational domain for a CFD solution, as sketched in Fig. P15–43b. In addition to the wall boundary conditions labeled in Fig. P15–43b, the inlet is specified as a velocity inlet and the outlet is specified as a pressure outlet with \( P_\text{out} = 0 \) gage pressure. The fluid is air at default conditions, and turbulent flow is assumed. The goal of this exercise is to test for grid independence in this flow field. Run FlowLab with template Jog_laminar_mesh.

(a) Generate CFD solutions for various levels of grid resolution. Plot streamlines in the region of the jog for each case. At what grid resolution does the streamline pattern appear to be grid independent? Discuss.

(b) For each case, calculate and record \( \Delta P = P_{\text{in}} - P_\text{out} \). At what grid resolution is \( \Delta P \) grid independent (to three significant digits of precision)? Plot \( \Delta P \) as a function of the number of cells. Discuss your results.

15–44 Repeat Prob. 15–43, but for laminar flow, using Jog_laminar_mesh as the FlowLab template. The jog is identical in shape, but scaled down by a factor of 1000 compared to that of Prob. 15–43 (the channel width is \( D_1 = 1.0 \text{ mm} \) everywhere). The inlet velocity is nearly uniform at \( V = 0.10 \text{ m/s} \), and the fluid is changed to water at room temperature. Discuss your results.

15–45 Repeat Prob. 15–44, but for laminar flow at a higher Reynolds number, using FlowLab template Jog_high_Re. Everything is identical to Prob. 15–44, except the inlet velocity is increased from to \( V = 0.10 \text{ to } 1.0 \text{ m/s} \). Compare the Reynolds numbers for the two cases and discuss.

15–46 Consider compressible flow of air through an axisymmetric converging-diverging nozzle (Fig. P15–46), in which the inviscid flow approximation is applied. The inlet conditions are fixed (\( P_{\text{inlet}} = 220 \text{ kPa} \), \( P_{\text{inlet}} = 210 \text{ kPa} \), and \( T_{\text{inlet}} = 300 \text{ K} \)), but the back pressure \( P_b \) can be varied. Run FlowLab, using template Nozzle_axisymmetric. Do several cases, with back pressure ranging from 100 to 219 kPa. For each case, calculate the mass flow rate (kg/s) through the nozzle, and plot \( m \) as a function of \( P_b/P_{\text{inlet}} \). Explain your results.

15–47 Run FlowLab with template Nozzle_axisymmetric (Prob. 15–46). For the case in which \( P_b = 100 \text{ kPa} \) (\( P_{\text{inlet}}/P_{\text{inlet}} = 0.455 \)), plot pressure and the Mach number contours to verify that a normal shock is present near the outlet of the computational domain. Generate a plot of average Mach number \( M_a \) and average pressure ratio \( P_{\text{out}}/P_{\text{inlet}} \) across several cross sections of the domain, as in Fig. 15–76. Point out the location of the normal shock, and compare the CFD results to one-dimensional compressible flow theory. Repeat for \( P_b = 215 \text{ kPa} \) (\( P_{\text{inlet}}/P_{\text{inlet}} = 0.977 \)). Explain.

15–48 Run FlowLab with template Nozzle_2d, which is the same as Prob. 15–46, except the flow is two-dimensional instead of axisymmetric. Note that the “axis” boundary condition is also changed to “symmetry.” Compare your results and discuss the similarities and differences.

15–49 Consider flow over a simplified, two-dimensional model of an automobile (Fig. P15–49). The inlet conditions...
are fixed at \( V = 60.0 \text{ mi/h} \) (26.8 m/s), with 10 percent turbulence intensity. The standard \( k-e \) turbulence model is used. Run FlowLab with template Automobile_drag. Vary the shape of the rear end of the car, and record the drag coefficient for each shape. Also plot velocity vectors in the vicinity of the rear end for each case. Compare and discuss. Which case gives the lowest drag coefficient? Why?

Run FlowLab with template Automobile_domain for several values of \( H/h \) (Fig. P15–49). Plot the calculated value of \( C_D \) as a function of \( H/h \). At what value of \( H/h \) does \( C_D \) level off? In other words, how far away must the upper symmetry boundary be in order to have negligible influence on the calculated value of drag coefficient? Discuss.

Run FlowLab, and start template Automobile_turbulence_model. In this exercise, we examine the effect of turbulence model on the calculation of drag on a simplified, two-dimensional model of a car (Fig. P15–49). Run all the available turbulence models. For each case, record \( C_D \). Is there much variation in the calculated values of \( C_D \)? Which one is correct? Discuss.

Run FlowLab, and start template Automobile_3d. In this exercise, we compare the drag coefficient for a fully three-dimensional automobile to that predicted by the two-dimensional approximation of Prob. 15–49. Note that the solution takes a long time to converge and requires a significant amount of computer resources. Therefore, the converged solution is already available in this template. Observe the three-dimensional pathlines around the car by rotating the view. Calculate the drag coefficient. Is it larger or smaller than the two-dimensional prediction? Discuss.

Run FlowLab, and start template Pipe_laminar_developed. In this exercise, we are not concerned about entrance effects. Instead, we want to analyze the fully developed flow downstream of the entrance region. Because of the axisymmetry, the computational domain consists of one slice (light blue region). The velocity profile at the inlet boundary is set to be the same as that at the outlet boundary, but a pressure drop from \( x = 0 \) to \( L \) is imposed to simulate fully developed flow. Run FlowLab with template Pipe_laminar_developed. The template is set up such that the outlet velocity profile gets fed into the inlet. In other words, the inlet and outlet are periodic.

Run FlowLab with template Automobile_3d. In this exercise, we compare the drag coefficient for a fully three-dimensional automobile to that predicted by the two-dimensional approximation of Prob. 15–49. Note that the solution takes a long time to converge and requires a significant amount of computer resources. Therefore, the converged solution is already available in this template. Observe the three-dimensional pathlines around the car by rotating the view. Calculate the drag coefficient. Is it larger or smaller than the two-dimensional prediction? Discuss.

Run FlowLab with template Pipe_laminar_developed. In this exercise, we study laminar flow in the entrance region of a round pipe (Fig. P15–53). Calculate the pressure along the pipe axis for each case. Also plot the pressure distribution along the pipe axis for each case. Estimate the end of the entrance region as the location where the pressure begins to drop linearly with \( x \). Compare your results with those obtained from the velocity profiles, and also with theory, \( L_e/D \approx 0.06Re \). Discuss.
boundary conditions, but with an imposed pressure drop. Run several cases corresponding to various values of the Reynolds number. For each case, look at velocity profiles to confirm that the flow is fully developed. Calculate and plot Darcy friction factor $f$ as a function of $Re$ and compare with the theoretical value for laminar flow, $f = 64/Re$. Discuss the agreement between CFD and theory.

15–56 Repeat Prob. 15–55, except for fully developed turbulent flow through a smooth-walled pipe. Use FlowLab template Pipe_turbulent_developed. Calculate and plot the Darcy friction factor $f$ as a function of $Re$. Compare $f$ with that predicted in Chap. 8 for fully developed turbulent pipe flow through a smooth pipe. Discuss.

15–57 In Prob. 15–56, we considered fully developed turbulent flow through a smooth pipe. In this exercise, we examine fully developed turbulent flow through a rough pipe. Run FlowLab with template Pipe_turbulent_rough. Run several cases, each with a different value of normalized pipe roughness, $\varepsilon/D$, but at the same Reynolds number. Calculate and tabulate Darcy friction factor $f$ as a function of normalized roughness parameter $\varepsilon/D$. Compare $f$ with that predicted by the Colebrook equation for fully developed turbulent pipe flow in rough pipes. Discuss.

15–58 Consider the laminar boundary layer developing over a flat plate (Fig. P15–58). Run FlowLab with template Plate_laminar. The inlet velocity and length are chosen such that the Reynolds number at the end of the plate, $Re = \rho VL/\mu$, is approximately $1 \times 10^5$, just on the verge of transition toward turbulence. From your CFD results, calculate the following, and compare to theory: (a) the boundary layer profile shape at $x = L$ (compare to the Blasius profile), (b) boundary layer thickness $\delta$ as a function of $x$, and (c) drag coefficient on the plate.

15–59 Repeat Prob. 15–58, but for turbulent flow on a smooth flat plate. Use FlowLab template Plate_turbulent. The Reynolds number at the end of the plate is approximately $1 \times 10^7$ for this case—well beyond the transition region.

15–60 Run FlowLab, and start template Plate_turbulence_models. In this exercise, we examine the effect of the turbulence model on the calculation of the drag coefficient on a flat plate (Fig. P15–58). Run for each of the available turbulence models. For each case, record $C_D$. Is there much variation in the calculated values of $C_D$? Which turbulence model yields the most correct value of drag coefficient? Discuss.

15–61 Consider laminar flow on a smooth heated flat plate (Fig. P15–61). Run FlowLab template Plate_laminar_temperature for two fluids: air and water. The inlet velocity is adjusted such that the Reynolds number for the air and water cases are approximately equal. Compare the 99 percent temperature thickness at the end of the plate to the 99 percent velocity thickness. Discuss your results. (Hint: What is the Prandtl number of air and of water?)

15–62 Repeat Prob. 15–61, except for turbulent flow on a smooth heated flat plate (Fig. P15–61). Use FlowLab template Plate_turbulent_temperature. Discuss the differences between the laminar and turbulent calculations. Specifically, which regime (laminar or turbulent) produces the largest variation between 99 percent temperature thickness and 99 percent velocity thickness? Explain.

15–63 Consider turbulent flow of water through a smooth, 90°, flanged elbow in a round pipe (Fig. P15–63). Because of symmetry, only half of the pipe is modeled; the center plane is specified as a “symmetry” boundary condition. The pipe walls are smooth. The inlet velocity and pipe diameter are chosen to yield a Reynolds number of 20,000. For the first
(default) case, the standard k-e turbulence model is used. Run FlowLab with template Elbow. This is a three-dimensional calculation, so expect significantly longer run times. The average pressure is calculated across several cross sections of the pipe: upstream of the elbow, in the elbow, and downstream of the elbow (sections A-A, B-B, etc., in Fig. P15–63). Plot average pressure as a function of axial distance along the pipe. Where does most of the pressure drop occur—in the pipe section upstream of the elbow, in the elbow itself, immediately downstream of the elbow, or in the pipe section downstream of the elbow? Discuss.

15–64 Run FlowLab with template Elbow, again using the standard k-e turbulence model. In this exercise, we study velocity vectors in the plane of several cross sections along the pipe. Compare the velocity vectors at a section upstream of the elbow, at a section in the elbow, and at several sections downstream of the elbow. At which locations do you observe counter-rotating eddies? How does the strength of the counter-rotating eddies change with downstream distance? Discuss. Explain why many manufacturers of pipe flowmeters recommend that their flowmeter be installed at least 10 or 20 pipe diameters downstream of an elbow.

15–65 Run FlowLab with template Elbow, again using the standard k-e turbulence model. In this exercise, we calculate the minor loss coefficient for the elbow of Prob. 15–63. In order to do so, we compare the pressure drop calculated through the pipe with the elbow to that through a straight pipe of the same overall length, and with identical inlet and outlet conditions. Calculate the pressure drop from inlet to outlet for both geometries. To calculate the elbow minor loss coefficient, subtract the straight pipe pressure drop from the elbow pressure drop, and compare to the value given in Chap. 8 for a smooth, 90°, flanged elbow.

15–66 In this exercise, we examine the effect of the turbulence model on the calculation of the minor loss coefficient of a pipe elbow (Fig. P15–63). Using FlowLab template Elbow, repeat Prob. 15–65, but with various turbulence models. For each case, calculate K_e. Is there much variation in the calculated values of K_e? Which turbulence model yields the most correct value, compared with the empirical result of Chap. 8? The Spallart-Allmaras model is the simplest, while the Reynolds stress model is the most complicated of the four. Do the calculated results improve with turbulence model complexity? Discuss.

15–67 Consider flow over a two-dimensional airfoil of chord length L_c at an angle of attack \( \alpha \) in a flow of free-stream speed \( V \) with density \( \rho \) and viscosity \( \mu \). Angle \( \alpha \) is measured relative to the free-stream flow direction. (Fig. P15–67). In this exercise, we calculate the nondimensional lift and drag coefficients \( C_L \) and \( C_D \) that correspond to lift and drag forces \( F_L \) and \( F_D \), respectively. Free-stream velocity and chord length are chosen such that the Reynolds number based on \( V \) and \( L_c \) is \( 1 \times 10^7 \) (turbulent boundary layer over nearly the entire airfoil). Run FlowLab with template Airfoil Reynolds number at several values of \( \alpha \), ranging from \(-2\) to \(20^\circ\). For each case, calculate \( C_L \) and \( C_D \). Plot \( C_L \) and \( C_D \) as functions of \( \alpha \). At approximately what angle of attack does this airfoil stall?

15–68 In this problem, we study the effect of Reynolds number on the lift and drag coefficients of an airfoil at various angles of attack. Note that the airfoil used here is of a different shape than that used in Problem 15–67. Run FlowLab with template "Airfoil Reynolds." For the case with Reynolds number equal to \( 3 \times 10^6 \), calculate and plot \( C_L \) and \( C_D \) as functions of \( \alpha \), with \( \alpha \) ranging from \(-2\) to \(24^\circ\). What is the stall angle for this case? Repeat for \( Re = 6 \times 10^6 \). Compare the two results and discuss the effect of Reynolds number on lift and drag of this airfoil.

15–69 In this exercise, we examine the effect of grid resolution on the calculation of airfoil stall (flow separation) for the airfoil of Problem 15–67 at \( \alpha = 15^\circ \) and \( Re = 1 \times 10^7 \). Run FlowLab with template Airfoil mesh. Run for several levels of grid resolution. For each case, calculate \( C_L \) and \( C_D \). How does grid resolution affect the stall angle? Has grid independence been achieved?

15–70 Consider creeping flow produced by the body of a microorganism swimming through water, represented here as...
In general, boundary conditions are shown for each edge in parentheses. The flow is laminar, and the default values of $V$ and $L$ are chosen such that the Reynolds number $Re = \frac{\rho V L}{\mu}$ is equal to 0.20. Run FlowLab with template Creep_domain. Vary the computational domain radius from $R/L = 3$ to 2000. For each case, calculate the drag coefficient $C_D$ on the body. How large of a computational domain is required for the drag coefficient to level off (far field boundary conditions no longer have significant influence)? Discuss. For the largest computational domain case ($R/L = 2000$), plot velocity vectors along a vertical line coincident with the $y$-axis. Compare to the velocity profile we would expect at very high Reynolds numbers. Discuss.

General CFD Problems*

15-71 Run FlowLab with template Creep_Reynolds. In this exercise, the Reynolds number is varied from 0.1 to 100 for flow over an ellipsoid (Fig. P15-70). Plot $C_D$ as a function of $Re$, and compare velocity profiles along the $y$-axis as $Re$ increases above the creeping flow regime. Discuss.

15-72 Consider the two-dimensional wye of Fig. P15-72. Dimensions are in meters, and the drawing is not to scale. Incompressible flow enters from the left, and splits into two parts. Generate three coarse grids, with identical node distributions on all edges of the computational domain: (a) structured multiblock grid, (b) unstructured triangular grid, and (c) unstructured quadrilateral grid. Compare the number of cells in each case and comment about the quality of the grid in each case.

15-73 Choose one of the grids generated in Prob. 15-72, and run a CFD solution for laminar flow of air with a uniform inlet velocity of 0.02 m/s. Set the outlet pressure at both outlets to the same value, and calculate the pressure drop through the wye. Also calculate the percentage of the inlet flow that goes out of each branch. Generate a plot of streamlines. Compare results with those of laminar flow (Prob. 15-73).

15-74 Repeat Prob. 15-73, except for turbulent flow of air with a uniform inlet velocity of 10.0 m/s. In addition, set the turbulence intensity at the inlet to 10 percent with a turbulent length scale of 0.5 m. Use the $k-e$ turbulence model with wall functions. Set the outlet pressure at both outlets to the same value, and calculate the pressure drop through the wye. Also calculate the percentage of the inlet flow that goes out of each branch. Generate a plot of streamlines. Compare results with those of laminar flow (Prob. 15-73).

15-75 Generate a computational domain to study the laminar boundary layer growing on a flat plate at $Re = 10,000$. Generate a very coarse mesh, and then continually refine the mesh until the solution becomes grid independent. Discuss.

15-76 Repeat Prob. 15-75, except for a turbulent boundary layer at $Re = 10^5$. Discuss.

15-77 Generate a computational domain to study ventilation in a room (Fig. P15-77). Specifically, generate a rectangular room with a velocity inlet in the ceiling to model the supply air, and a pressure outlet in the ceiling to model the return air. You may make a two-dimensional approximation for simplicity (the room is infinitely long in the direction normal to the page in Fig. P15-77). Use a structured rectangular grid. Plot streamlines and velocity vectors. Discuss.

15-78 Repeat Prob. 15-77, except use an unstructured triangular grid, keeping everything else the same. Do you get the same results as those of Prob. 15-77? Compare and discuss.

15-79 Repeat Prob. 15-77, except move the supply and/or return vents to various locations in the ceiling. Compare and discuss.

15-80 Choose one of the room geometries of Probs. 15-77 and 15-79, and add the energy equation to the calculations. In particular, model a room with air-conditioning, by specifying the supply air as cool ($T = 18^\circ$C), while the walls, floor, and ceiling are warm ($T = 26^\circ$C). Adjust the supply air speed until the average temperature in the room is as close as possible to $22^\circ$C. How much ventilation (in terms of number of room air volume changes per hour) is required to cool this room to an average temperature of $22^\circ$C? Discuss.

* These problems require CFD software, although not any particular brand. Unlike the FlowLab problems of the previous section, these problems do not have premade templates. Instead, students must do the following problems “from scratch.”
15–81 Repeat Prob. 15–80, except create a three-dimensional room, with an air supply and an air return in the ceiling. Compare the two-dimensional results of Prob. 15–80 with the more realistic three-dimensional results of this problem. Discuss.

15–82 Generate a computational domain to study compressible flow of air through a converging nozzle with atmospheric pressure at the nozzle exit (Fig. P15–82). The nozzle walls may be approximated as inviscid (zero shear stress). Run several cases with various values of inlet pressure. How much inlet pressure is required to choke the flow? What happens if the inlet pressure is higher than this value? Discuss.

**FIGURE P15–82**

15–83 Repeat Prob. 15–82, except remove the inviscid flow approximation. Instead, let the flow be turbulent, with smooth, no-slip walls. Compare your results to those of Prob. 15–82. What is the major effect of friction in this problem? Discuss.

15–84 Generate a computational domain to study incompressible, laminar flow over a two-dimensional streamlined body (Fig. P15–84). Generate various body shapes, and calculate the drag coefficient for each shape. What is the smallest value of \( C_D \) that you can achieve? (Note: For fun, this problem can be turned into a contest between students. Who can generate the lowest-drag body shape?)

**FIGURE P15–84**

15–85 Repeat Prob. 15–84, except for an axisymmetric, rather than a two-dimensional, body. Compare to the two-dimensional case. Which has the lower drag coefficient? Discuss.

15–86 Repeat Prob. 15–85, except for turbulent, rather than laminar, flow. Compare to the laminar case. Which has the lower drag coefficient? Discuss.

15–87 Generate a computational domain to study Mach waves in a two-dimensional supersonic channel (Fig. P15–87). Specifically, the domain should consist of a simple rectangular channel with a supersonic inlet \((M_a = 2.0)\), and with a very small bump on the lower wall. Using air with the inviscid flow approximation, generate a Mach wave, as sketched. Measure the Mach angle, and compare with theory (Chap. 12). Also discuss what happens when the Mach wave hits the opposite wall. Does it disappear, or does it reflect, and if so, what is the reflection angle? Discuss.

**FIGURE P15–87**

15–88 Repeat Prob. 15–87, except for several values of the Mach number, ranging from 1.10 to 3.0. Plot the calculated Mach angle as a function of Mach number and compare to the theoretical Mach angle (Chap. 12). Discuss.

**Review Problems**

15–90C For each statement, choose whether the statement is true or false, and discuss your answer briefly.
(a) The physical validity of a CFD solution always improves as the grid is refined.
(b) The \(x\)-component of the Navier-Stokes equation is an example of a transport equation.
(c) For the same number of nodes in a two-dimensional mesh, a structured grid typically has fewer cells than an unstructured triangular grid.
(d) A time-averaged turbulent flow CFD solution is only as good as the turbulence model used in the calculations.

15–91C In Prob. 15–18 we take advantage of top–bottom symmetry when constructing our computational domain and grid. Why can’t we also take advantage of the right–left symmetry in this exercise? Repeat the discussion for the case of potential flow.

15–91C Gerry creates the computational domain sketched in Fig. P15–91C to simulate flow through a sudden contraction in a two-dimensional duct. He is interested in the time-averaged pressure drop (minor loss coefficient) created by the sudden contraction. Gerry generates a grid and calculates the flow with a CFD code, assuming steady, turbulent, incompressible flow (with a turbulence model).
(a) Discuss one way that Gerry could improve his computational domain and grid so that he would get the same results in approximately half the computer time.

(b) There may be a fundamental flaw in how Gerry has set up his computational domain. What is it? Discuss what should be different about Gerry’s setup.

15–92C Think about modern high-speed, large-memory computer systems. What feature of such computers lends itself nicely to the solution of CFD problems using a multi-block grid with approximately equal numbers of cells in each individual block? Discuss.

15–93C What is the difference between multigrid and multiblocking? Discuss how each may be used to speed up a CFD calculation. Can these two be applied together?

15–94C Suppose you have a fairly complex geometry and a CFD code that can handle unstructured grids with triangular cells. Your grid generation code can create an unstructured grid very quickly. Give some reasons why it might be wiser to take the time to create a multiblock structured grid instead. In other words, is it worth the effort? Discuss.

15–95 Generate a computational domain and grid, and calculate flow through the single-stage heat exchanger of Prob. 15–21, with the heating elements set at a 45° angle of attack with respect to horizontal. Set the inlet air temperature to 20°C, and the wall temperature of the heating elements to 120°C. Calculate the average air temperature at the outlet.

15–96 Repeat the calculations of Prob. 15–95 for several angles of attack of the heating elements, from 0° (horizontal) to 90° (vertical). Use identical inlet conditions and wall conditions for each case. Which angle of attack provides the most heat transfer to the air? Specifically, which angle of attack yields the highest average outlet temperature?

15–97 Generate a computational domain and grid, and calculate flow through the two-stage heat exchanger of Prob. 15–24, with the heating elements of the first stage set at a 45° angle of attack with respect to horizontal, and those of the second stage set to an angle of attack of −45°. Set the inlet air temperature to 20°C, and the wall temperature of the heating elements to 120°C. Calculate the average air temperature at the outlet.

15–98 Repeat the calculations of Prob. 15–97 for several angles of attack of the heating elements, from 0° (horizontal) to 90° (vertical). Use identical inlet conditions and wall conditions for each case. Note that the second stage of heating elements should always be set to an angle of attack that is the negative of that of the first stage. Which angle of attack provides the most heat transfer to the air? Specifically, which angle of attack yields the highest average outlet temperature? Is this the same angle as calculated for the single-stage heat exchanger of Prob. 15–96? Discuss.

15–99 Generate a computational domain and grid, and calculate stationary turbulent flow over a spinning circular cylinder (Fig. P15–99). In which direction is the side force on the body—up or down? Explain. Plot streamlines in the flow. Where is the upstream stagnation point?

15–100 For the spinning cylinder of Fig. P15–99, generate a dimensionless parameter for rotational speed relative to free-stream speed (combine variables \( \omega \), \( D \), and \( V \) into a nondimensional Pi group). Repeat the calculations of Prob. 15–99 for several values of angular velocity \( \omega \). Use identical inlet conditions for each case. Plot lift and drag coefficients as functions of your dimensionless parameter. Discuss.

15–101 Consider the flow of air into a two-dimensional slot along the floor of a large room, where the floor is coincident with the \( x \)-axis (Fig. P15–101). Generate an appropriate computational domain and grid. Using the inviscid flow approximation, calculate vertical velocity component \( y \) as a function of distance away from the slot along the \( y \)-axis. Compare with the potential flow results of Chap. 10 for flow into a line sink. Discuss.

15–102 For the slot flow of Prob. 15–101, change to laminar flow instead of inviscid flow, and recompute the flow field. Compare your results to the inviscid flow case and to the potential flow case of Chap. 10. Plot contours of vorticity.
Where is the irrotational flow approximation appropriate? Discuss.

15-103 Generate a computational domain and grid, and calculate the flow of air into a two-dimensional vacuum cleaner inlet (Fig. P15–103), using the inviscid flow approximation. Compare your results with those predicted in Chap. 10 for potential flow. Discuss.

15-104 For the vacuum cleaner of Prob. 15–103, change to laminar flow instead of inviscid flow, and recompute the flow field. Compare your results to the inviscid flow case and to the potential flow case of Chap. 10. Discuss.