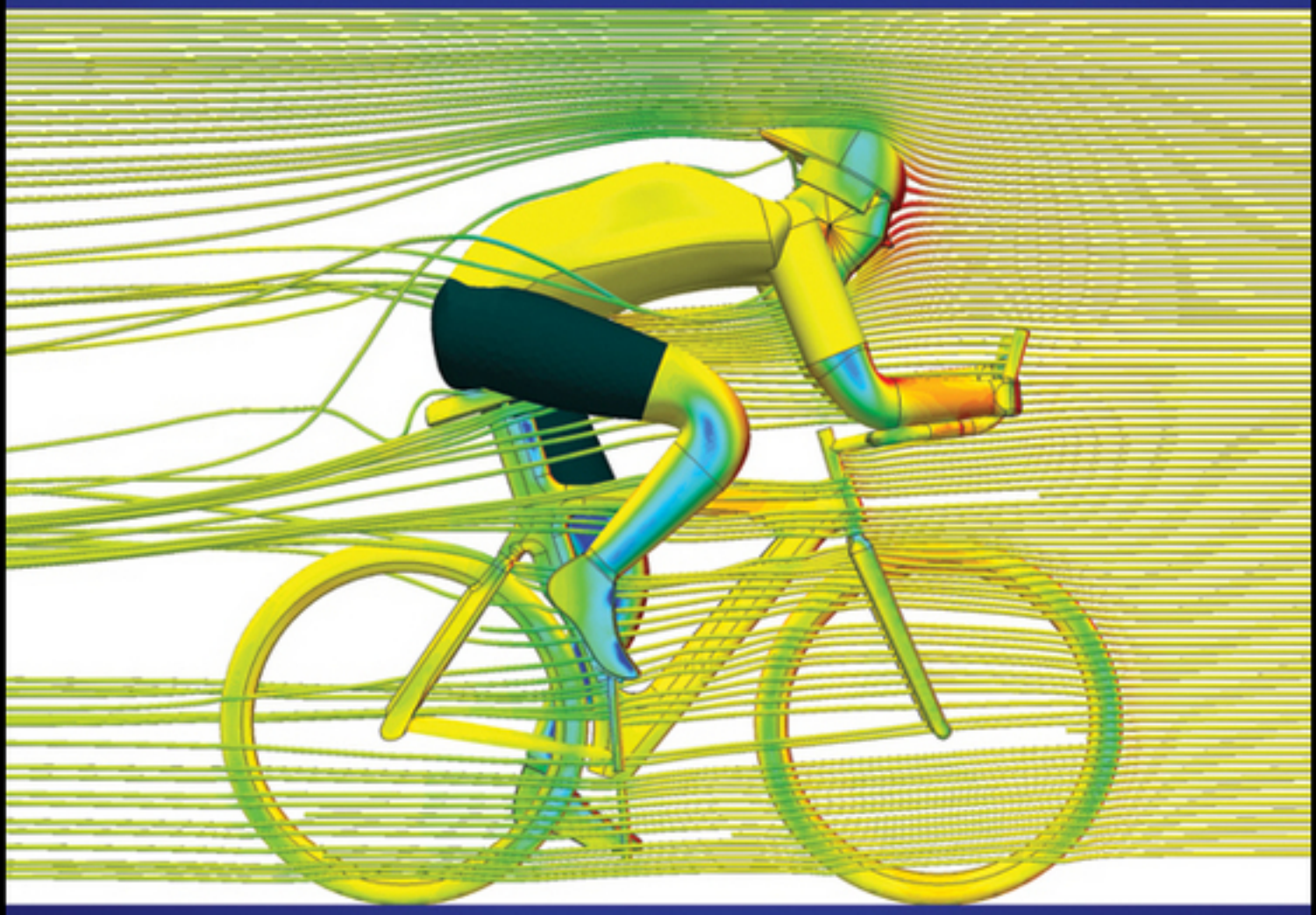
ESSENTIAL COMPUTATIONAL FLUID DYNAMICS



OLEG ZIKANOV

ESSENTIAL COMPUTATIONAL FLUID DYNAMICS

ESSENTIAL COMPUTATIONAL FLUID DYNAMICS

Oleg Zikanov

WILEY

JOHN WILEY & SONS, INC.

This book is printed on acid-free paper. ♾

Published by John Wiley & Sons, Inc., Hoboken, New Jersey

Published simultaneously in Canada

Library of Congress Cataloging-in-Publication Data:
Zikanov, Oleg.
Essential computational fluid dynamics / Oleg Zikanov.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-470-42329-5 (cloth)
1. Fluid dynamics–Mathematics. I. Title.
QA911.Z55 2010
532′.0501515–dc22
2009042596

ISBN: 978-0-470-42329-5

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

To Elena

CONTENTS

PREFACE

This book is a complete and self-contained introduction into computational fluid dynamics and heat transfer, commonly abbreviated as CFD. The text addresses this subject on the very basic level suitable for a first course of CFD taught to beginning graduate or senior undergraduate students. No prior knowledge of CFD is assumed on the part of the reader.

To appreciate the purpose and flavor of the book, we have to consider the major shift that currently occurs in the scope and character of CFD applications. From being a primarily research discipline just 20 years ago, CFD has transformed into a tool of everyday engineering practice. It would be safe to say that, worldwide, tens of thousands of engineers are directly employed to run CFD computations at companies or consulting firms. Many others encounter CFD at some stages of their work.

Unlike solution of research problems, CFD analysis in industrial environment does not, typically, involve development of new algorithms. Instead, one of the general purpose codes is used. Such codes, nowadays, tend to provide a fusion of all the necessary tools: equation solver, mesh generator, turbulence and multiphysics models, and modules for post-processing and parallel computations. Two key factors contribute to the success in applying such codes: (1) Understanding of physical and engineering aspects of the analyzed process; and (2) Ability to conduct the CFD analysis properly, in a way that guarantees an accurate and efficient solution.

I recognized the need for a new textbook when I was teaching the graduate and senior undergraduate courses in CFD at the Department of Mechanical Engineering of the University of Michigan–Dearborn. The majority of our graduate students are either working engineers or researchers in applied engineering fields. The undergraduate students tend to pursue industrial employment after graduation. Potential future exposure of our students to CFD is often limited to the use of general purpose codes. To respond to their needs, the instruction is focused on

two areas: the fundamentals of the method (what we call the *essential CFD*) and the correct way of conducting the analysis using readily available software. A survey of the existing texts on CFD, although revealing many excellent research-oriented texts, does not reveal a book that fully corresponds to this concept.

A comment is in order regarding the bias of the text. All CFD texts are, to some degree, biased in correspondence to the chosen audience and research interests of the authors. More weight is given to some of the methods (finite difference, finite element, spectral, etc.) and to some of the fields of application (heat transfer, incompressible fluid dynamics, or gas dynamics). The choices made in this book reflect the assumption of mechanical, chemical, and civil engineering students as the target audience rather than aerospace engineering students, and the intended use of the text for applied CFD instruction. The focus is on the finite difference and finite volume methods. The finite element and spectral techniques are introduced only briefly. Also, somewhat more attention is given to numerical methods for incompressible fluid dynamics and heat transfer than for compressible flows.

The text can be used in combination with exercises in practical CFD analysis. As an example, our course at the University of Michigan–Dearborn is divided into two parts. The first part (about 60 percent of the total course time) is reserved for classroom instruction of the basic methods of CFD. It covers Part I, "Fundamentals," and Part II, "Methods." It includes a simple programming project (solving a one-dimensional heat or wave equation). The remainder of the course includes exercises with a CFD software and parallel discussion of the topics of Part III, "Art of CFD" dealing with turbulence modeling, computational grids, and rules of good CFD practice. This part is conducted in a computer laboratory and includes a project in which students perform a full-scale CFD analysis.

Acknowledgments: It is a pleasure to record my gratitude to many people who made writing this book possible. This includes generations of students at the University of Michigan–Dearborn, who suffered through the first iterations of the text and provided priceless feedback. I wish to thank friends and colleagues who read the manuscript and gave their insightful and constructive suggestions: Thomas Boeck, Dmitry Krasnov, Svetlana Poroseva, Tariq Shamim, Olga Shishkina, Sergey Smolentsev, Axelle Viré, and Anatoly Vorobev. The first serious attempt to write the book was undertaken during a sabbatical stay at the Ilmenau University of Technology. I appreciate the hospitality of Andre Thess and support by the German Science Foundation (DFG) that made this possible. Finally, and above all, I would like to thank my wife, Elena, and my children, Kirill and Sophia, for their understanding and support during the many hours it took to complete this book.

1

WHAT IS CFD?

1.1 INTRODUCTION

We start with a definition:

> CFD (computational fluid dynamics) is a set of numerical methods applied to obtain approximate solutions of problems of fluid dynamics and heat transfer.

According to this definition, CFD is not a science by itself but a way to apply the methods of one discipline (numerical analysis) to another (heat and mass transfer). We will deal with details later. Right now, a brief discussion is in order of why exactly we need CFD.

A distinctive feature of the science of fluid flow and heat and mass transfer is the approach it takes toward description of physical processes. Instead of bulk properties, such as momentum or angular momentum of a body in mechanics or total energy or entropy of a system in thermodynamics, the analysis focuses on *distributed properties.* We try to determine entire *fields* such as temperature $T(\boldsymbol{x}, t)$ velocity $\boldsymbol{v}(\boldsymbol{x}, t)$, density $\rho(\boldsymbol{x}, t)$, etc.[1] Even when an integral characteristic, such as the friction coefficient or the net rate of heat transfer, is the ultimate goal of analysis, it is derived from distributed fields.

The approach is very attractive by virtue of the level of details it provides. Evolution of the entire temperature distribution within a body can

[1]Throughout the book, we will use $\boldsymbol{x} = (x, y, z)$ for the vector of space coordinate and t for time.

be determined. Internal processes of a fluid flow such as motion, rotation, and deformation of minuscule fluid particles can be taken into account. Of course, the opportunities come at a price, most notably in the form of dramatically increased complexity of the governing equations. Except for a few strongly simplified models, the equations for distributed properties are *partial differential equations*, often nonlinear.

As an example of complexity, let us consider a seemingly simple task of mixing and dissolving sugar in a cup of hot coffee. An innocent question of how long or how many rotations of a spoon would it take to completely dissolve the sugar leads to a very complex physical problem that includes a possibly turbulent two-phase (coffee and sugar particles) flow with a chemical reaction (dissolving). Heat transfer (within the cup and between the cup and surroundings) may also be of importance because temperature affects the rate of the reaction. No simple solution of the problem exists. Of course, we can rely on the experience acquired after repeating the process daily (perhaps more than once) for many years. We can also add a couple of extra, possibly unnecessary, stirs. If, however, the task in question is more serious—for example, optimizing an oil refinery or designing a new aircraft—relying on everyday experience or excessive effort is not an option. We must find a way to *understand* and *predict* the process.

Generally, we can distinguish three approaches to solving fluid flow and heat transfer problems:

1. *Theoretical approach*—using governing equations to find analytical solutions
2. *Experimental approach*—staging a carefully designed experiment using a model of the real object
3. *Numerical approach*—using computational procedures to find a solution

Let's look at these approaches in more detail.

Theoretical approach. The approach has a crucial advantage of providing exact solutions. Among the disadvantages, the most important is that analytical solutions are only possible for a very limited class of problems, typically formulated in an artificial, idealized way. One example is the Poiseuille solution for a flow in an infinitely long pipe (see Figure 1.1). The steady-state laminar velocity profile is

$$U(r) = \frac{r^2 - R^2}{4\mu}\frac{dp}{dx},$$

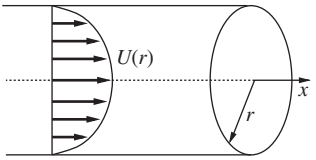


Figure 1.1 Laminar flow in an infinite pipe.

where U is the velocity, R is the pipe radius, dp/dx is the constant pressure gradient that drives the flow, and μ is the dynamic viscosity of the fluid. On the one hand, the solution is, indeed, simple and gives insight into the nature of flows in pipes and ducts, so its inclusion into all textbooks of fluid dynamics is not surprising. On the other hand, the solution is correct only if the pipe is infinitely long,[2] temperature is constant, and the fluid is perfectly incompressible. Furthermore, even if we were able to build such a pipe and find a useful application for it, the solution would be correct only at Reynolds numbers $Re = UR\rho/\mu$ (ρ is the density of the fluid) that are below approximately 2,000. Above this limit, the flow would assume fully three-dimensional and time-dependent turbulent form, for which no analytical solution is possible.

It can also be noted that derivation of analytical solutions often requires substantial mathematical skills, which are not among the strongest traits of many modern engineers and scientists, especially if compared to the situation of 30 or 40 years ago. Several reasons can be named for the deterioration of such skills, one, no doubt, being development of computers and numerical methods, including the CFD.

Experimental approach. Well-known examples are the wind tunnel experiments, which help to design and optimize the external shapes of airplanes (also of ships, buildings, and other objects). Another example is illustrated in Figure 1.2. The main disadvantages of the experimental approach are the technical difficulty (sometimes it takes several years before an experiment is set up and all technical problems are resolved) and high cost.

Numerical (computational) approach. Here, again, we employ our ability to describe almost any fluid flow and heat transfer process as a solution of a set of partial differential equations. An approximation to this solution is found in the result of a computational procedure. This approach is not problem-free, either. We will discuss the problems throughout the book.

[2]In practice, the solution is considered to be a good approximation for laminar flows in pipes at sufficiently large distance (dependent on the Reynolds number but, at least few tens of diameters) from the entrance.

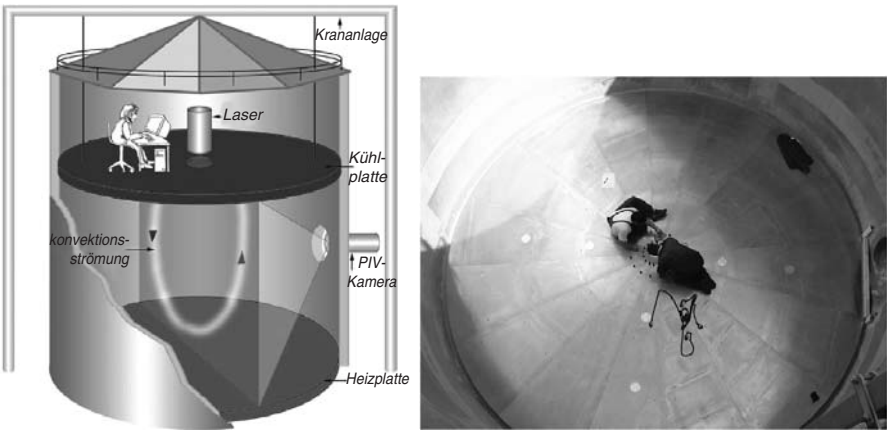


Figure 1.2 The experiment for studying thermal convection at the Ilmenau University of Technology, Germany (courtesy of A. Thess). Turbulent convection similar to the convection observed in the atmosphere of Earth or Sun is simulated by air motion within a large barrel with thermally insulated walls and uniformly heated bottom.

The computational approach, however, beats the analytical and experimental methods in some very important aspects: universality, flexibility, accuracy, and cost.

1.2 BRIEF HISTORY OF CFD

The history of CFD is a fascinating subject, which, unfortunately, we can only touch in passing. The idea to calculate approximate solutions of differential equations describing fluid flows and heat transfer is relatively old. It is definitely older than computers themselves. Development of numerical methods for solving ordinary and partial differential equations started in the first half of the twentieth century. The computations at that time required use of tables and dull mechanical work of dozens, if not hundreds, of people. No wonder that only the most important (primarily military-related) problems were addressed and only simple, one-dimensional equations were solved.

Invention and subsequent fast development of computers (see Figure 1.3) opened a wonderful possibility of performing millions—and then millions of millions—of arithmetic operations in a matter of seconds. This caused a rapid growth of the efforts to develop and apply methods of numerical simulations. Again, military applications, such as modeling shock waves from an explosion or a flow past a hypersonic

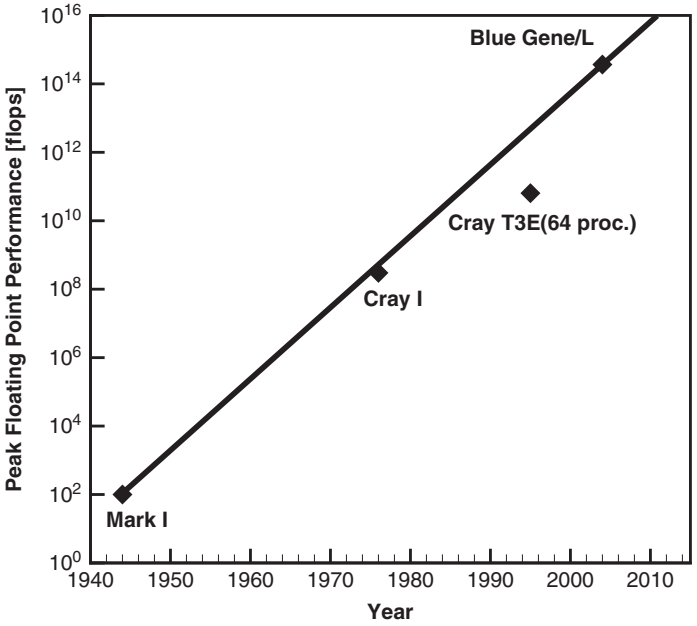


Figure 1.3 Development of high-performance computers. The speed measured as the number of floating operations per second grows approximately tenfold every five years.

jet aircraft were addressed first. In fact, development of faster and bigger computers until 1980s was largely motivated by the demands of military-related CFD. First simulations of realistic two-dimensional flows were performed in the late 1960s, while three-dimensional flows could not be seriously approached until the 1980s.

In the last 20 to 30 years, the computer revolution has changed the field of CFD entirely. From a scientific discipline, in which researchers worked on unique projects using specially developed codes, it has transformed into *an everyday tool of engineering design, optimization, and analysis*. The simulations are routinely used as a replacement of or addition to prototyping and other design techniques. The problem-specific codes are still developed for scientific purposes, but the engineering practice has almost entirely switched to the use of commercial or open-source CFD codes. The market is largely divided between a few major brands, such as FLUENT, STAR-CD, CFX, OpenFOAM, and COMSOL. They differ in appearance and capabilities but are all essentially the numerical solvers of partial differential equations with attached physical and turbulence models, as well as modules for grid generation and post-processing the results.

1.3 OUTLINE OF THE BOOK

This book is intended as a brief but complete introduction into CFD. The focus is not on development of algorithms but on the fundamental principles, formulation of CFD problems, the most basic and common computational techniques, and essentials of a good CFD analysis. The book's main task is to prepare the reader to make educated choices while using one of the ready CFD codes. A reader seeking deeper and more detailed understanding of specific computational methods is encouraged to use more advanced and more specialized texts, references to some of which are presented at the end of each chapter.

A comment is in order regarding the bias of the text. All CFD texts are, to some degree, biased in correspondence with the chosen audience and personal research interests of the authors. More weight is given to some of the methods (finite difference, finite element, spectral, etc.) and some of the fields of application (heat transfer, incompressible fluid dynamics, or gas dynamics). The preferences made in this book reflect the choice of mechanical, chemical, and civil engineers as the target audience and the intended use for applied CFD instruction. The focus is on the finite difference and finite volume methods. The finite element and spectral techniques are introduced, but only briefly. Also, more attention is given to numerical methods for incompressible fluid dynamics and heat transfer than for compressible sub- and supersonic flows.

The book contains 13 chapters. We are already at the end of Chapter 1. The remaining chapters are separated into three parts: "Fundamentals," "Methods," and "Art of CFD." Part I deals with the basic concepts of numerical solution of partial differential equations. It starts with Chapter 2 introducing the equations we are most likely to solve: the governing equations of fluid flows and heat transfer. We consider various forms of the equations used in CFD and review common boundary conditions. Necessary mathematical background and the concept of numerical approximation are presented in Chapter 3. Chapter 4 discusses the basics of the finite difference method. We also introduce the key concepts associated with all CFD methods, such as the truncation error and consistency of numerical approximation. The principles and main tools of the finite volume method are presented in Chapter 5. Chapter 6 is devoted to the concept of stability of numerical time integration. Some popular and important (both historically and didactically) schemes for one-dimensional model equations are presented in Chapter 7. The material summarizes the discussion of the fundamental concepts and can be used for a midterm programming project.

Part II, which includes Chapters 8 through 10, contains a compact description of some of the most important and commonly used CFD techniques. Methods of solution of systems of algebraic equations appearing in the result of the CFD approximation are discussed in Chapter 8. Chapter 9 presents some schemes used for nonsteady heat conduction and compressible flows. The discussion is deliberately brief for such voluminous subjects. It is expected that a reader with particular interest in any of them will refer to other, more specialized texts. Significantly more attention is given to the methods developed for computation of flows of incompressible fluids. Chapter 10 provides a relatively broad explanation of the issues, presents the projection method, and introduces some popular algorithms.

Part III consists of Chapters 11 to 13 and deals with subjects that are not directly related to the numerical solution of partial differential equations, but nevertheless are irreplaceable in practical CFD analysis. They all belong to a somewhat imprecise science in the sense that the approach is often decided on the basis of knowledge and experience rather than exact knowledge alone. The subjects in question are the turbulence modeling (Chapter 11), types and quality of computational grids (Chapter 12), and the complex of issues arising in the course of CFD analysis, such as uncertainty and validation of results (Chapter 13). The discussion is, by necessity, brief. A reader willing to acquire truly adequate understanding of these difficult but fascinating topics should consult the books listed at the end of each chapter.

REFERENCES AND SUGGESTED READING

http://www.top500.org/—Official Web site of the TOP500 project providing reliable and detailed information on the world most powerful supercomputers.

http://www.cfd-online.com/—A rich source of information on CFD: books, links, discussion forums, jobs, etc.

Part I

FUNDAMENTALS

2

GOVERNING EQUATIONS OF FLUID DYNAMICS AND HEAT TRANSFER

The methods of CFD can, at least in principle, be applied to any set of partial differential equations. The main area of application, however, has always been the solution of the equations describing processes of fluid flow and heat transfer. This chapter provides a brief description of the equations and can be skipped by a reader familiar with the matter. The material is included for the sake of completeness and is not intended as a replacement of the complete account found in comprehensive texts on fluid dynamics. Several such texts are listed at the end of the chapter.

2.1 PRELIMINARY CONCEPTS

From the physical viewpoint, the equations describing fluid flows and heat and mass transfer are simply versions of the conservation laws of classical physics, namely:

- Conservation of chemical species (law of conservation of mass)
- Conservation of momentum (Newton's second law of motion)
- Conservation of energy (first law of thermodynamics)

In some cases, additional equations are needed to account for other phenomena, such as, for example, entropy transport (the second law of thermodynamics) or electromagnetic fields.

Our starting point is the concept of the *continuous* media (solid or liquid) consisting of *elementary volumes* that are infinitesimal from the

macroscopic viewpoint but sufficiently large in comparison with the typical distance between molecules so they can themselves be considered as continua. In the case of a fluid flow, the elementary volumes, also called *fluid elements*, move around, rotate, and deform under the action of the forces acting in the flow and are defined as consisting of the same molecules at all times.

The conservation laws must be satisfied by any such fluid element. This can be mathematically expressed in two different ways. We can follow the so-called Largangian approach, where the equations are formulated directly in terms of properties of a given elementary volume moving in space. This approach is rarely used in CFD. Much more common is the Eulerian approach, in which the conservation principles applied to an elementary volume are reformulated in terms of distributed properties such as density $\rho(\boldsymbol{x}, t)$, temperature $T(\boldsymbol{x}, t)$, or velocity $\boldsymbol{v}(\boldsymbol{x}, t)$ considered as vector or scalar functions of space $\boldsymbol{x}$ and time t.

Our next step is to introduce the *material derivative*. Let us consider an element moving with the velocity $\boldsymbol{V}(x, y, z, t)$ in the fluid with density $\rho(x, y, z, t)$ as shown in Figure 2.1 (any other scalar field can be used instead of density without the loss of generality). The position vector of the element in the Cartesian coordinate system varies with time as $\boldsymbol{R}(t) = (x(t), y(t), z(t))$.

Differentiation of ρ with respect to time gives the rate of change of density *within the element*.

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho}{\partial x}\frac{dx(t)}{dt} + \frac{\partial \rho}{\partial y}\frac{dy(t)}{dt} + \frac{\partial \rho}{\partial z}\frac{dz(t)}{dt} = \frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x} + v\frac{\partial \rho}{\partial y} + w\frac{\partial \rho}{\partial z}, \tag{2.1}$$

where we have identified the time derivatives of the components of the position vector as the correspondent components of the local velocity $\boldsymbol{V} = u\boldsymbol{i} + v\boldsymbol{j} + w\boldsymbol{k}$. The right-hand side of the equation bears the name of

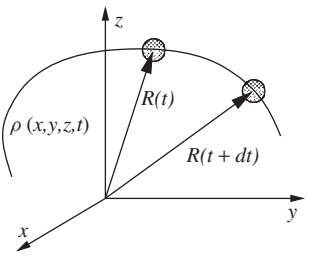


Figure 2.1 Elementary volume of fluid (fluid element) moving in a variable density field.

the *material derivative* and has special notation

$$\frac{D\rho}{Dt} \equiv \frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x} + v\frac{\partial \rho}{\partial y} + w\frac{\partial \rho}{\partial z} = \frac{\partial \rho}{\partial t} + \boldsymbol{V} \cdot \nabla \rho. \tag{2.2}$$

Similarly, the rate of change of temperature is given by

$$DT/Dt \equiv \partial T/\partial t + \boldsymbol{V} \cdot \nabla T,$$

while for the velocity component u we have

$$Du/Dt \equiv \partial u/\partial t + \boldsymbol{V} \cdot \nabla u.$$

The formulas clearly show that the rate of change of any distributed property consists of two parts, one due to the time variation of the property at a given location and another due to the motion of the element in a spatially variable field of this property.

Another important concept is associated with the fact that, while the mass of an element is conserved, its volume continuously changes as it moves and transforms in the flow. It can be viewed as the change of volume that occurs because the velocity field is space-dependent and so the velocity values at opposite sides of the element are different. Let us consider the two-dimensional situation illustrated in Figure 2.2. The element has the sizes dx and L, volume $\delta\mathcal{V} = Ldx$. The velocity field is purely one-dimensional $\boldsymbol{V} = u\boldsymbol{i}$, but x-dependent with $u = u(x)$. During the time interval dt, the right-hand side boundary moves together with fluid molecules by the distance $u(x + dx)dt$. The corresponding increase of volume is $Ldtu(x + dx)$. At the same time, the volume decreases by

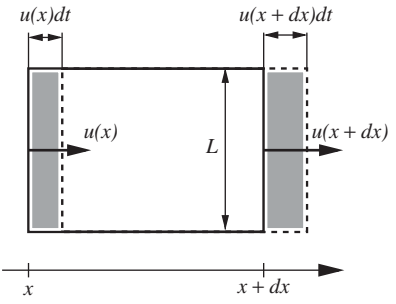


Figure 2.2 Change of the volume of fluid element because of spatial variability of velocity.

$Ldtu(x)$ due to the motion of the left-hand side boundary. The time rate of volume change per unit volume is

$$\frac{1}{\delta\mathcal{V}}\frac{d(\delta\mathcal{V})}{dt} = \frac{1}{Ldx}\lim_{dt\to 0}\frac{Lu(x+dx)dt - Lu(x)dt}{dt}$$
$$= \frac{(Lu(x+dx) - Lu(x))}{Ldx} = \frac{u(x+dx)-u(x)}{dx}.$$

Taking the limit of an infinitely small element $dx \to 0$ we find

$$\frac{1}{\delta\mathcal{V}}\frac{d(\delta\mathcal{V})}{dt} = \frac{du}{dx}.$$

In the general case of a three-dimensional velocity field $\boldsymbol{V} = (U, V, W)$, this formula generalizes to

$$\frac{1}{\delta\mathcal{V}}\frac{d(\delta\mathcal{V})}{dt} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = \nabla\cdot\boldsymbol{V}. \tag{2.3}$$

2.2 MASS CONSERVATION

We are now prepared to write down the first of the governing equations of the fluid motion. It expresses the law of conservation of mass. In a flow with density $\rho(\boldsymbol{x}, t)$ and velocity $\boldsymbol{V}(\boldsymbol{x}, t)$, we consider a fluid element of volume $\delta\mathcal{V}$. Since, according to the definition, the element consists of the same molecules at all times, its mass $\delta m = \rho\delta\mathcal{V}$ must remain constant:

$$\frac{d(\rho\delta\mathcal{V})}{dt} = \delta\mathcal{V}\frac{D\rho}{Dt} + \rho\frac{d(\delta\mathcal{V})}{dt} = 0.$$

Note that material derivative is used to represent the rate of change of density within the fluid element. Dividing by $\delta\mathcal{V}$ and applying (2.3) we obtain the *continuity equation*:

$$\frac{D\rho}{Dt} + \rho\nabla\cdot\boldsymbol{V} = 0, \tag{2.4}$$

which can be rewritten using (2.2) as

$$\frac{\partial\rho}{\partial t} + \boldsymbol{V}\cdot\nabla\rho + \rho\nabla\cdot\boldsymbol{V} = \frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\boldsymbol{V}) = 0. \tag{2.5}$$

In many flows, the compressibility of the fluid can be neglected (fluid dynamics textbooks provide the exact criteria). If this is the case, we can

assume that $\rho = const$ and reduce (2.4) or (2.5) to the *incompressibility equation*

$$\nabla \cdot \boldsymbol{V} = 0. \tag{2.6}$$

2.3 CONSERVATION OF CHEMICAL SPECIES

Let us now assume that the fluid is a composition of several chemical species, which can transform into each other by chemical reactions. A good example is the flow in a combustion chamber, where a mixture of a hydrocarbon fuel and air is burned to produce exhaust gases and energy. The law of conservation of mass still holds, of course, but the equations (2.4) and (2.5) have to be modified to account for chemical reactions and interspecies diffusion.

The diffusion is, to put it simply, a process of self-induced transport of chemical species from the location where their relative concentration is high to the location where the concentration is lower (see the books listed at the end of the chapter for an appropriately detailed and rigorous description). The transport is quantified by the vector field $\boldsymbol{J}_i(\boldsymbol{x}, t)$ of the *flux of a species* i, which denotes the direction and the rate of the mass flux of the species per unit area at the point $\boldsymbol{x}$. In the same manner as in the derivation of (2.3) we can find that the rate of change by diffusion of the mass content of species i in a fluid element of unit volume is $\nabla \cdot \boldsymbol{J}_i$.

The concentration of species can be expressed in terms of the *mass fraction* $m_i(\boldsymbol{x}, t)$, which is the ratio of the mass of species i to the total mass of the mixture in the same small volume. Another possibility is to use the concentration of species $C_i = m_i \rho$ defined as the mass of species i per unit volume. The conservation law is

$$\frac{\partial}{\partial t}(\rho m_i) + \nabla \cdot (\rho m_i \boldsymbol{V} + \boldsymbol{J}_i) = R_i, \tag{2.7}$$

where we introduced the source term R_i that accounts for the production/consumption of the species by chemical reactions.

The next step is to apply the Fick's law of diffusion, which is an empirical relation shown in experiments to be valid when variations of concentration are not very strong:

$$\boldsymbol{J}_i = -\Gamma_i \nabla m_i. \tag{2.8}$$

The conservation equation becomes

$$\frac{\partial}{\partial t}(\rho m_i) + \nabla \cdot (\rho m_i \boldsymbol{V}) = R_i + \nabla \cdot (\Gamma_i \nabla m_i). \tag{2.9}$$

If the Fick diffusion coefficients Γ_i are approximated as constants, the equation simplifies to

$$\frac{\partial}{\partial t}(\rho m_i) + \nabla \cdot (\rho m_i \boldsymbol{V}) = R_i + \Gamma_i \nabla^2 m_i. \tag{2.10}$$

2.4 CONSERVATION OF MOMENTUM

The underlying physical principle is Newton's second law, which states that the rate of change of momentum of a body is equal to the net force acting on it:

$$\frac{d}{dt}(m\boldsymbol{V}) = \boldsymbol{F}. \tag{2.11}$$

For a fluid element of unit volume moving within a flow, the left-hand side of (2.11) is replaced by the material derivative

$$\rho \frac{D}{Dt}(\boldsymbol{V}) = \rho \left[\frac{\partial}{\partial t}(\boldsymbol{V}) + (\boldsymbol{V} \cdot \nabla)\boldsymbol{V} \right]. \tag{2.12}$$

In the Cartesian coordinates, (2.12) is

$$\begin{aligned}
\rho \frac{Du}{Dt} &= \rho \left[\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} \right] \\
\rho \frac{Dv}{Dt} &= \rho \left[\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} \right] \\
\rho \frac{Dw}{Dt} &= \rho \left[\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} \right].
\end{aligned} \tag{2.13}$$

We can distinguish between two kinds of forces acting on a fluid element:

1. *Body forces*. They act directly on the mass of the fluid and originate from a remote source. The examples are the gravity, electric (Coulomb), magnetic, and Lorentz forces. Fictitious centrifugal and Coriolis forces, which appear when the flow is described in a rotating reference frame, also belong to this list. The total body force acting on a fluid element is proportional to its mass. In the following, we will assume that the body forces are lumped together into a net force of strength $\boldsymbol{f}$ per unit mass, so that the force per unit volume is $\rho\boldsymbol{f}$.

2. *Surface forces*. They are the pressure and friction forces acting between neighboring fluid elements and between a fluid element and an adjacent wall. It is shown in the fluid dynamics books that the vector field of surface forces can be represented as divergence of a symmetric 3×3 tensor called the *stress tensor* τ. Its component τ_{ij} can be seen as the i-component of the surface force acting on a unit area surface, which is normal to the j-axis of the Cartesian coordinate system. Here and in the rest of the book we assume that the values 1, 2, and 3 of indices i and j correspond to the Cartesian coordinates x, y, and z. The diagonal elements τ_{ii} cause extension/contraction of the fluid element, while the off-diagonal elements are responsible for its deformation by the shear (see Figure 2.3).

The Newton's second law can be written for a fluid element of unit volume as

$$\begin{aligned}
\rho\frac{Du}{Dt} &= \rho f_x + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} \\
\rho\frac{Dv}{Dt} &= \rho f_y + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} \\
\rho\frac{Dw}{Dt} &= \rho f_z + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z}.
\end{aligned} \tag{2.14}$$

The stress tensor can be separated into the isotropic pressure part, which is always present, and the viscous (friction) part, which exists only in flowing fluid and must be zero if the fluid is at rest:

$$\tau_{ij} = -p\delta_{ij} + \sigma_{ij}, \tag{2.15}$$

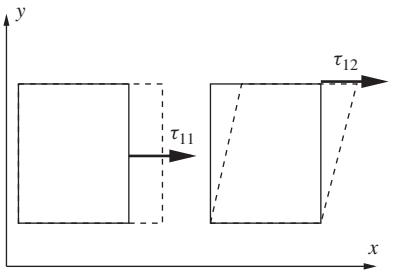


Figure 2.3 Illustration of normal (left) and shear (right) stresses acting on a fluid element.

where

$$\delta_{ij} = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases}$$

is the Kroneker delta-tensor.

For the equations to fully describe the flow, a model for the viscous stresses σ_{ij} has to be introduced. Newton was first to suggest that the shear stress must be proportional to the velocity gradient. This was later developed by Stokes into the linear model for the stress tensor:

$$\sigma_{ij} = \lambda \delta_{ij} (\nabla \cdot \boldsymbol{V}) + \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \tag{2.16}$$

where μ and λ are the first and second viscosity coefficients and we used u_i with $i = 1, 2, 3$ for the velocity components u, v, w. Note that in an incompressible fluid with $\nabla \cdot \boldsymbol{V} = 0$, the term with the second viscosity disappears. For compressible fluids, it is generally believed that $\lambda = -\frac{2}{3}\mu$ is an accurate approximation except for interior of shock waves in hypersonic flows and for absorption and attenuation of acoustic waves.

The model (2.16) does not have a fully satisfactory theoretical justification. It has, however, being validated in experiments and simply in everyday practice of applying the resulting equations. The fluids whose behavior satisfies the model are called Newtonian. There are non-Newtonian fluids that behave quite differently (e.g., polymer melts and solutions, human blood at high shear stress, etc.).

After substituting (2.16) into (2.14) and using the second viscosity assumption we obtain the final form of the momentum conservation equations, the Navier-Stokes equations:

$$\begin{aligned}
\rho \frac{Du}{Dt} &= \rho f_x - \frac{\partial p}{\partial x} + \frac{\partial}{\partial x} \left[\mu \left(-\frac{2}{3} \nabla \cdot \boldsymbol{V} + 2 \frac{\partial u}{\partial x} \right) \right] \\
&\quad + \frac{\partial}{\partial y} \left[\mu \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right) \right] + \frac{\partial}{\partial z} \left[\mu \left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \right) \right] \\
\rho \frac{Dv}{Dt} &= \rho f_y - \frac{\partial p}{\partial y} + \frac{\partial}{\partial y} \left[\mu \left(-\frac{2}{3} \nabla \cdot \boldsymbol{V} + 2 \frac{\partial v}{\partial y} \right) \right] \\
&\quad + \frac{\partial}{\partial x} \left[\mu \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right) \right] + \frac{\partial}{\partial z} \left[\mu \left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} \right) \right]
\end{aligned} \tag{2.17}$$

$$\rho\frac{Dw}{Dt} = \rho f_z - \frac{\partial p}{\partial z} + \frac{\partial}{\partial z}\left[\mu\left(-\frac{2}{3}\nabla\cdot\boldsymbol{V} + 2\frac{\partial w}{\partial z}\right)\right] + \frac{\partial}{\partial x}\left[\mu\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right)\right] + \frac{\partial}{\partial y}\left[\mu\left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right)\right].$$

The equations can be written in a shorter form if we introduce the rate of strain tensor with components

$$S_{ij} \equiv \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{2.18}$$

and use the Einstein summation convention, according to which repeated indices in a term imply summation over all their possible values (1, 2, 3 in our case):

$$\rho\frac{Du_i}{Dt} = \rho f_i - \frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[2\mu S_{ij} - \frac{2}{3}\mu(\nabla\cdot\boldsymbol{V})\delta_{ij}\right] \tag{2.19}$$

For the special case of an incompressible fluid with constant viscosity coefficient μ, the Navier-Stokes equations become

$$\rho\frac{D\boldsymbol{V}}{Dt} = -\nabla p + \mu\nabla^2\boldsymbol{V} + \rho\boldsymbol{f}. \tag{2.20}$$

Another special case is that of an inviscid fluid with $\mu = \lambda = 0$, for which the so-called Euler equations are valid:

$$\rho\frac{D\boldsymbol{V}}{Dt} = -\nabla p + \rho\boldsymbol{f}. \tag{2.21}$$

Of course, both (2.20) and (2.21) must be understood as idealizations, strictly speaking, achievable only as asymptotic limits of flows with very low compressibility and very small viscosity, respectively. This does not prevent them from being widely used as approximations.

2.5 CONSERVATION OF ENERGY

The energy conservation principle can be formulated for a fluid element in the manner similar to the mass and momentum conservation (see the books listed at the end of the chapter for a derivation) as

$$\rho\frac{De}{Dt} = -\nabla\cdot\boldsymbol{q} - p(\nabla\cdot\boldsymbol{V}) + \dot{Q}, \tag{2.22}$$

where $e(\boldsymbol{x},t)$ is the internal energy per unit mass, $\boldsymbol{q}(\boldsymbol{x},t)$ is the vector field of the heat flux by thermal conduction, and $\dot{Q}$ is the rate of internal heat generation by the effects such as, for example, viscous friction or radiation. The conduction heat flux can be described by the Fourier conduction law

$$\boldsymbol{q} = -\kappa \nabla T, \tag{2.23}$$

where $T(\boldsymbol{x},t)$ is the temperature field and κ is the coefficient of thermal conductivity.

The energy conservation equation can also be written in the enthalpy form

$$\rho \frac{Dh}{Dt} = \frac{Dp}{Dt} + \dot{Q} - \nabla \cdot \boldsymbol{q}, \tag{2.24}$$

where $h = e + p/\rho$ is the specific enthalpy. Yet another possibility is the equation for the total (internal plus mechanical) energy E

$$\rho \frac{DE}{Dt} = -\nabla \cdot \boldsymbol{q} - \nabla \cdot (p\boldsymbol{V}) + \dot{Q} + \rho \boldsymbol{f} \cdot \boldsymbol{V}. \tag{2.25}$$

The energy equation has more complex form if extra effects such as exo- and endothermal chemical reactions, radiation heat transfer, or Joule dissipation are explicitly shown in the right-hand side. In some cases, the equation can be brought into much simpler form. This is, in particular, true when the internal heat generation can be neglected and the fluid can be considered incompressible (the Boussinesq approximation). For an incompressible fluid or a solid, the specific internal energy is $e = CT$, where $C = C_p = C_v$ is the specific heat. The energy equation becomes

$$\rho C \frac{DT}{Dt} = -\nabla \cdot \boldsymbol{q}. \tag{2.26}$$

Substituting (2.23), we obtain the equation of convection heat transfer

$$\rho C \frac{DT}{Dt} = \rho C \left(\frac{\partial T}{\partial t} + \boldsymbol{V} \cdot \nabla T \right) = \nabla \cdot (\kappa \nabla T), \tag{2.27}$$

which, in the case of a quiescent fluid or a solid and constant conduction coefficient κ, reduces to the classical heat conduction equation

$$\rho C \frac{\partial T}{\partial t} = \kappa \nabla^2 T. \tag{2.28}$$

2.6 EQUATION OF STATE

To close the system of governing equations, we have to add an equation of state, which connects the thermodynamic variables p, ρ, and T. We also need an expression for the internal energy in terms of the thermodynamic variables. The simplest and most widely used are the ideal gas model

$$p/\rho = RT, \quad e = e(T), \tag{2.29}$$

and the model of incompressible fluid

$$\rho = const, \quad e = CT, \tag{2.30}$$

although many other models are possible and often necessary.

If the physical coefficients, such as viscosity μ or conductivity κ, are not assumed constant, we have to include formulas giving values of these coefficients as functions of temperature and other variables.

2.7 EQUATIONS IN INTEGRAL FORM

A different approach to the derivation of governing equations can be taken, in which the conservation principles are applied not to a fluid element moving with the flow but to a control volume fixed in space. Instead of analyzing the effect of moving boundaries as, for example, in Figure 2.2 we have, in this case, to take into account the fluid flow through the boundaries of the element and the associated transport of conserved quantities.

Let us start with the conservation of mass. The total mass of fluid in a control volume Ω (see Figure 2.4) is $M = \int_\Omega \rho d\Omega$. By virtue of

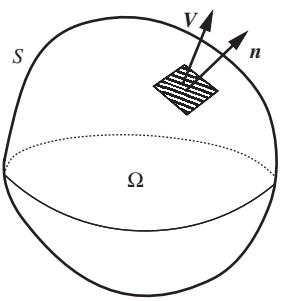


Figure 2.4 Control volume for derivation of equations in integral form.

conservation of mass, M can only change because of transport of mass into or out of the control volume by the flow. The correct term is the *flux* of mass. We are only interested in the flux component normal to the boundary S of Ω, since the tangential component does not change the mass inside Ω. The magnitude of this component per unit time and unit surface area is $\boldsymbol{V} \cdot \boldsymbol{n}\rho$, where $\boldsymbol{n}$ is the normal vector of unit length shown in Figure 2.4. It is conventional to use an outward facing normal, so a positive flux means mass flow out of the control volume. Integrating over the boundary, we obtain the equation for the net mass balance

$$\frac{d}{dt}\int_{\Omega} \rho d\Omega + \int_{S} \rho \boldsymbol{V} \cdot \boldsymbol{n} dS = 0. \tag{2.31}$$

This is the mass conservation equation in the integral form. It is important to realize that the control volume Ω can be of arbitrary size and shape. For example, it can be the entire flow domain or a small elementary cell of a finite volume grid (see Chapter 5).

The integral equation of conservation of momentum is derived similarly to (2.31). The fluxes are defined for the three momentum components as $\rho u \boldsymbol{V}$, $\rho v \boldsymbol{V}$, and $\rho w \boldsymbol{V}$. Formulating the balance, we have to include the body forces within the control volume Ω and the surface forces at the surface S. The equations are

$$\begin{aligned}
\frac{d}{dt}\int_{\Omega} \rho u d\Omega + \int_{S} \rho u \boldsymbol{V} \cdot \boldsymbol{n} dS &= \int_{S} \boldsymbol{t}_x \cdot \boldsymbol{n} dS + \int_{\Omega} \rho f_x d\Omega \\
\frac{d}{dt}\int_{\Omega} \rho v d\Omega + \int_{S} \rho v \boldsymbol{V} \cdot \boldsymbol{n} dS &= \int_{S} \boldsymbol{t}_y \cdot \boldsymbol{n} dS + \int_{\Omega} \rho f_y d\Omega \\
\frac{d}{dt}\int_{\Omega} \rho w d\Omega + \int_{S} \rho w \boldsymbol{V} \cdot \boldsymbol{n} dS &= \int_{S} \boldsymbol{t}_z \cdot \boldsymbol{n} dS + \int_{\Omega} \rho f_z d\Omega,
\end{aligned} \tag{2.32}$$

where $\boldsymbol{t}_x$, $\boldsymbol{t}_y$, and $\boldsymbol{t}_z$ are the vectors with the first, second, and third rows of the stress tensor τ_{ij} as components. The stresses should be expressed through the velocity components. For the Navier-Stokes model (2.15), (2.16), we obtain, for the i-component of momentum,

$$\begin{aligned}
&\frac{d}{dt}\int_{\Omega} \rho u_i d\Omega + \int_{S} \rho u_i \boldsymbol{V} \cdot \boldsymbol{n} dS \\
&\quad = \int_{S} \left[(-p + \lambda \nabla \cdot \boldsymbol{V}) n_i + \sum_j \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) n_j \right] dS + \int_{\Omega} \rho f_i d\Omega.
\end{aligned} \tag{2.33}$$

For the energy balance, we remember that energy flow in or out of Ω can be accomplished in two ways: by heat conduction and by motion of matter through the boundary surface S. Accordingly, we consider the normal conduction and convection fluxes $\boldsymbol{q} \cdot \boldsymbol{n}$ and $\rho E \boldsymbol{V} \cdot \boldsymbol{n}$ at the boundary. Taking into account the work by body and surface forces, the integral analog of (2.25) is formulated as

$$\frac{d}{dt}\int_{\Omega} \rho E d\Omega + \int_{S} \boldsymbol{q} \cdot \boldsymbol{n} dS + \int_{S} \rho E \boldsymbol{V} \cdot \boldsymbol{n} dS = \int_{S} -p\boldsymbol{V} \cdot \boldsymbol{n} dS + \int_{\Omega} \dot{Q} d\Omega + \int_{\Omega} \rho \boldsymbol{f} \cdot \boldsymbol{V} d\Omega. \tag{2.34}$$

When the fluid is incompressible and the energy generation by internal sources is negligible, we obtain the integral analog of the equation of convection heat transfer (2.27)

$$\frac{d}{dt}\int_{\Omega} \rho C T d\Omega + \int_{S} \rho C T \boldsymbol{V} \cdot \boldsymbol{n} dS = \int_{S} \kappa \nabla T \cdot \boldsymbol{n} dS. \tag{2.35}$$

The integral equations (2.31)–(2.35) all have similar mathematical structure, which reflects conceptual similarity of the physical processes they describe. Each equation contains a term with time derivative of a volume integral, which gives the rate of change of the amount of the conserved quantity within the control volume. The rate of change is balanced by several volume and surface integrals, each corresponding to a certain factor responsible for the change. The integrals are of the following types:

- *Convective flux integrals*. The surface integrals that do not contain derivatives of the conserved field. They represent the transport by the velocity $\boldsymbol{V}$ through the boundary of the control volume. The examples are $\int_S \rho \boldsymbol{V} \cdot \boldsymbol{n} dS$ in (2.31), $\int_S \rho u_i \boldsymbol{V} \cdot \boldsymbol{n} dS$ in (2.33), and $\int_S \rho C T \boldsymbol{V} \cdot \boldsymbol{n} dS$ in (2.35).
- *Diffusive flux integrals*. The surface integrals that contain first derivatives of the conserved field. These integrals represent the transport through the boundary by diffusion, heat conduction, or viscosity. The derivatives are typically multiplied by the corresponding transport coefficients. The example are $\int_S \mu \left(\partial u_i/\partial x_j + \partial u_j/\partial x_i\right) n_j dS$ in (2.33) and $\int_S \kappa \nabla T \cdot \boldsymbol{n} dS$ in (2.35).
- *Volume source integrals*. The volume integrals corresponding to distributed sources or sinks of the conserved quantity within the control volume, such as $\int_\Omega \rho f_i d\Omega$ in (2.33) or $\int_\Omega \dot{Q} d\Omega$ and $\int_\Omega \rho \boldsymbol{f} \cdot \boldsymbol{V} d\Omega$ in (2.34).

- *Surface force integrals*. The surface integrals representing the work by normal surface forces on the boundary of the control volume. Only one example is present in our equations, the pressure term $\int_S (-p)\boldsymbol{n}\,dS$ in (2.33).

It is convenient for future use, especially for development of finite volume schemes (see Chapter 5), to leave the pressure term for separate consideration and combine the other three types in a formal integral conservation equation for an arbitrary scalar field Φ:

$$\underbrace{\frac{d}{dt}\int_\Omega \Phi d\Omega}_{\textit{Rate of change}} + \underbrace{\int_S \Phi \boldsymbol{V}\cdot\boldsymbol{n}\,dS}_{\textit{Convective flux}} = \underbrace{\int_S \chi\nabla\Phi\cdot\boldsymbol{n}\,dS}_{\textit{Diffusive flux}} + \underbrace{\int_\Omega Q d\Omega}_{\textit{Volume source}}. \tag{2.36}$$

2.8 EQUATIONS IN CONSERVATION FORM

Since the integral equations describe the same physical processes as the differential governing equations such as (2.4), (2.14), and (2.25), they have to be equivalent mathematically. Let us try to derive the differential equations from the integral ones. We will do it for the formal conservation law (2.36). The procedure is very simple and consists of two steps. First, we convert the surface integrals into volume integrals by using the divergence theorem:

$$\int_S \Phi\boldsymbol{V}\cdot\boldsymbol{n}\,dS = \int_\Omega \nabla\cdot(\Phi\boldsymbol{V})d\Omega, \quad \int_S \chi\nabla\Phi\cdot\boldsymbol{n}\,dS = \int_\Omega \nabla\cdot(\chi\nabla\Phi)d\Omega, \tag{2.37}$$

which is true for any vector field ($\Phi\boldsymbol{V}$ and $\chi\nabla\Phi$ in our case) with continuous first derivatives and any volume Ω with a piecewise smooth boundary S. The equation (2.36) can be rewritten as

$$\int_\Omega \left(\partial\Phi/\partial t + \nabla\cdot(\Phi\boldsymbol{V}) - \nabla\cdot(\chi\nabla\Phi) - Q\right) d\Omega = 0.$$

We now remember that it is satisfied for an arbitrary volume Ω, which is only possible if the integrand itself is zero, which leads to

$$\frac{\partial\Phi}{\partial t} + \nabla\cdot(\Phi\boldsymbol{V}) = Q + \nabla\cdot(\chi\nabla\Phi). \tag{2.38}$$

Similar procedures applied to the mass and momentum conservation equations (2.31) and (2.32) result in

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{V}) = 0, \tag{2.39}$$

$$\begin{aligned}
\frac{\partial (\rho u)}{\partial t} + \nabla \cdot (\rho u \boldsymbol{V}) &= \rho f_x + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} \\
\frac{\partial (\rho v)}{\partial t} + \nabla \cdot (\rho v \boldsymbol{V}) &= \rho f_y + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} \\
\frac{\partial (\rho w)}{\partial t} + \nabla \cdot (\rho w \boldsymbol{V}) &= \rho f_z + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z}.
\end{aligned} \tag{2.40}$$

The integral equation for energy conservation (2.34) transforms into

$$\frac{\partial (\rho E)}{\partial t} + \nabla \cdot (\rho E \boldsymbol{V}) = -\nabla \cdot \boldsymbol{q} - \nabla \cdot (p \boldsymbol{V}) + \dot{Q} + \rho \boldsymbol{f} \cdot \boldsymbol{V}. \tag{2.41}$$

The equations (2.39)–(2.41) are in the *conservation form* also called *conservation-law form*, *conservative form*, or *divergence form*. Their defining property is that each term corresponds directly to a term of the integral equation: the time derivative to the rate of change of the conserved quantity in a fixed control volume, nonderivative terms to the volume integrals of sources, and divergence terms to the surface integrals of fluxes. The derivative terms have coefficients, which are either constant or, if variable, not appearing elsewhere in the equations under a derivative sign.

It can be easily shown that the equations in the conservation form (2.39)–(2.41) are mathematically equivalent to the original equations (2.4), (2.14), and (2.25). When, however, the equations are solved numerically on a computational grid, the approximations are not necessarily equivalent and the results of calculations can be different.

An important feature of the numerical schemes based on the approximation of the equations in their conservation form is that such schemes, if properly arranged, conserve the quantities (mass, momentum, energy, etc.) exactly and in the global sense, that is for the entire computational domain.

2.9 EQUATIONS IN VECTOR FORM

It is sometimes convenient for development and analysis of computational algorithms to present the governing equations in a compact vector form.

We can easily do this for the equations in conservation form (2.39)–(2.41). Let us introduce the vector fields

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix}, \boldsymbol{Q} = \begin{bmatrix} 0 \\ \rho f_x \\ \rho f_y \\ \rho f_z \\ \dot{Q} + \rho(\boldsymbol{f} \cdot \boldsymbol{V}) \end{bmatrix}, \boldsymbol{A} = \begin{bmatrix} \rho u \\ \rho u^2 + p - \sigma_{xx} \\ \rho uv - \sigma_{xy} \\ \rho uw - \sigma_{xz} \\ (\rho E + p)u + q_x \end{bmatrix}, \tag{2.42}$$

$$\boldsymbol{B} = \begin{bmatrix} \rho v \\ \rho uv - \sigma_{xy} \\ \rho v^2 + p - \sigma_{yy} \\ \rho vw - \sigma_{yz} \\ (\rho E + p)v + q_y \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} \rho w \\ \rho uw - \sigma_{xz} \\ \rho vw - \sigma_{yz} \\ \rho w^2 + p - \sigma_{zz} \\ (\rho E + p)w + q_z \end{bmatrix}. \tag{2.43}$$

The system (2.39)–(2.41) abbreviates to

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{A}}{\partial x} + \frac{\partial \boldsymbol{B}}{\partial y} + \frac{\partial \boldsymbol{C}}{\partial z} = \boldsymbol{Q} \tag{2.44}$$

or, when the body forces and internal heat sources are negligible, to

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{A}}{\partial x} + \frac{\partial \boldsymbol{B}}{\partial y} + \frac{\partial \boldsymbol{C}}{\partial z} = 0. \tag{2.45}$$

2.10 BOUNDARY CONDITIONS

In principle, one can say that all parts of the universe are connected to each other by fluxes of heat and mass and, thus, must be included into a good CFD solution. Since such an enterprise is hardly feasible, we have to compromise and formulate CFD problems for *finite* domains limited by *boundaries*. Such boundaries often appear naturally. For example, they can follow rigid walls. Sometimes, however, the choice is, by necessity, artificial. Several examples of such artificial boundaries are considered in sections 2.10.2 and 2.10.3. In any case, a correctly formulated CFD problem should include a set of proper *boundary conditions* for velocity, temperature, and other variables.

The importance of setting appropriate boundary conditions should not be underestimated. No correct CFD solution can be obtained without them. In practical CFD, when one of the general purpose codes is used, setting

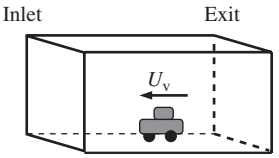


Figure 2.5 Example of a solution domain requiring boundary conditions at solid walls, inlet, and exit.

the boundary conditions is one of the key "creative" acts performed by the user.

The exact meaning and detailed discussion of appropriate boundary conditions for particular problems and physical models can be found in the books on fluid dynamics and heat transfer. Here, we give only a brief review of the most common types using simple examples. In the first example shown in Figure 2.5, a car is moving within a tunnel. The task is to calculate the air flow and temperature distribution around the car. The flow in the entire tunnel is impossible to simulate (the reasons of which will be discussed throughout the book) so we use the computational domain in the form of a tunnel segment extending few meters ahead and behind the vehicle.[1]

2.10.1 Rigid Wall Boundary Conditions

At the rigid walls, the velocity boundary conditions are different for viscous ($\mu \neq 0$) and inviscid ($\mu = 0$) flows. For viscous flows, the *no-slip conditions* are applied:

$$\boldsymbol{V} = \boldsymbol{U}_{\text{wall}} \text{ at the wall.} \tag{2.46}$$

In our example, if we use a reference frame moving with the car, the conditions are

$\boldsymbol{V} = 0$ at the surface of the vehicle

$\boldsymbol{V} = -\boldsymbol{U}_V$ at the walls of the tunnel.

For inviscid flows, the *impermeable wall* conditions are applied, according to which only the velocity component normal to the wall is required to

[1]The nature of this particular flow requires a rather short (few meters) distance before the vehicle, while the distance behind it has to be larger if we want to analyze the wake behind the car.

match the corresponding component of the wall velocity. The tangential component can *slip:*

$$\boldsymbol{V} \cdot \boldsymbol{n} = \boldsymbol{U}_{\text{wall}} \cdot \boldsymbol{n} \text{ at the wall.} \tag{2.47}$$

We will assume that the normal $\boldsymbol{n}$ faces outward with respect to a fluid element and into the wall.

For temperature, two asymptotic limits can be used. One is the condition of known wall temperature T_{wall} (imagine a wall in the form of a large copper slab kept at this temperature):

$$T = T_{\text{wall}} \text{ at the wall.} \tag{2.48}$$

Another is the condition of known normal heat flux into the wall q_{wall}:

$$\frac{\partial T}{\partial n} = \nabla T \cdot \boldsymbol{n} = -\frac{1}{\kappa} q_{\text{wall}} \text{ at the wall.} \tag{2.49}$$

The special case of the latter is a perfectly insulating wall:

$$\frac{\partial T}{\partial n} = \nabla T \cdot \boldsymbol{n} = 0 \text{ at the wall.} \tag{2.50}$$

The Newton's cooling law can be used as a more realistic boundary condition when neither of the two asymptotic limits is acceptable. The heat flux is taken to be proportional to the difference between the temperatures on two sides of the boundary:

$$q_{\text{wall}} = h(T - T_{\text{wall}}), \tag{2.51}$$

where h is an empirical cooling constant.[2] A combination of (2.49) and (2.51) results in the boundary condition

$$\kappa \nabla T \cdot \boldsymbol{n} + h(T - T_{\text{wall}}) = 0. \tag{2.52}$$

For our example of a car in a tunnel, the tunnel walls can be assumed perfectly insulating (2.50) while (2.49) or (2.52) can be used for the vehicle surface.

[2]A common problem with this approximation is that the coefficient h, which is determined by properties of often turbulent thermal boundary layer, cannot be given a reliable and accurate quantitative estimate.

2.10.2 Inlet and Exit Boundary Conditions

If the computational domain has open boundaries, such as the inlet and exit in our example, special boundary condition must be set at them. The common choice for the inlet is to prescribe velocity and temperature:

$$\boldsymbol{V} = \boldsymbol{U}_{\text{inlet}}, \quad T = T_{\text{inlet}}, \quad \text{at the inlet.} \tag{2.53}$$

Parameters of turbulent fluctuations should also be prescribed if the flow is turbulent (see Chapter 11).

At the exit, any boundary condition would be artificial since we artificially cut off a part of the flow generated in the car-tunnel system and have no possibility to predict what happens there and how this can affect the flow inside the computational domain. One commonly used approximation is that of zero streamwise gradient (gradient in the direction of the flow x)

$$\frac{\partial \boldsymbol{V}}{\partial x} = 0, \quad \frac{\partial T}{\partial x} = 0, \quad \text{at the exit.} \tag{2.54}$$

2.10.3 Other Boundary Conditions

In many situations, we can make more or less plausible assumptions about the nature of the solution before it is actually computed. This can help to reduce the size of the computational domain, computational grid, and, thus, the amount of computations. An illustration is given in Figure 2.6. A flow in a circular pipe with a series of equidistant ringlike obstructions is calculated. Two assumptions can be made, especially if our interest is in the mean (average) state of a turbulent flow: that the flow is axially symmetric and that its structure is periodic, repeating itself in every groove between the obstructions.

Relying on the first assumption allows us to consider a two-dimensional solution with all variables depending on the axial z and radial r coordinates of the cylindrical coordinate system instead of the general three-dimensional solution. The computational domain lies in the r-z plane and is limited by the solid walls and the symmetry axis. Special boundary conditions that guarantee regularity of solution have to be imposed at $r = 0$. The engineering CFD codes usually provide such conditions as an option.

The assumption of periodicity allows us to reduce the computational domain in the axial direction. Since the flows in the grooves are identical, only one of them needs to be computed. We can introduce periodic (cyclic)

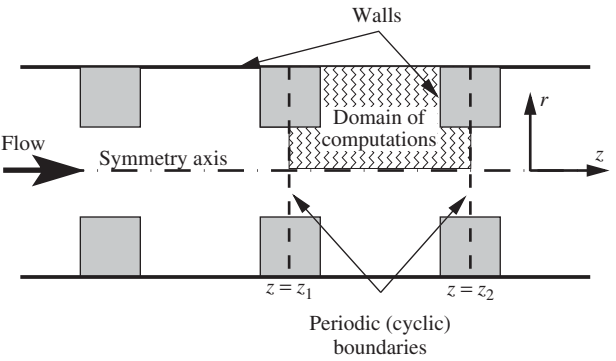


Figure 2.6 Flow in a circular pipe with periodic obstructions as an example of setting symmetry and cyclic boundary conditions.

boundaries as shown in Figure 2.6 and require that the solution variables are reproduced on such boundaries periodically:

$$\boldsymbol{V}(r, z_1) = \boldsymbol{V}(r, z_2), \quad T(r, z_1) = T(r, z_2). \tag{2.55}$$

The geometry-based simplifying assumptions like those just illustrated are useful. They allow us to reduce the size of the computational domain and to consider two-dimensional flows instead of three-dimensional. As a result, solutions can be obtained more accurately, on a finer computational grid, and at lower computational cost. The assumptions should, however, be used with caution. The actual flow structure does not necessarily follow the symmetries suggested by the geometry. For example, hydrodynamic instabilities and other effects would, in many cases, transform the flow in Figure 2.6 into a three-dimensional and nonperiodic form.

REFERENCES AND SUGGESTED READING

Kundu, P. K. and I. M. Cohen, *Fluid Mechanics*. London: Elsevier Academic Press, 2004.

Landau, L. D. and E. M. Lifshitz, *Fluid Mechanics (Volume 6 of the course of theoretical physics)*. Woburn, MA: Butterworth-Heinemann, 1987.

White, F. M. *Viscous Fluid Flow*, New York: McGraw Hill, 2005.

PROBLEMS

1. Write the formula for the material derivative of concentration C of salt in a flow of salt solution. What does it represent?

2. What is Newtonian fluid?

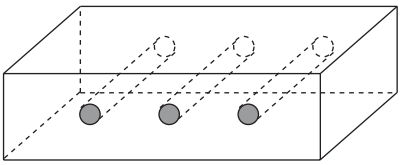
3. What are the models of incompressible fluid and ideal gas?

4. Verify that the Navier-Stokes equations (2.17) reduce to the equations (2.20) in the case of a flow with constant density and viscosity.

5. Following the procedure described in section 2.8, derive the continuity equation (2.5) from the integral mass conservation equation (2.31).

6. Verify that the Poiseuille solution for a laminar flow in an infinitely long circular pipe discussed in Chapter 1 satisfies the mass and momentum conservation equations for a steady flow of an incompressible Newtonian fluid. Refer to fluid dynamics books for the equations expressed in cylindrical coordinates.

7. Define the computational domain and write the full system of governing equations and boundary conditions for the following situations. In all of them, consider a long straight duct with smooth walls and uniformly distributed circular pipes crossing the duct in the direction perpendicular to the duct axis and parallel to one set of walls:



a) There is a flow of air along the duct. Air can be assumed incompressible and having constant temperature equal to the temperature of the duct walls and pipes.

b) The same as in (a), but now temperature varies. The cylinders are maintained at constant temperature T_c, which is significantly higher than the air temperature T_i at the duct inlet. The duct walls are thermally perfectly insulating. Air is still assumed incompressible.

c) The duct is now filled with a solid material of density ρ, specific heat C, and conductivity κ. Temperature of the cylinders is T_{c} and the temperature of the walls is T_{w}.

3

PARTIAL DIFFERENTIAL EQUATIONS

From the mathematical viewpoint, the equations of fluid flows and heat transfer are partial differential equations (PDE). Before we actually start solving them and see the difficulties, we need to consider their mathematical properties. The reason for that will be illustrated in this and the following chapters. Certain properties of the equations have profound effect on the behavior of solutions and, significantly for us, on the choice of numerical method.

The full governing equations, such as (2.39)–(2.41), are complex. Exact analytical solution can only be found in a few strongly simplified situations. It should not, therefore, be surprising that the analysis of the mathematical issues and the initial development of numerical methods are usually conducted for the simple model equations, for which analytical solutions are available, and which possess principal mathematical properties of the original equations.

We will follow this approach and start by presenting model partial differential equations for a scalar field $u(\boldsymbol{x}, t)$. These equations will be used throughout the Part I of the book to illustrate the basic principles of numerical methods. In this chapter, the model equations will help us to present the elements of a well-posed PDE problem and to discuss the mathematical classification of PDE, its consequences for the solution properties, and relevance to fluid dynamics and heat transfer. At the end of the chapter, we will introduce the concept of numerical discretization of a PDE problem and review the main discretization techniques.

3.1 MODEL EQUATIONS; FORMULATION OF A PDE PROBLEM

3.1.1 Model Equations

We begin with the list of all the model equations used in our discussion.

Heat Equation: The heat equation was derived in section 2.5. It expresses the energy conservation principle in the case of conduction heat transfer with constant physical properties and absent sources of internal heat generation:

$$\frac{\partial u}{\partial t} = a^2 \nabla^2 u, \tag{3.1}$$

where $u(\boldsymbol{x}, t)$ is the temperature field and $a^2 = \kappa/\rho C$ is the temperature diffusivity coefficient. In fact, the same equation can be used to describe many other processes, such as, for example, diffusion of an admixture in a quiescent fluid or evolution of an initially sharp velocity gradient in a viscous flow. In the one-dimensional case, the equation reduces to

$$\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2}. \tag{3.2}$$

Wave Equation: The wave equation

$$\frac{\partial^2 u}{\partial t^2} = a^2 \nabla^2 u \tag{3.3}$$

describes wavelike phenomena such as sound propagation or oscillations of a string or membrane. In the one-dimensional case, the equation is

$$\frac{\partial^2 u}{\partial t^2} = a^2 \frac{\partial^2 u}{\partial x^2}. \tag{3.4}$$

Linear Convection Equation: Another, even simpler, equation can be used as a representative of the equations with wavelike solutions. This is the so-called linear convection equation

$$\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} = 0, \tag{3.5}$$

where c is a positive constant.

Laplace and Poisson Equations: The Laplace equation

$$\nabla^2 u = 0 \tag{3.6}$$

can be considered as a version of the heat equation (3.1) when $\partial u/\partial t = 0$. An important generalization is the Poisson equation

$$\nabla^2 u = f(\boldsymbol{x}), \tag{3.7}$$

where f is a known function of spatial coordinates. The simplest PDE form of (3.7) is for the two-dimensional case $u = u(x, y)$:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y). \tag{3.8}$$

Burgers and Generic Transport Equations: The Burgers equation is

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \mu\frac{\partial^2 u}{\partial x^2}, \tag{3.9}$$

where $u = u(x, t)$ and $\mu \geq 0$ is a constant coefficient. The equation was suggested by J. M. Burgers in 1948 as a one-dimensional model for the Navier-Stokes dynamics, and, presumably, for turbulence in fluid flows. The terms of the equation can be considered as counterparts of unsteady, convective, and viscous terms of the momentum conservation equation of the Navier-Stokes system.

A modification of (3.9) often considered in the literature is the one-dimensional generic transport equation:

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} = \mu\frac{\partial^2 \phi}{\partial x^2}, \tag{3.10}$$

where ϕ is a transported and diffused scalar field (e.g., temperature) and $u(x, t)$ is a known function acting as a velocity-like transporting agent.

It has become clear with time that turbulence is an essentially three-dimensional phenomenon and cannot be modeled by (3.9). Similarly, (3.10) is not a good model for the majority of heat and mass transfer processes, which are either two- or three-dimensional. It has also become clear that the equations (3.9) and (3.10) serve as excellent benchmarks for development and testing of CFD methods.

3.1.2 Domain, Boundary, and Initial Conditions

The main entity of the PDE analysis is not a separate equation but a complete PDE problem consisting of an equation, domain of solution, boundary and initial conditions. The problem has to be solved in a *spatial domain* Ω and, in the case of time-dependency, in a *time interval* between t_0 and t_{end} (see Figure 3.1). t_{end} can be a finite number or infinity. Similarly, the domain Ω may have a finite size or extend to infinity in one or several directions. In numerical simulations, the infinite limits of the spatial or time domain are replaced by sufficiently large finite numbers.

Boundary conditions have to be imposed at the boundaries of the spatial domain $\partial\Omega$. This is necessary not only to account for the effect of real physical boundaries but also from a purely mathematical viewpoint. Only a problem with properly set boundary conditions is *well-posed* (i.e., consistent and having a unique solution).

According to our discussion in Chapter 2, the physical boundary conditions are usually expressed in terms of the boundary values of the unknown field u or its normal derivative. Mathematically, the possibilities are: the Dirichlet boundary condition:

$$u(\boldsymbol{x},t)|_{\partial\Omega} = g \text{ at } t_0 < t < t_{\text{end}}, \tag{3.11}$$

the Neumann boundary condition

$$\left.\frac{\partial u(\boldsymbol{x},t)}{\partial n}\right|_{\partial\Omega} = g \text{ at } t_0 < t < t_{\text{end}}, \tag{3.12}$$

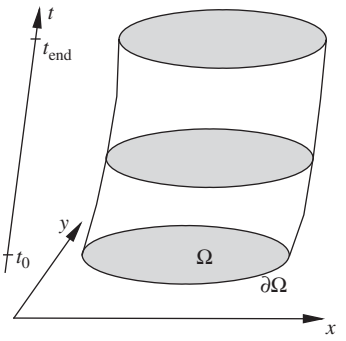


Figure 3.1 Solution domain.

the Robin (mixed) boundary condition

$$\left(a_1 \frac{\partial u(\boldsymbol{x},t)}{\partial n} + a_2 u(\boldsymbol{x},t)\right)\Big|_{\partial\Omega} = g \text{ at } t_0 < t < t_{\text{end}}, \tag{3.13}$$

and the periodicity condition

$$u(\boldsymbol{x},t)|_{x_0} = u(\boldsymbol{x},t)|_{x_0+L} \text{ at } t_0 < t < t_{\text{end}}. \tag{3.14}$$

In equations (3.11)–(3.14), g is a known function of space and time defined at the boundary, and L is the length of periodicity, which we, as an example, assume to be in the x-direction. It is mathematically allowed and sometimes required by the physics that the boundary conditions of different types are applied at different parts of the boundary.

If the domain Ω is infinite, special boundary conditions have to be applied at infinity. For example, if the domain extends to ∞ in the x-direction, the conditions may be

$$u \to 0 \quad \text{or} \quad u \to A = const \quad \text{or} \quad u \text{ is bounded} \quad \text{at } x \to \infty. \tag{3.15}$$

In the problems where the solution is a function of time, *initial conditions* have to be imposed at $t = t_0$. Depending on the type of the equation, one or two conditions are necessary. The most common situations are when the solution itself is known:

$$u(\boldsymbol{x},t_0) = h(\boldsymbol{x}) \text{ in } \Omega, \tag{3.16}$$

and when its first time-derivative is known:

$$\frac{\partial u}{\partial t}(\boldsymbol{x},t_0) = f(\boldsymbol{x}) \text{ in } \Omega. \tag{3.17}$$

Among our model equations, the heat equation (3.2), linear convection equation (3.5), Burgers equation (3.9), and generic transport equation (3.10) require the initial condition (3.16), while the wave equation (3.4) needs both (3.16) and (3.17).

3.1.3 Equilibrium and Marching Problems

In principle, all fluid flow and heat transfer processes evolve with time. From the practical viewpoint, it is, however, desirable to classify them into two groups: equilibrium (time-independent) and transient (time-evolving).

The classification is determined by the nature of the process and by the purpose of the analysis. The equilibrium problems appear when our interest is in a *steady* state of the system, where the properties do not significantly change with time. For example, such a situation arises when we want to know the air resistance coefficient of an airplane cruising with constant speed and latitude or the temperature distribution within a bioreactor operating in a steady-state mode. As an approximation, we assume that the distributed properties are functions of space but not time and replace the time derivatives in the governing equations by zeros.

In other cases, the evolution toward the equilibrium state is of interest or the equilibrium state does not exist even as an approximation. Returning to our examples, this would correspond to an airplane taking off or landing, or to a change in the regime of operation of a bioreactor. Full transient equations should be solved in such cases.

There are special situations in which the PDE problem is formulated as transient even though the underlying physical process is steady-state. This is done when the solution is known to evolve in one particular direction in a timelike manner. The coordinate in this direction assumes the role of a new time line. A prominent example of such behavior is the solution of the approximate equations derived for steady-state boundary layer flows. We will further discuss properties of such systems in section 3.2.3.

Among our model equations, the Laplace (3.6) and Poisson (3.7) equations correspond to equilibrium problems, while the heat equation (3.1), wave equation (3.3), linear convection equation (3.5), Burgers equation (3.9), and generic transport equation (3.10) describe transient evolution of time-dependent systems.

CFD can be applied to both kinds of processes, but different numerical approaches are required. For equilibrium problems, the equation has to be solved numerically only once to determine an approximation to the time-independent solution $u(\boldsymbol{x})$ in Ω. For transient processes, the so-called *marching problems* must be solved. Starting with initial conditions, which give the state at $t = t_0$, the solution is advanced or *marched* forward in time to determine approximations to $u(\boldsymbol{x}, t)$ at $t > t_0$.

3.1.4 Examples

Let us illustrate the formulation of PDE problems using simple examples.

One-dimensional Heat Equation: We invoke the classical example and consider (3.2) as an equation describing the temperature distribution in a thin long rod with thermally insulated sidewalls (see Figure 3.2).

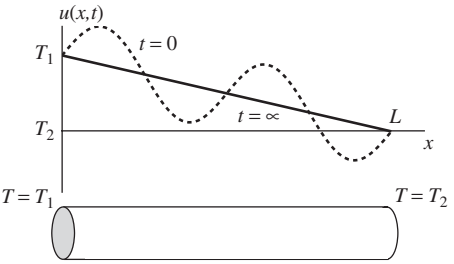


Figure 3.2 Temperature distribution in an one-dimensional rod as an example of a physical system described by the heat equation (3.2).

In this case, we disregard temperature variations across the rod and assume that the temperature T is a function of the coordinate x and time t.

The solution domain consists of the space interval $[0, x]$ and the time interval, which can be finite $[t_0, t_{\text{end}}]$ or extended to infinity $[t_0, \infty)$. Different kinds of boundary conditions are possible. The situation when the ends of the rod are kept at constant temperature corresponds to the Dirichlet boundary conditions

$$u(0,t) = a_0, \quad u(L,t) = a_1 \text{ at } t > t_0. \tag{3.18}$$

Constant heat flux at the ends is described by the Neumann boundary conditions:

$$\frac{\partial u}{\partial x}(0,t) = a_0, \quad \frac{\partial u}{\partial x}(L,t) = a_1 \text{ at } t > t_0. \tag{3.19}$$

Periodic boundary conditions are also possible:

$$u(0,t) = u(L,t) \text{ at } t > t_0. \tag{3.20}$$

Initial temperature distribution $u_0(x)$ is used for the initial conditions:

$$u(x,t_0) = u_0(x) \text{ at } 0 < x < L. \tag{3.21}$$

The complete PDE problem is of marching type and includes the PDE (3.2), the computational domain, one boundary condition, such as (3.18), (3.19), or (3.20) on each end, and the initial condition (3.21).

Laplace Equation: The Laplace equation (3.6) can be obtained as an equation that describes a steady-state temperature distribution in a domain Ω. For example, let us assume that we consider a heat transfer problem in a body with fixed boundary temperature or fixed boundary heat flux and

are not interested in transients. We only want to know the final equilibrium distribution of temperature. Setting the time-derivative of temperature to zero transforms (3.1) into (3.6). The boundary conditions can be $u|_{\partial\Omega} = g$ or $\partial u/\partial n|_{\partial\Omega} = g$.

If internal heat sources are present within Ω, their intensity being defined by the function $f(\boldsymbol{x})$, the final steady-state temperature distribution is a solution of the Poisson equation (3.7).

Another situation described by the Laplace equation is the irrotational flow, in which velocity is a gradient of a scalar potential $\boldsymbol{V} = \nabla\phi(\boldsymbol{x})$. If the fluid is incompressible, the continuity equation becomes

$$\nabla \cdot \boldsymbol{V} = \nabla^2\phi = 0.$$

It should be stressed that, in this case, the Laplace equation does not imply the steady-state character of the process. An important example of a similar situation is the behavior of pressure in incompressible flows. As discussed in Chapter 10, the pressure field satisfies a Poisson equation with a time-dependent right-hand side.

Despite the fact that it can describe time-dependent behavior, the Laplace or Poisson PDE problem is formally of equilibrium type. It consists of the equation (3.6), domain Ω, and one boundary condition (Dirichlet or Neumann type) at each point of the boundary.

One-dimensional Wave Equation: As derived in the textbooks of applied mathematics, the shape of a one-dimensional perfectly elastic string is a solution of the one-dimensional wave equation (3.4). The string motion is frictionless and limited to the x-y plane. The displacement from the line $y = 0$ is defined as $y = u(x,t)$ (see Figure 3.3). The solution domain includes the space interval $[0, L]$ and the time interval $[t_0, t_{\text{end}}]$. The boundary conditions at $x = 0$ and $x = L$ can be, for example, those of a fixed end

$$u(0,t) = u_0 \text{ at } t > t_0,$$

or a freely moving end

$$\frac{\partial u}{\partial x}(L,t) = 0 \text{ at } t > t_0.$$

The initial conditions must include both the shape and velocity of the string at $t = t_0$:

$$u(x,0) = f(x), \quad \frac{\partial u}{\partial t}(x,0) = g(x) \text{ at } 0 < x < L. \tag{3.22}$$

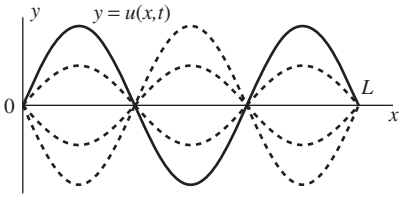


Figure 3.3 Oscillations of a one-dimensional elastic string described by the wave equation (3.4).

The PDE problem is of transient type. A correct formulation includes the equation (3.4), the solution domain, one boundary condition at each boundary, and the two initial conditions (3.22).

3.2 MATHEMATICAL CLASSIFICATION OF PDE OF SECOND ORDER

Partial differential equations of fluid dynamics and heat transfer belong to the class of *quasilinear* PDE, which means that they are linear in their highest-order derivatives, but perhaps not in other terms. Inspection of the governing equations in Chapter 2 and of the model equations in the previous section shows that this is, indeed, the case. The quasilinear PDE can be classified into three types according to the existence and form of their *characteristics*, the special lines in the solution domain. As usual, we leave the exact definition and detailed discussion to specialized texts, this time on the mathematical theory of PDE. One aspect is, however, very important for us: The information in the solutions tends to propagate along the characteristics if they exist. This has deep implications not only for the mathematical properties of the solution but also for the choice of numerical methods. To put it simply, *different numerical methods must be used for equations of different types*.

3.2.1 Classification

The classification is applicable to a broad range of systems of quasilinear PDE. For simplicity, we will limit the discussion to a single linear equation of second order for a function of two variables $\phi(x, y)$. The most general form of such an equation is

$$A\phi_{xx} + B\phi_{xy} + C\phi_{yy} + D\phi_x + E\phi_y + F\phi = G, \qquad (3.23)$$

where A, B, C, D, E, F, and G are known coefficients that can be either constants or functions of x and y. Our choice of (3.23) is not as arbitrary as it may seem. Many equations of fluid dynamics and heat transfer are of the second order, for example, the Navier-Stokes equations (2.17) or the heat transfer equation (2.27). Their highest-order derivatives have the same general form as the highest-order derivatives of (3.23). The lower-order terms are quite different, but, as we will learn imminently, they are of little importance for the classification.

The characteristics of (3.23) can be defined as curves, on which the second derivatives ϕ_{xx}, ϕ_{yy}, and ϕ_{yy} are not uniquely determined by the other terms of the equation. It can be shown that, if a characteristic curve $y = y(x)$ exists, its slope is given by

$$h(x) = \frac{dy}{dx} = \frac{B \pm \sqrt{B^2 - 4AC}}{2A}. \tag{3.24}$$

The classification is based on this relation, more specifically, on the value of $B^2 - 4AC$. The characteristics at the point (x, y) can be of three possible forms:

1. $B^2 - 4AC > 0$. There are two real characteristics intersecting at this point. The equation is called *hyperbolic*.
2. $B^2 - 4AC = 0$. There is one real characteristic. The equation is called *parabolic*.
3. $B^2 - 4AC < 0$. No real characteristics exist at this point. The equation is called *elliptic*.

If A, B, and C are constants, the classification holds in the entire domain Ω. If A, B, or C are functions of x and y, the classification must be done separately for each point (x, y). The equation may be of a different type in different parts of Ω.

Examples Let us determine the types of the model equations considered in the previous section. For the one-dimensional heat equation (3.2), we replace t by y and obtain $A = a^2$, $B = 0$, and $C = 0$, so $B^2 - 4AC = 0$ and the equation is parabolic. For the one-dimensional wave equation (3.4), the same substitution gives $A = a^2$, $B = 0$, and $C = -1$, which gives $B^2 - 4AC = 4a^2 > 0$, so the equation is of hyperbolic type. The two-dimensional Laplace and Poisson equations (3.6) and (3.7) have $A = 1$, $B = 0$, $C = 1$, and $B^2 - 4AC = -4 < 0$. Both the equations are elliptic.

The situations are more complicated for the remaining model equations. The linear convection equation (3.5) is of the first order and does not belong to our classification. Nevertheless, as discussed in section 3.2.2, it has essential features of a hyperbolic system. The classification of the Burgers equation (3.9) and the generic transport equation (3.10) depends on the value of the coefficient μ. At $\mu > 0$, the equations are parabolic. If $\mu = 0$, they look as variations of the linear convection equation, in which the constant c is replaced by a variable, and can be considered hyperbolic. We will discuss this matter further when we consider the hyperbolic equations.

Classification can also be applied to systems of linear or quasilinear PDE of the first order. The simplest example is the system

$$\frac{\partial \boldsymbol{q}}{\partial t} + \boldsymbol{R}\frac{\partial \boldsymbol{q}}{\partial x} = 0, \tag{3.25}$$

where $\boldsymbol{q}(x,t)$ is an unknown vector function and $\boldsymbol{R}(x,t)$ is a square matrix of known coefficients. The system is classified as hyperbolic if all the eigenvalues of $\boldsymbol{R}$ are real and distinct. The eigenvalues determine the slopes of the characteristic curves. Another identifiable case is when all the eigenvalues are complex. Such systems are called elliptic.

The classification of first-order systems plays an important role in gas dynamics, acoustics, and other fields, where the governing equations are expressed in the form of such systems. Further details can be found in the books listed at the end of the chapter. Here, we only mention that the two classifications produce identical results if they are applied to mathematically identical cases. For example, the one-dimensional wave equation (3.4) and the Laplace equation (3.6) can be rewritten as two-equation systems (3.25). Analyzing the eigenvalues we find that the systems should be classified as hyperbolic and elliptic, respectively. This is in full agreement with the classifications of the original PDEs of second order.

3.2.2 Hyperbolic Equations

To illustrate typical properties of hyperbolic systems, we will analyze solutions of the one-dimensional wave equation (3.4). From (3.24), we find the slope of the characteristics as

$$h = \frac{dt}{dx} = \frac{0 \pm \sqrt{0 + 4a^2}}{2a^2} = \pm\frac{1}{a}.$$

There are two families of characteristics: left-running $x + at = const$ and right-running $x - at = const$.

It can be shown that the general solution of (3.4) can be represented as

$$u(x,t) = F_1(x + at) + F_2(x - at), \tag{3.26}$$

where F_1 and F_2 are functions determined by initial and boundary conditions. If we ignore the effect of boundaries, the solution for the specified initial conditions $u(x,0) = f(x)$, $\partial_t u(x,0) = g(x)$ can be written explicitly in the d'Alembert form

$$u(x,t) = \frac{f(x+at) + f(x-at)}{2} + \frac{1}{2a}\int_{x-at}^{x+at} g(\tau)d\tau. \tag{3.27}$$

The first part of the solution can be interpreted using the illustration in Figure 3.4a. The initial perturbation of $u(x,0) = f(x)$ around the point x_0 (e.g., a localized deformation of a string) is split into halves, which propagate without changing their shape along the characteristics $x + at = x_0$ and $x - at = x_0$.

The second part of the solution (3.27) represents the response to the initial perturbation of 'velocity' $\partial_t u$. If, for example, the initial velocity is a delta function $g(x) = \delta(x - x_0)$, the solution evolves as illustrated in Figure 3.4b. $u(x,t)$ is a constant equal to $1/2a$ within the cone between the left-running and right-running characteristics and zero outside this cone.

An important feature of the hyperbolic systems is illustrated by (3.27). The perturbations propagate in space with a finite speed. Let us consider the situation, when a source of perturbations suddenly appears at the time moment t_0 and space location x_0 (point P in Figure 3.5). An observer located at the distance L from the source will not notice the perturbations

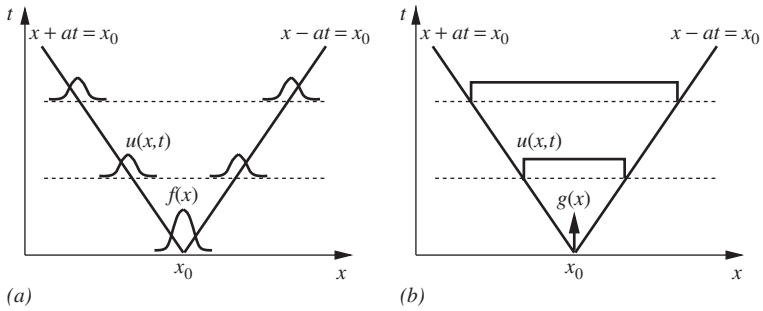


Figure 3.4 D'Alembert solution (3.27) of the hyperbolic equation (3.4): *(a)* effect of the initial perturbation of u; *(b)* effect of the initial perturbation of $\partial_t u$.

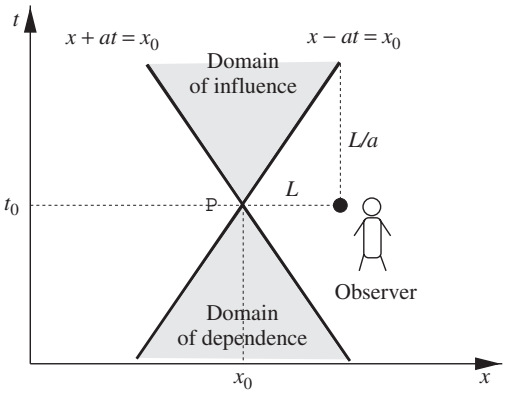


Figure 3.5 Characteristics and domains of influence and dependence of the hyperbolic equation (3.4).

until the time $t_0 + L/a$. In general, the state of the solution at the point P only affects the solution within a cone between the left-running and right-running characteristics intersecting at P. The cone is called the *domain of influence*. Similarly, the solution at P itself is affected only by the solution within the *domain of dependence* (see Figure 3.5).

The behavior described by hyperbolic equations and, thus, determined by characteristics appears in many physical systems—for example, in supersonic flows, propagation of acoustic and electromagnetic waves, and flows in stratified systems such as waves on water surface of internal waves in the ocean. All these processes involve wavelike motions along the characteristics or discontinuities across them (e.g., shock waves in supersonic flows). The Navier-Stokes equations have some features of a hyperbolic system due to the presence of the nonlinear (material derivative) terms.

We now return to the linear convection equation (3.5) and explain why it was classified as hyperbolic. It is easy to check by direct substitution that the equation has the exact solution in the form of a wave propagating along the right-running characteristic

$$u(x,t) = F(x - ct), \tag{3.28}$$

where F is a function determined by the initial and boundary conditions.

In the case of zero diffusivity coefficient μ, the Burgers and generic transport equations become

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0 \tag{3.29}$$

and

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} = 0. \tag{3.30}$$

Their only difference from the linear convection equation is that the constant coefficient c is replaced by the unknown solution in the case of (3.29) or by a variable coefficient in the case of (3.30). This does not change the hyperbolic, characteristic-determined nature of the solution, except that now the characteristics are not straight lines but curves in the (x, t) space.

3.2.3 Parabolic Equations

The parabolic equations of second order have only one family of real characteristics. For example, for the one-dimensional heat equation (3.1), the slope is

$$h = \frac{dt}{dx} = \frac{0 \pm \sqrt{0+0}}{2a^2} = 0.$$

The characteristics are lines $t = const$ (see Figure 3.6a). The perturbation that occurs at the space location x_0 and time moment t_0 (point P in Figure 3.6a) affects the solution in the entire space domain $0 < x < L$, although the effect becomes weaker with the distance to P. The domain of influence of the point P includes the domain $0 < x < L$ and times $t > t_0$. Accordingly, the domain of dependence of P includes all points $0 < x < L$ and all moments of time prior to the time of P.

In the physical systems described by parabolic equations, the perturbations are usually propagated by *diffusion*. The interaction occurs at infinite

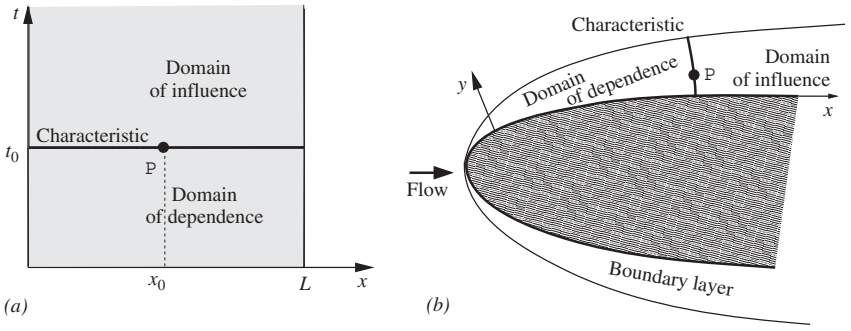


Figure 3.6 *(a)* Characteristics and domains of influence and dependence of the parabolic equation (3.2); *(b)* Parabolic behavior of viscous boundary layer.

speed but relaxes with distance. For example, in the problem of conduction heat transfer in a one-dimensional rod, increase of temperature at a single point is initially felt only slightly at other points (weaker signal for larger distances). With time, however, the effect gradually becomes stronger and the nonuniformity of the temperature field is diffused or smoothed out.

Similar diffusive processes occur in other physical systems. For example, viscous terms in the Navier-Stokes equations (2.17) lead to diffusion of gradients of the velocity field, thus giving the equations parabolic properties. The nonlinear parts of the material derivatives and the pressure effects can be neglected for some flows. The remaining equations such as

$$\frac{\partial u}{\partial t} = \frac{\mu}{\rho}\nabla^2 u$$

are of purely parabolic type.

Yet another example of the parabolic behavior is the steady-state viscous boundary layers. The reduced Navier-stokes equations that describe the flow within a boundary layer are of the parabolic type. The characteristics are perpendicular to the wall, and the flow evolves along the boundary layer similarly to the time evolution of solutions of other parabolic systems (see Figure 3.6b).

3.2.4 Elliptic Equations

The elliptic equations do not have real characteristics at all. Effect of any perturbation is felt immediately and to full degree in the entire domain of solution. There are no limited domains of influence or dependence.

As opposite to the hyperbolic and parabolic systems that involve time evolution (or evolution along a spatial direction as in the case of a boundary layer flow) and have to be treated as marching problems in CFD, the elliptic PDE problems are always of equilibrium type. The solution has to be found at once in the entire domain Ω.

Physically, the elliptic systems describe equilibrium distributions of properties in spatial domains with boundary conditions. As examples, we name the steady-state heat transfer, electrostatics, and irrotational fluid flows. We will also learn in Chapter 10 that pressure field in incompressible flows is a solution of an elliptic Poisson equation.

3.3 NUMERICAL DISCRETIZATION: DIFFERENT KINDS OF CFD

In the rest of the book, we shall assume that the analytical solution of a PDE is unavailable or cannot be used for some reason. The equation has

to be solved numerically. This section introduces the key concept of the numerical solution—the concept of *discretization*. We also briefly review the main discretization techniques.

Discretization can be understood as replacement of an *exact* solution of a PDE or a system of PDEs in a *continuum* domain by an *approximate* numerical solution in a *discrete* domain. Instead of continuous distributions of solution variables we find a finite set of numerical values that represent an approximation of the solution.

The discretization can be implemented in different ways. The majority of the methods used in the computational fluid dynamics and heat transfer follow one of the three general approaches: finite difference, finite element, or spectral.

3.3.1 Spectral Methods

Conceptually, the spectral methods are similar to the technique of separation of variables used to solve PDE analytically. The solution is represented by a series of linearly independent and, in many cases, orthogonal functions (cos, sin, Bessel functions, orthogonal polynomials, etc.) with unknown coefficients. The principal difference is that the series is infinite in the analytical method, and we try to find the coefficients such that the series converges to the exact solution of the PDE. In the numerical spectral method, the series is finite and the coefficients are chosen so that they *minimize the error of approximation*.

Let us consider a simple example. The modified one-dimensional heat equation

$$\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2} + \sin 5x \tag{3.31}$$

is solved at $0 < x < \pi$ and $0 < t < T$ with the boundary conditions $u(0,t) = u(\pi,t) = 0$ and initial conditions $u(x,0) = x(\pi - x)$. We will use the Galerkin method, which is a version of the general spectral approach. First, we have to find a set of orthogonal functions that satisfies the boundary conditions (the so-called trial functions). An obvious possibility is

$$\sin x, \sin 2x, \sin 3x, \ldots$$

The solution is sought in the form of a series of trial functions

$$\tilde{u}(x,t) = \sum_{n=1}^{N} A_n(t) \sin nx, \tag{3.32}$$

where we use the notation $\tilde{u}$ for the numerical approximation in order to distinguish it from the exact solution u. The number of trial functions N determines the accuracy of the solution. Substituting (3.32) into the equation (3.31), we obtain the error of approximation

$$\epsilon(x,t) = \frac{\partial \tilde{u}}{\partial t} - a^2 \frac{\partial^2 \tilde{u}}{\partial x^2} - \sin 5x = -\sin 5x + \sum_{n=1}^{N} \left[A_n' + (na)^2 A_n\right] \sin nx. \tag{3.33}$$

In order to minimize ϵ, we require that its projection on the functional subspace spanned by N orthogonal functions (the *test functions*) is zero. In other words, we require that the inner product of the error with each test function is zero:

$$\langle \epsilon, \phi_m \rangle = \int_0^{\pi} \epsilon(x,t), \phi_m(x) dx = 0, \quad m = 1, \ldots, N. \tag{3.34}$$

In the Galerkin method, the same set of functions serves as a test and trial set (different approach may be taken by other spectral methods). Using the orthogonality

$$\int_0^{\pi} \sin nx \sin mx dx = \begin{cases} 0 & \text{if} \quad m \neq n \\ \pi/2 & \text{if} \quad m = n, \end{cases}$$

we obtain, from (3.33) and (3.34), a system of N ordinary differential equations for $A_n(t)$:

$$A_n' = -(na)^2 A_n + \delta_{5n}, \quad n = 1, \ldots, N. \tag{3.35}$$

The initial conditions are found by multiplying each side of the expression $\tilde{u}(x,0) = x(\pi - x)$ by $\sin nx$ and integrating from 0 to π. The results is

$$A_n(0) = \frac{2}{\pi} \int_0^{\pi} x(\pi - x) \sin nx dx = -\frac{4(\cos(n\pi) - 1)}{\pi n^3}, \; n = 1, \ldots, N.$$

The system can be solved numerically using any of the methods developed for ordinary differential equations—for example, the Runge-Kutta method.

The spectral methods have one strong advantage over the other numerical methods. Their approximation error reduces quickly with increase of the number of expansion terms N. Typically, a moderate number of trial functions is sufficient to achieve good accuracy.

Naturally, there are also bad news. One of them is that the series such as (3.32) can only be developed for multidimensional PDE if the solution

domain has simple geometry, such as a rectangle, sphere, or cylinder. This strongly limits the application of the method to practical engineering problems, which typically have complex, three-dimensional geometry. On the contrary, the fundamental research problems can often be simplified to model geometries. It is, therefore, not surprising that the spectral methods have found widespread use in the theoretical fluid mechanics, especially in investigations of turbulence. We will show an example in section 11.2.

3.3.2 Finite Element Methods

The finite element methods are widely applied in many areas of engineering—for example, in structural analysis and conduction heat transfer. They are also used for simulating fluid flows, although not as widely as the finite difference and finite volume methods. The basic concept is similar to that of the spectral methods. The main difference is that the decomposition (3.32) is done not in the entire solution domain but within each of many small elements, into which the domain is subdivided. A small number of trial functions is used. The functions are chosen so that they are zero outside the element. Projecting the residuals in a manner similar to the Galerkin procedure and performing summation over all elements results in a system of ordinary differential equations not unlike (3.35).

3.3.3 Finite Difference and Finite Volume Methods

The focus of this book is on the finite difference approach (which also includes the finite volume methods in our classification). The discretization is achieved by approximating the continuous solution at discrete grid points and time layers of a *computational grid*.

The general strategy is summarized in Table 3.1 and illustrated in Figure 3.7. The domain of solution Ω is covered by a grid of points with coordinates $(x, y, z)_i$, where the index i is used to number the points. If a marching problem is solved, the time range $[t_0, t_{\text{end}}]$ should also be covered by discrete points (time layers) t^n.

Our goal is to find the set of numbers u_i^n that approximates the exact solution $u(\boldsymbol{x}, t)$ at points $(x, y, z)_i$ and layers t^n. This is achieved by approximating the original PDE using *finite differences* and solving the resulting system of algebraic equations.

We will discuss in the following chapters how the finite difference grids are designed and how the equations are approximated by finite difference

Table 3.1 Relation between Elements of a Mathematical PDE Problem and Its Finite Difference Approximation

PDE Problem	Finite Difference Approximation
Partial differential equation	System of algebraic discretization equations
Exact analytical solution	Approximate finite difference solution
Domain Ω and time-interval $[t_0, t_1]$	Computational grid—set of points $(x, y, z)_i$ and time layers t^n
Exact solution—function $u(x, y, z, t)$	Approximate solution—set of values u_i^n approximating u at $(x, y, z)_i$ and t^n

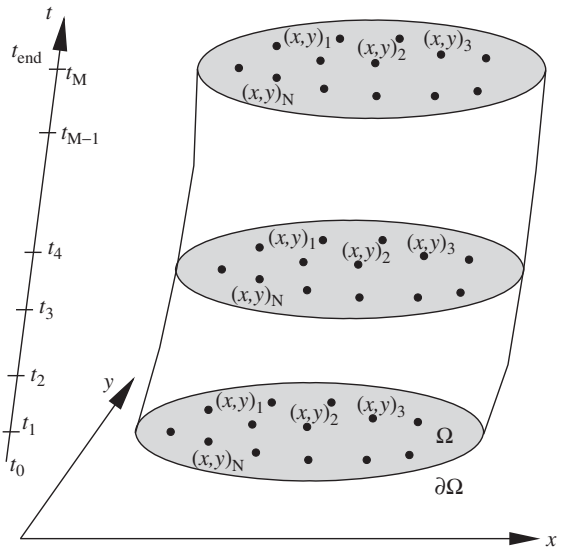


Figure 3.7 Finite difference discretization of the solution domain.

formulas. For now, as a preliminary example, we consider the same PDE problem as in the discussion of spectral methods in section 3.3.1:

$$\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2} + \sin 5x, \quad 0 \le x \le \pi, \quad 0 < t < T,$$

$$u(0,t) = u(\pi,t) = 0, \quad u(x,0) = x(\pi - x).$$

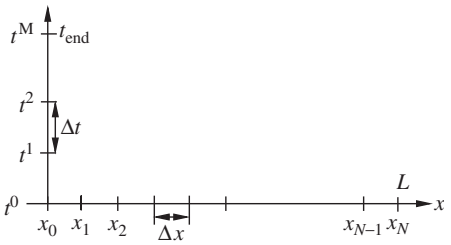


Figure 3.8 Computational grid used to solve the one-dimensional heat equation.

The computational grid is illustrated in Figure 3.8. It includes $N+1$ equally spaced points $x_i = i\,\Delta x$ with $i = 0, 1, 2, \ldots, N$ and $\Delta x = L/N$, and $M+1$ equally spaced time layers $t^n = n\,\Delta t$ with $n = 0, 1, 2, \ldots, M$ and $\Delta t = t_{end}/M$. Note that there is a grid point at each boundary $x_0 = 0$ and $x_N = \pi$ and a time layer $t^0 = 0$ at the initial moment of time.

The terms of the PDE can be approximated, for example, as

$$\left.\frac{\partial u}{\partial t}\right|_{x_i,t^n} \approx \frac{u_i^{n+1} - u_i^n}{\Delta t}, \quad \left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i,t^n} \approx \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{(\Delta x)^2}. \tag{3.36}$$

The approximations (3.36) are substituted for the terms of the PDE at every inner discretization point. The result is

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = a^2 \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{(\Delta x)^2} + \sin 5x_i \tag{3.37}$$

for $i = 1, 2, \ldots, N-1$ and $n = 1, 2, \ldots, M-1$. The boundary and initial conditions are approximated as

$$u_0^n = 0, \quad u_N^n = 0 \text{ at } n = 1, 2, \ldots, M, \tag{3.38}$$

$$u_i^0 = x_i(\pi - x_i) \text{ at } i = 0, 1, \ldots, N. \tag{3.39}$$

(3.37)–(3.39) represent a system of linear algebraic equations. It is important to realize that the equations are coupled with each other. For example, the finite difference equations (3.37) written for the four adjacent points (x_i, t^{n-1}), (x_i, t^n), (x_{i-1}, t^n), (x_{i+1}, t^n) have a common unknown u_i^n. The boundary and initial conditions are coupled with the finite difference equations at the internal points next to the boundary. The result of the coupling is that, in general, all the finite difference equations have to be solved simultaneously as an algebraic system. One of the tasks of CFD is

to formulate such systems in a way that allows efficient solution. This is easy in the case of (3.37)–(3.39). Regrouping the terms in (3.37) as

$$u_i^{n+1} = u_i^n + \frac{a^2 \Delta t}{(\Delta x)^2} \left(u_{i-1}^n - 2u_i^n + u_{i+1}^n \right)$$

and using the boundary conditions (3.38), we can calculate all the values u_i^{n+1} at the $(n+1)$st layer, provided the values u_i^n at the nth layer have already been found. We already have the values of u_i^0 at the zeroth time layer from (3.39). Starting with it, we can advanced to the layer t^1, then t^2, and so on in a *marching procedure* and solve the problem in no time.

REFERENCES AND SUGGESTED READING

Canuto, C., M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods: Fundamentals in Single Domains*. Berlin Heidelberg: Springer, 2006.

Fletcher, C. A. J. *Computational Techniques for Fluid Dynamics*. Vol. 1, *Fundamental and General Techniques*. (2nd ed.) Berlin: Springer-Verlag, 1991.

Greenberg, M. *Advanced Engineering Mathematics* (2nd ed.). Englewood Cliffs, NJ: Prentice Hall, 1998.

Haberman, R. *Applied Partial Differential Equations* (4th ed.). Upper Saddle River, NJ: Prentice Hall, 2003.

Leveque, R. J. *Finite Volume Methods for Hyperbolic Problems*. Cambridge, UK: Cambridge University Press, 2002.

Logan, D. L. *A First Course in the Finite Element Method*. Florence, KY: Cengage-Engineering, 2006.

Tannehill, J. C., D. A. Andersen, and R. H. Pletcher, *Computational Fluid Mechanics and Heat Transfer* (2nd ed.). Philadelphia: Taylor & Francis, 1997.

PROBLEMS

1. Formulate complete PDE problems (specify the equation, space domain, time interval, boundary, and initial conditions) for the following model situations:

a) Conduction heat transfer occurs in a thin rod of length L with insulated side walls (see Figure 3.2). Temperature is initially constant $T(0) = T_0$. We are asked to find the temperature distribution in the time period $0 < t < t_1$, during which the left

end of the rod is kept at the temperature T_0, and the right end is subject to cooling with the constant heat transfer rate q.

b) We are asked to find equilibrium temperature distribution in a rectangular metal plate $0 \le x \le L_x, 0 \le y \le L_y$. The boundaries $x = 0$ and $x = L_x$ are kept at constant temperatures T_1 and T_2. The boundaries $y = 0$ and $y = L_y$ are thermally insulated.

c) One-dimensional string stretched between the points $(x, y) = (0, 0)$ and $(x, y) = (L_x, 0)$ oscillates elastically during the time period $0 < t < t_1$ (see Figure 3.3). At the initial moment $t = 0$, the deviation of the string from the horizontal and its velocity are given by function $f(x)$ and $g(x)$, respectively.

2. What are the characteristics of a quasilinear equation of second order? How are they determined?

3. Classify (determine the type of) the following PDEs of second order:

$$\frac{\partial^2 u}{\partial x \partial y} = 0$$

$$(x^2 - 1)\frac{\partial^2 u}{\partial x^2} + 2\frac{\partial^2 u}{\partial y^2} = 25(x^3 - 1)\frac{\partial u}{\partial x}$$

$$\frac{\partial^2 u}{\partial t^2} + \frac{\partial^2 u}{\partial x^2} + \frac{\partial u}{\partial x} = \cos(5t)$$

4. Transform the one-dimensional wave equation (3.4) into a system of PDE of the first order (3.25). Find the eigenvalues of matrix $\boldsymbol{R}$ and determine the type of the system. Does it agree with the type of (3.4) as an equation of second order? *Hint*: Introduce new variables $v = \partial u/\partial t$ and $w = a\partial u/\partial x$

5. Conduct the same analysis as in the previous problem, but for the Poisson equation (3.8).

6. Define the domain of dependence and domain of influence of a point P in a solution of a PDE? How are these domains determined for each type of the PDE of second order?

7. Verify coupling of the linear algebraic equations of the system (3.37)–(3.39) by writing the discretization formulas (3.37) for the interior grid points (x_i, t^n), (x_{i+1}, t^n), and (x_i, t^{n-1}) and the points (x_0, t^n) and (x_1, t^n) near the boundary.

Programming Exercise Develop the algorithm for solution of the heat equation problem (3.31) using the Galerkin spectral method. Use $a = 0.5$ and $T = 50$. Apply the Runge-Kutta scheme of the second order with the time step $\Delta t = 0.02$ to solve the ordinary differential equations for the coefficients (3.35). Compute the solution 5, 10, 50, and 100 test and trial functions. Compare the results. Analyze, how the amplitudes of the coefficients $A_n(t)$ change with n and t.

4

BASICS OF FINITE DIFFERENCE APPROXIMATION

The simple example at the end of the last chapter only illustrates the finite difference method. There is much more to the method than a few simple formulas. We begin a thorough discussion in this chapter. The goal is to introduce the main concepts and answer the questions:

- How do we construct the finite difference schemes?
- How close is the numerical approximation to the exact solution?
- How can we reliably achieve desired accuracy of the approximation?

4.1 COMPUTATIONAL GRID

The first step toward a finite difference approximation is to cover the solution domain together with its boundaries by a computational grid.

4.1.1 Time Discretization

The grid covering the time interval $[t_0, t_{\text{end}}]$ of the solution consists of time layers $t^n = t_0 + n\Delta t$. They can be distributed *uniformly* that is with constant time step Δt, or *nonuniformly*, with varying Δt. The second approach is called the *variable time step* method and can be applied to accelerate solution of marching problems by increasing Δt, where possible. We stress the word *possible* since, as discussed in this and the following chapters, there are upper limits on Δt set by requirements of accuracy and numerical stability.

4.1.2 Space Discretization

If the space domain Ω is one-dimensional (an interval), the computational grid is a one-dimensional set of points $x_i = x_0 + i\Delta x$, with Δx being either constant (*uniform* grids) of a function of x (*nonuniform*, also called *clustered* or *stretched* grids).

In the multidimensional case (Ω is a two-dimensional or three-dimensional domain), the choice is much richer. We review the main options here, but postpone the thorough discussion until Chapter 12. One can choose between structured and unstructured grids. In an *unstructured* grid, the nodes are placed irregularly (as illustrated in Figure 3.7). Such grids are never used with the classical finite difference methods considered in this chapter. They, however, have become a powerful tool of modern practical CFD analysis in combination with the finite volume discretization, which we discuss in Chapter 5 and later in Chapter 12. The main advantage of the unstructured grids is their geometric flexibility. Any geometric shape can be covered by the grid, with some nodes placed exactly at the boundaries.

The *structured* grids have traditionally been used with finite difference discretization. They are simpler to work with than unstructured grids and, thus, represent a better tool for developing numerical schemes and for learning. In such grids, the nodes are placed at the intersections of the lines of a coordinate system. The nodes of a structured grid do not have to be numbered throughout by one index. Instead, two (in the two-dimensional case) or three (in the three-dimensional case) indices can be assigned to every node, so that each index represents the node's location along the corresponding coordinate. As an example, Figure 4.1 shows two-dimensional structured grids built along the lines of a Cartesian and a polar coordinate systems. The point A in Figure 4.1 numbered as (i, j) has Cartesian coordinates (x_i, y_j) or polar coordinates (θ_i, r_j). The point immediately to the right of A, which has coordinates $(x_i + dx, y_j)$ in part (a) of Figure 4.1, is marked by indices $(i + 1, j)$, etc.

A multidimensional structured grid can be uniform with constant grid steps Δx, Δy, and Δz or nonuniform (clustered), in which the grid steps vary in space.

Throughout this chapter and in much of the following discussion, we predominantly use uniform structured grids. The two-dimensional rectangular Cartesian grid shown in Figure 4.1a is our favorite. The value of a function u at point (x_i, y_j) and time layer t^n is written as

$$u_{i,j}^n \approx u(x_i, y_j, t^n), \tag{4.1}$$

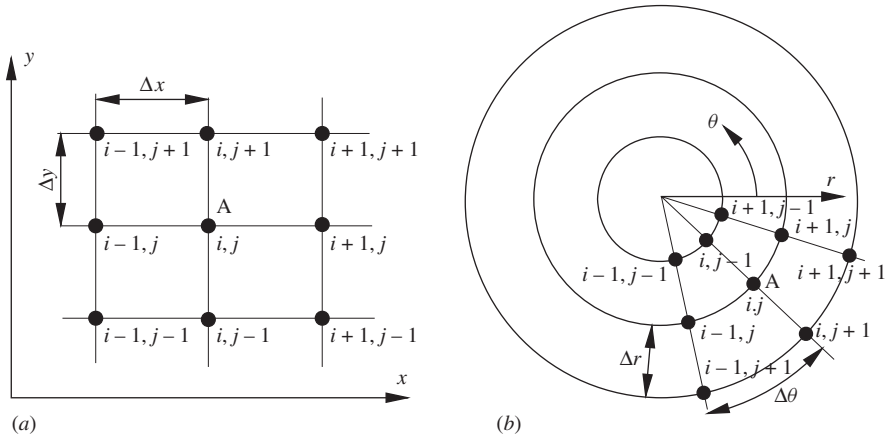


Figure 4.1 Structured discretization grids built along the lines of Cartesian *(a)* and polar *(b)* coordinate systems.

or, if time is not involved, as

$$u_{i,j} = u(x_i, y_j) \tag{4.2}$$

4.2 FINITE DIFFERENCES AND INTERPOLATION

4.2.1 Approximation of $\partial u/\partial x$

The concept of a finite difference approximation of a partial derivative is, in fact, familiar to us from elementary calculus. The partial derivative of a function $u(x, y)$ with respect to x taken at the point (x_0, y_0) is defined as

$$\left.\frac{\partial u}{\partial x}\right|_{x_0,y_0} \equiv \lim_{\Delta x \to 0} \frac{u(x_0 + \Delta x, y_0) - u(x_0, y_0)}{\Delta x}. \tag{4.3}$$

In a numerical solution, we do not have access to the function $u(x, y)$ itself. Only the approximate values of u at the grid points (x_i, y_j) are available. Let $x_0 = x_i$ and $y_0 = y_j$. The closest available approximation of (4.3) is

$$\left.\frac{\partial u}{\partial x}\right|_{i,j} \approx \frac{u(x_{i+1}, y_j) - u(x_i, y_j)}{\Delta x} = \frac{u_{i+1,j} - u_{i,j}}{\Delta x}, \tag{4.4}$$

which, of course, makes sense only if Δx is "sufficiently small."

More rigorously, we have to prove that the expression in the right-hand side of (4.4) approximates the partial derivative and to determine the error of the approximation. This can be easily done using the Taylor series expansion of $u(x, y)$. We shall assume that the function is sufficiently smooth, so the Taylor series converges, and use one-dimensional expansion with respect to x at constant y

$$u_{i+1,j} = u_{i,j} + \left(\frac{\partial u}{\partial x}\right)\Big|_{i,j} \Delta x + \left(\frac{\partial^2 u}{\partial x^2}\right)\Big|_{i,j} \frac{(\Delta x)^2}{2!} + \left(\frac{\partial^3 u}{\partial x^3}\right)\Big|_{i,j} \frac{(\Delta x)^3}{3!} + \cdots + \left(\frac{\partial^{n-1} u}{\partial x^{n-1}}\right)\Big|_{i,j} \frac{(\Delta x)^{n-1}}{(n-1)!} + \left(\frac{\partial^n u}{\partial x^n}\right)\Big|_{\zeta,j} \frac{(\Delta x)^n}{n!}, \tag{4.5}$$

where ζ is some point between x_i and x_{i+1}. We now rearrange the formula, moving the term with the partial derivative $\partial u/\partial x$ into the left-hand side and everything else into the right-hand side and dividing by Δx:

$$\left(\frac{\partial u}{\partial x}\right)\Big|_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{\Delta x} - \left(\frac{\partial^2 u}{\partial x^2}\right)\Big|_{i,j} \frac{(\Delta x)}{2!} - \cdots - \left(\frac{\partial^n u}{\partial x^n}\right)\Big|_{\zeta,j} \frac{(\Delta x)^{n-1}}{n!}. \tag{4.6}$$

The formula provides all the necessary information. We see that the finite difference approximation (4.4) includes only the first term of the expansion (4.6). All the other terms in the right-hand side are dropped. The error of the approximation is the sum of these dropped terms. We also see that the error decreases with decreasing Δx and vanishes in the limit $\Delta x \to 0$.

4.2.2 Truncation Error, Consistency, Order of Approximation

The formula (4.6) can be rewritten as

$$\left(\frac{\partial u}{\partial x}\right)\Big|_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{\Delta x} + \text{T.E.}, \tag{4.7}$$

where we used the notation T.E. for the *truncation error* of discretization of a partial derivative. It contains the rest of the series dropped in the finite difference approximation and, thus, shows the difference between the exact expression (4.6) and the finite difference approximation (4.4). The truncation error is a key characteristic of the accuracy of a finite

difference scheme. Although it cannot be explicitly evaluated without a-priori knowledge of function u, it provides vital information about *how fast the error of approximation decreases with decreasing grid step*.

At this point, we need to review some facts of basic calculus. The notation $f = O(\alpha)$ (f is of the order of α) means that f remains smaller than $K\alpha$ with some constant K as $\alpha \to 0$. In other words, f decreases with α not slower than α itself.

In our case, the small parameter is Δx. Each term of the truncation error is of the order of a certain power of Δx:

$$\left(\frac{\partial^2 u}{\partial x^2}\right)\Bigg|_{i,j} \frac{(\Delta x)}{2!} = O(\Delta x), \left(\frac{\partial^3 u}{\partial x^3}\right)\Bigg|_{i,j} \frac{(\Delta x)^2}{3!} = O((\Delta x)^2),$$

$$\ldots, \left(\frac{\partial^n u}{\partial x^n}\right)\Bigg|_{\zeta} \frac{(\Delta x)^{n-1}}{n!} = O((\Delta x)^{n-1}).$$

As Δx decreases, the higher-order terms become negligible in comparison with the first-order term, and the entire truncation error is evaluated as

$$\text{T.E.} = O(\Delta x).$$

We can rewrite equation (4.6) again, this time as

$$\left(\frac{\partial u}{\partial x}\right)\Bigg|_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{\Delta x} + O(\Delta x). \tag{4.8}$$

We see that only the lowest-order term of the truncation error is important. Its order is called *the order of truncation error* or *the order of approximation of the finite difference scheme*. It defines how fast the error of approximation decreases with the grid step.

Another important concept is that of *consistency* of a finite difference approximation. It is obvious that the approximation makes sense only if the truncation error vanishes at $\Delta x \to 0$; that is, if the scheme has, at least, the first order of approximation. A scheme satisfying this requirement is *consistent*.

One consideration should be taken into account in practical computations, where the limit $\Delta x \to 0$ is never attained. In addition to the order of approximation and the size of grid step, the magnitude of the truncation error is determined by the amplitudes of derivatives $\partial u/\partial x$, $\partial^2 u/\partial x^2$, and so on. This means that the behavior of the approximated function is a factor affecting the accuracy of the approximation. If the function has strong gradients, the amplitudes of derivatives are large and the truncation

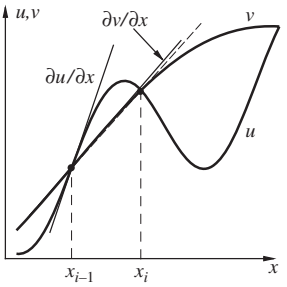


Figure 4.2 Effect of strong gradients of function on truncation error. The finite difference approximation (4.4) (slope of dashed line) accurately reproduces $\partial v/\partial x$ but not $\partial u/\partial x$.

error can be significant even when the consistency is satisfied and Δx is not particularly large.

A simple view on this phenomenon is illustrated in Figure 4.2. On the same computational grid, the derivative of function u characterized by strong variations over small distances is approximated with lower accuracy than the derivative of v. The figure also shows that the rather poorly defined term *small distances* should be understood as "small in comparison with the grid step." It is clear that accurate approximation of derivatives requires a grid with steps smaller—desirably, several times smaller—than the smallest distance on which the function experiences significant variation.

4.2.3 Other Formulas for $\partial u/\partial x$: Evaluation of the Order of Approximation

The *forward difference* formula (4.4) is not the only possible approximation for $\partial u/\partial x$. In fact, infinitely many approximations can be developed. For example, we can use *backward difference*:

$$\left.\frac{\partial u}{\partial x}\right|_{i,j} \approx \frac{u_{i,j} - u_{i-1,j}}{\Delta x} \tag{4.9}$$

or *central difference*:

$$\left.\frac{\partial u}{\partial x}\right|_{i,j} \approx \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}. \tag{4.10}$$

Each of the formulas has clear meaning as an approximation of the slope of the curve $u(x, y = const)$. It is obvious from the illustration in Figure 4.3

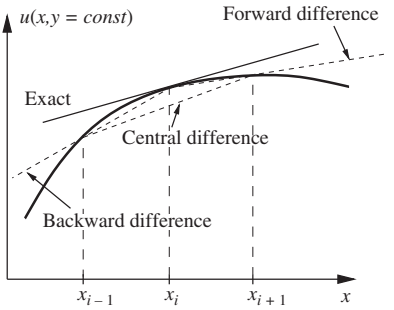


Figure 4.3 Approximation of the partial derivative $\partial u(x,y)/\partial x$ by finite difference formulas (4.4), (4.9), and (4.10).

that the approximations approach the exact value of the slope as $\Delta x \to 0$. In some other cases, a graphic representation is insufficient. We need a formal procedure that verifies that the finite difference formula is consistent. We also have to determine the order of approximation. The best way to do it is to apply the Taylor series expansions around the point (x_i, y_j) at which the derivative is approximated. For example, we can substitute

$$u_{i-1,j} = u_{i,j} - \left.\frac{\partial u}{\partial x}\right|_{i,j} \Delta x + \frac{1}{2!} \left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} (\Delta x)^2 + O((\Delta x)^3)$$

into (4.9) to obtain

$$\left.\frac{\partial u}{\partial x}\right|_{i,j} = \frac{u_{i,j} - u_{i-1,j}}{\Delta x} + O(\Delta x). \tag{4.11}$$

Substitution of the same expression for $u_{i-1,j}$ and

$$u_{i+1,j} = u_{i,j} + \left.\frac{\partial u}{\partial x}\right|_{i,j} \Delta x + \frac{1}{2!} \left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} (\Delta x)^2 + O((\Delta x)^3)$$

into (4.10) results in incidental cancelation of the first-order term (proportional to $(\partial^2 u/\partial x^2)\big|_{i,j}$) and in the formula

$$\left.\frac{\partial u}{\partial x}\right|_{i,j} = \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + O((\Delta x)^2). \tag{4.12}$$

Let us see what we have. First, all three schemes—forward, backward, and central differences—are consistent since the truncation errors vanish as $\Delta x \to 0$. Second, the schemes have different orders of approximation:

first order for the forward and backward differences (4.4) and (4.9) and second order for the central difference (4.10). This is of fundamental importance, since it shows that the T.E. of the central difference scheme decreases faster with Δx than the T.E. of the backward and forward schemes. For example, if Δx is reduced from 10^{-2} to 10^{-3}, the error decreases by a factor of 10 for a first-order approximation, but by a factor of 100 for a second-order approximation. The exact evaluation of T.E. is rarely possible, since the values of partial derivatives of u included in the T.E. are normally a-priori unknown. We can, however, say that the central difference scheme (4.10) inevitably becomes more accurate than the forward and backward schemes (4.4) and (4.9) at sufficiently small grid steps Δx.

Two other finite difference formulas for the first derivative $\partial u/\partial x$ are

$$\left.\frac{\partial u}{\partial x}\right|_{i,j} = \frac{-3u_{i,j} + 4u_{i+1,j} - u_{i+2,j}}{2\Delta x} + O((\Delta x)^2) \tag{4.13}$$

and

$$\left.\frac{\partial u}{\partial x}\right|_{i,j} = \frac{3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}}{2\Delta x} + O((\Delta x)^2). \tag{4.14}$$

They are often used at the boundaries, where values of u on only one side of the (i, j) point are available, but the second order of accuracy is desired.

4.2.4 Schemes of Higher Order for First Derivative

Schemes of the order higher than second are possible and, sometimes, desirable. On the one hand, such schemes are accurate. On the other hand, the discretization schemes based on higher-order approximations are more difficult to program and may require larger amount of computations. They also tend to have more stringent numerical stability requirements on the time step (the numerical stability is discussed in Chapter 6).

As an example, the central fourth-order scheme is

$$\left.\frac{\partial u}{\partial x}\right|_{i,j} = \frac{-u_{i+2,j} + 8u_{i+1,j} - 8u_{i-1,j} + u_{i-2,j}}{12\Delta x} + O((\Delta x)^4). \tag{4.15}$$

A separate class of finite difference approximations is the so-called Padé or compact schemes. In these schemes, the approximation of the derivative at (i, j)-point is not explicitly expressed in terms of the values

of u at neighboring nodes, but is found as a part of the solution of a system of coupled linear algebraic equations. As an example, we give the fourth-order scheme, where the values of $\partial u/\partial x$ at different grid points are found as a solution of

$$\frac{1}{3}\left.\frac{\partial u}{\partial x}\right|_{i+1,j} + \frac{4}{3}\left.\frac{\partial u}{\partial x}\right|_{i,j} + \frac{1}{3}\left.\frac{\partial u}{\partial x}\right|_{i-1,j} = \frac{u_{i+1,j} - u_{i-1,j}}{\Delta x}, \quad i = 1, \ldots, N. \tag{4.16}$$

4.2.5 Higher-Order Derivatives

Similarly to $\partial u/\partial x$, numerous finite difference approximations exist for the higher-order derivatives. The following examples are given for the x-derivatives. The same formulas can be applied for the derivatives with respect to y, z, and t after trivial substitution of, say, y and j for x and i.

Particularly important are the three-point forward, backward, and central schemes for the second derivative:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} = \frac{u_{i,j} - 2u_{i+1,j} + u_{i+2,j}}{(\Delta x)^2} + O(\Delta x) \tag{4.17}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} = \frac{u_{i,j} - 2u_{i-1,j} + u_{i-2,j}}{(\Delta x)^2} + O(\Delta x) \tag{4.18}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} = \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{(\Delta x)^2} + O((\Delta x)^2) \tag{4.19}$$

Some other schemes are

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} = \frac{-u_{i+3,j} + 4u_{i+2,j} - 5u_{i+1,j} + 2u_{i,j}}{(\Delta x)^2} + O((\Delta x)^2) \tag{4.20}$$

$$\left.\frac{\partial^3 u}{\partial x^3}\right|_{i,j} = \frac{u_{i+2,j} - 2u_{i+1,j} + 2u_{i-1,j} - u_{i-2,j}}{2(\Delta x)^3} + O((\Delta x)^2) \tag{4.21}$$

$$\left.\frac{\partial^4 u}{\partial x^4}\right|_{i,j} = \frac{u_{i+2,j} - 4u_{i+1,j} + 6u_{i,j} - 4u_{i-1,j} + u_{i-2,j}}{(\Delta x)^4} + O((\Delta x)^2) \tag{4.22}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} = \frac{-u_{i+2,j} + 16u_{i+1,j} - 30u_{i,j} + 16u_{i-1,j} - u_{i-2,j}}{12(\Delta x)^2} + O((\Delta x)^4). \tag{4.23}$$

The consistency and the order of approximation of the schemes can be verified in the same way as in the previous sections: by developing the Taylor series expansions around (x_i, y_j) and substituting them into the right-hand sides of (4.17)–(4.23).

More complex finite difference formulas can often be considered as results of repeated application of the elementary forward, backward, and central schemes (4.4), (4.9), and (4.10). As an example, let us derive the central difference approximation of the second derivative (4.19) from the central scheme (4.10). Using the shorthand notation $f = \partial u/\partial x$, we write the second derivative as $\partial^2 u/\partial x^2 = \partial/\partial x\,(\partial u/\partial x) = \partial f/\partial x$, and apply (4.10) to approximate $\partial f/\partial x$ at (x_i, y_j). The formula is modified a little. We use the step $\Delta x/2$ instead of Δx and the values of f at the half-integer points $x_{i-1/2} = x_i - \Delta x/2$ and $x_{i+1/2} = x_i + \Delta x/2$. These are fictitious grid points. As will be seen very soon, we do not need values of u at them. The result is

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} \approx \frac{f_{i+1/2,j} - f_{i-1/2,j}}{\Delta x} = \frac{(\partial u/\partial x)_{i+1/2,j} - (\partial u/\partial x)_{i-1/2,j}}{\Delta x}. \tag{4.24}$$

Derivatives $\partial u/\partial x$ can be again discretized using the central difference formula (4.10) with half-step as

$$\left.\frac{\partial u}{\partial x}\right|_{i+1/2,j} \approx \frac{u_{i+1,j} - u_{i,j}}{\Delta x}, \quad \left.\frac{\partial u}{\partial x}\right|_{i-1/2,j} \approx \frac{u_{i,j} - u_{i-1,j}}{\Delta x}. \tag{4.25}$$

Substitution into (4.24) gives the central difference formula (4.19), which, as we see now, is a result of repeated use of (4.10).

4.2.6 Mixed Derivatives

The mixed derivatives can be approximated by applying the formulas already given, first in one direction and then in the other. For example, let us approximate the mixed derivative $\partial^2 u/\partial x \partial y$ on the two-dimensional uniform structured grid shown in Figure 4.1a. We rewrite it as $\partial\,(\partial u/\partial y)\,/\partial x$ and approximate the x-derivative by the forward difference (4.4):

$$\left.\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial y}\right)\right|_{i,j} = \frac{(\partial u/\partial y)\,|_{i+1,j} - (\partial u/\partial y)\,|_{i,j}}{\Delta x} + O(\Delta x).$$

Each y-derivative also has to be approximated. Let us apply the backward difference (4.9):

$$\left.\frac{\partial u}{\partial y}\right|_{i+1,j} = \frac{u_{i+1,j} - u_{i+1,j-1}}{\Delta y} + O(\Delta y),$$

$$\left.\frac{\partial u}{\partial y}\right|_{i,j} = \frac{u_{i,j} - u_{i,j-1}}{\Delta y} + O(\Delta y).$$

Substituting into the previous expression we obtain

$$\left.\frac{\partial^2 u}{\partial x \partial y}\right|_{i,j} = \frac{1}{\Delta x}\left(\frac{u_{i+1,j} - u_{i+1,j-1}}{\Delta y} - \frac{u_{i,j} - u_{i,j-1}}{\Delta y}\right) + O(\Delta x, \Delta y), \tag{4.26}$$

where $O(\Delta x, \Delta y)$ stands for a combination of the truncation errors of the orders of Δx and Δy. Other combinations of finite-difference formulas lead to different approximations. For example, backward differentiation in x and forward in y results in

$$\left.\frac{\partial^2 u}{\partial x \partial y}\right|_{i,j} = \frac{1}{\Delta x}\left(\frac{u_{i,j+1} - u_{i,j}}{\Delta y} - \frac{u_{i-1,j+1} - u_{i-1,j}}{\Delta y}\right) + O(\Delta x, \Delta y), \tag{4.27}$$

while taking forward formula in x and central in y we obtain

$$\left.\frac{\partial^2 u}{\partial x \partial y}\right|_{i,j} = \frac{1}{\Delta x}\left(\frac{u_{i+1,j+1} - u_{i+1,j-1}}{2\Delta y} - \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y}\right) + O(\Delta x, (\Delta y)^2). \tag{4.28}$$

At last, the approximation of the second order in both directions is generated if we use central differences in both x and y:

$$\left.\frac{\partial^2 u}{\partial x \partial y}\right|_{i,j} = \frac{1}{2\Delta x}\left(\frac{u_{i+1,j+1} - u_{i+1,j-1}}{2\Delta y} - \frac{u_{i-1,j+1} - u_{i-1,j-1}}{2\Delta y}\right) + O((\Delta x)^2, (\Delta y)^2). \tag{4.29}$$

Verification of these formulas requires two-dimensional Taylor series or sequential application of Taylor expansion in the x- and y-directions. The

accuracy of the approximation is determined by the terms of the lowest order with respect to both Δx and Δy. The truncation error is, therefore, expressed as a function of Δx and Δy, and the order of approximation is defined as, for example, "first in x, second in y" for (4.28) or "second in x and y" for (4.29).

4.2.7 Truncation Error of Linear Interpolation

In addition to the approximation of partial derivatives, development of many finite difference and finite volume methods requires approximate evaluation of variables at points that do not belong to the computational grid. This is usually done using linear interpolation from neighboring grid points. The operation introduces truncation error of the second order.

Let us prove the order of approximation for the one-dimensional example, in which the function $f(x)$ has to be estimated at the half-integer point $x_{i+1/2} = x_i + \Delta x/2$. The proof can be easily generalized to the case of arbitrary and even multidimensional interpolation.

The linear interpolation formula in our example is

$$f_{i+1/2} \approx \frac{f_i + f_{i+1}}{2}. \tag{4.30}$$

We use the Taylor series expansions around the point $(x_{i+1/2})$:

$$f_{i+1} = f_{i+1/2} + \left.\frac{\partial f}{\partial x}\right|_{i+1/2} \left(\frac{\Delta x}{2}\right) + \frac{1}{2!} \left.\frac{\partial^2 f}{\partial x^2}\right|_{i+1/2} \left(\frac{\Delta x}{2}\right)^2 + O\left((\Delta x)^3\right),$$

$$f_i = f_{i+1/2} - \left.\frac{\partial f}{\partial x}\right|_{i+1/2} \left(\frac{\Delta x}{2}\right) + \frac{1}{2!} \left.\frac{\partial^2 f}{\partial x^2}\right|_{i+1/2} \left(\frac{\Delta x}{2}\right)^2 + O\left((\Delta x)^3\right).$$

Adding the two formulas together, dividing by 2, and rearranging, we obtain

$$f_{i+1/2} = \frac{f_i + f_{i+1}}{2} - \frac{1}{2} \left.\frac{\partial^2 f}{\partial x^2}\right|_{i+1/2} \left(\frac{\Delta x}{2}\right)^2 + O\left((\Delta x)^3\right) = \frac{f_i + f_{i+1}}{2} + O\left((\Delta x)^2\right). \tag{4.31}$$

4.3 APPROXIMATION OF PARTIAL DIFFERENTIAL EQUATIONS

4.3.1 Approach and Examples

A finite difference representation of a PDE is obtained by replacing each term of the equation by its finite difference approximation. It is imperative that *the approximations of all terms are derived at the same grid point and the same time layer*. Let us consider, as an example, the modified heat equation

$$\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2} + f(x,t), \tag{4.32}$$

and derive, in a more educated manner, the scheme introduced at the end of Chapter 3.

The scheme is developed on a uniform grid with steps Δx and Δt (see Figure 3.8). We write the approximation at the point x_i and time layer t^n. A forward difference of the first order is used for the time derivative:

$$\left.\frac{\partial u}{\partial t}\right|_i^n = \frac{u_i^{n+1} - u_i^n}{\Delta t} + O(\Delta t).$$

The central difference of the second order is used for the x-derivative:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_i^n = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} + O((\Delta x)^2).$$

The nonderivative term in the right-hand side is represented by its value at the discretization point as $f_i^n = f(x_i, t^n)$. Substitution into the PDE yields

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + O(\Delta t) = a^2 \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} + O((\Delta x)^2) + f(x_i, t^n),$$

or

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = a^2 \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} + f(x_i, t^n) + \text{T.E.}, \tag{4.33}$$

where the truncation error is a combination of the truncated terms in formulas for $\partial u/\partial t$ and $\partial^2 u/\partial x^2$:

$$\text{T.E.} = O(\Delta t) + O((\Delta x)^2) = O(\Delta t, (\Delta x)^2). \tag{4.34}$$

This is *the truncation error of discretization of a partial differential equation*. Its order, which is also called *the order of the finite difference scheme* is defined by the lowest order truncated terms with respect to Δt and Δx. As seen in (4.34), the forward in time, central in space scheme for the heat equation is of the first order in time and second order in space.

The finite difference schemes are often illustrated by the *difference molecules*. This is particularly convenient for describing the schemes applied to model equations, such as the heat equation, with only two independent variables, for example, x and t. A difference molecule shows all the grid points that are involved into the finite difference approximation of the equation at the point (x_i, t^n). The difference molecule for the scheme (4.33) is shown in Figure 4.4a.

The next example shows how to discretize a PDE with variable coefficients. We consider one-dimensional heat transfer with variable thermal conductivity $\kappa(x)$. The PDE is derived from (2.27) by setting the velocity $\boldsymbol{V}$ to zero and assuming that the temperature field T depends only on x and t:

$$\rho C \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(\kappa \frac{\partial T}{\partial x}\right). \tag{4.35}$$

The new issue is how to correctly approximate the spatial derivative term in the right-hand side. The finite difference formula approximating the external derivative should be written for the *entire* expression under the derivative, including the variable coefficient $\kappa(x)$. Let us, for example, derive a scheme of the second order in x and first order in t based on the difference molecule shown in Figure 4.4a. We apply the procedure similar to the derivation of the central difference formula (4.19) for the second

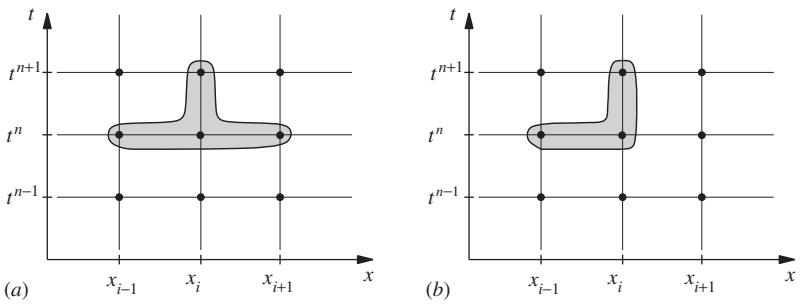


Figure 4.4 *(a)* Difference molecule for the scheme (4.33) for heat equation. *(b)* Difference molecule for the upwind scheme (4.41) for linear convection equation.

derivative $\partial^2 u/\partial x^2$ by repeated application of the approximation (4.10) of the first derivative (see (4.24) and (4.25)). The external derivative is approximated at (x_i, t^n) using the values at the half-integer grid points $x_{i-1/2} = x_i - \Delta x/2$ and $x_{i+1/2} = x_i + \Delta x/2$:

$$\frac{\partial}{\partial x}\left(\kappa \frac{\partial T}{\partial x}\right)\Bigg|_i^n = \frac{\kappa_{i+1/2}(\partial u/\partial x)_{i+1/2} - \kappa_{i-1/2}(\partial u/\partial x)_{i-1/2}}{\Delta x}, \tag{4.36}$$

where $\kappa_{i\pm1/2}$ stands for the values of $\kappa(x)$ at $x_{i\pm1/2}$. We now approximate $\partial u/\partial x$ at $(x_{i\pm1/2}, t^n)$ by the central differences to obtain

$$\frac{\partial}{\partial x}\left(\kappa \frac{\partial T}{\partial x}\right)\Bigg|_i^n = \frac{1}{\Delta x}\left[\kappa_{i+1/2}\frac{u_{i+1}^n - u_i^n}{\Delta x} - \kappa_{i-1/2}\frac{u_i^n - u_{i-1}^n}{\Delta x}\right]. \tag{4.37}$$

If κ is a known function of x, its values at the half-integer points can be calculated directly. If not, for example when κ depends on the solution u (conductivity coefficient varies with temperature), only the integer-point values of κ are available. In this case, we have to apply the linear interpolations

$$\kappa_{i-1/2} \approx \frac{\kappa_i + \kappa_{i-1}}{2}, \quad \kappa_{i+1/2} \approx \frac{\kappa_i + \kappa_{i+1}}{2}. \tag{4.38}$$

This operation is of the second order of accuracy and, thus, the accuracy of the entire scheme is not compromised.

For the time derivative, we use the same forward scheme of the first order as before. The final finite difference approximation of the PDE is

$$(\rho C)_i^n \frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{1}{\Delta x}\left[\frac{(\kappa_{i+1} + \kappa_i)(u_{i+1}^n - u_i^n)}{2\Delta x} - \frac{(\kappa_{i-1} + \kappa_i)(u_i^n - u_{i-1}^n)}{2\Delta x}\right]. \tag{4.39}$$

The truncation error is of the first order in t and second order in x.

The last example is the approximation of the linear convection equation

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0, \tag{4.40}$$

where $c > 0$ is a constant. We approximate the equation at the grid point (x_i, t^n) and apply the forward difference for the time derivative and backward difference for the x-derivative. The result is the *upwind scheme*:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_i^n - u_{i-1}^n}{\Delta x} = 0. \tag{4.41}$$

Both finite difference approximations of the derivatives are of the first order. The truncation error of discretization of the equation is, therefore, $O(\Delta t, dx)$. The difference molecule for the upwind scheme is shown in Figure 4.4b.

4.3.2 Interpretation of Truncation Error: Numerical Dissipation and Dispersion

The consistency and the order of the truncation error are the main characteristics of a finite difference approximation. Further information about the truncation error and its possible effect on the solution can be obtained from the *modified equation* as is briefly discussed here using the example of the upwind scheme (4.41). A more detailed discussion can be found in the books listed at end of this chapter.

To interpret the truncation error, we substitute the Taylor series expansions

$$u_i^{n+1} = u_i^n + \left(\frac{\partial u}{\partial t}\right)\Big|_{i,j}^n \Delta t + \left(\frac{\partial^2 u}{\partial t^2}\right)\Big|_{i,j}^n \frac{(\Delta t)^2}{2} + \cdots$$

$$u_{i-1}^n = u_i^n - \left(\frac{\partial u}{\partial x}\right)\Big|_{i,j}^n \Delta x + \left(\frac{\partial^2 u}{\partial x^2}\right)\Big|_{i,j}^n \frac{(\Delta x)^2}{2} + \cdots$$

into (4.41). After rearranging, we obtain

$$u_t + cu_x = -\frac{\Delta t}{2}u_{tt} + \frac{c\Delta x}{2}u_{xx} - \frac{(\Delta t)^2}{6}u_{ttt} - c\frac{(\Delta x)^2}{6}u_{xxx} + \cdots, \tag{4.42}$$

where we use the abbreviations u_t, u_x, and so on for the partial derivatives of u taken at the grid point (x_i, t^n). The equation (4.42) is quite remarkable. The left-hand side is the original PDE and, thus, would be equal to zero, if u were the exact solution of the original PDE. Since u is an extension of the finite difference solution, it satisfies not the original PDE but the *modified equation* (4.42). The nonzero right-hand side of (4.42) represents the truncation error.

The composition and behavior of the truncation error can be better understood if we replace the time derivatives u_{tt} and u_{ttt} by spatial derivatives. It can be done by differentiating (4.42) with respect to x and t and performing algebraic operations with the resulting equations. The final equation is

$$u_t + cu_x = \frac{c\Delta x}{2}(1-\nu)u_{xx} - c\frac{(\Delta x)^2}{6}(2\nu^2 - 3\nu + 1)u_{xxx} + O((\Delta x)^3, (\Delta x)^2\Delta t, \Delta x(\Delta t)^2, (\Delta t)^3), \tag{4.43}$$

where $\nu = c\Delta t/dx$.

Let us analyze the right-hand side of (4.43). The lowest-order term $(c\Delta x/2)(1-\nu)u_{xx}$ looks familiar. The second spatial derivative multiplied by a constant can be found in the right-hand sides of heat, diffusion, or Navier-Stokes equations. On all occasions, it represents the physical processes of diffusion or dissipation—that is, smoothing of gradients of the solution. In our case, the diffusion has numerical rather than physical character. The commonly used name is *numerical dissipation*. The constant coefficient at u_{xx} is called *numerical viscosity* or *artificial viscosity*.

The effect of numerical dissipation is illustrated in Figure 4.5. Let the initial condition for the linear convection equation (4.40) has the form of a sharp front, as in Figure 4.5a. The exact solution of the PDE retains the shape and travels to the right with the constant speed c. Due to the presence of numerical dissipation, the computed solution does not behave in this manner. The front is gradually smoothed out and takes the form illustrated in Figure 4.5b.

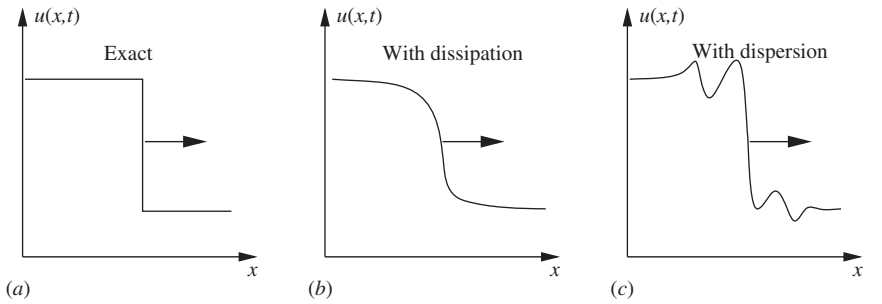


Figure 4.5 Effect of numerical dissipation and dispersion on wavelike solutions: *(a)* exact solution, *(b)* numerical solution with strong dissipation, *(c)* numerical solution with strong dispersion.

Another type of truncation error is associated with the second term $c((\Delta x)^2/6)(2\nu^2 - 3\nu + 1)u_{xxx}$ in the right-hand side of (4.43). It can also be identified as an artificial pseudo-physical effect. A third-order spatial derivative is known to lead to dispersion, which is the phenomenon of dependence of the wave speed on the wavelength. The exact solution of a PDE, such as, for example, the solution shown in Figure 4.5a, can be represented as a superposition of Fourier harmonics with different wavelengths λ. The presence of the dispersion term in (4.43) makes them propagate with different speeds rather than with the common speed c. With time, the harmonics separate in phase and form a solution illustrated in Figure 4.5c. The dispersion, when it is caused by the truncation error, is called *numerical dispersion*.

The numerical dissipation and dispersion are common in finite difference solutions. Generally, any term with an even-order space derivative in the right-hand side of the modified equation creates numerical dissipation, while any term with an odd-order space derivative creates numerical dispersion. Of course, only the terms of the lowest order in Δx and Δt are important. For example, the truncation error of the first-order scheme (4.41) is clearly dominated by the numerical dissipation term $c\Delta x(1 - \nu)u_{xx}/2$. Such schemes are called *predominantly dissipative*, or simply *dissipative*. We should expect the behavior illustrated in Figure 4.5b, rather than Figure 4.5c. Other schemes, some of which are discussed in the following chapters, have the lowest-order term of T.E. proportional to the odd derivative. They are *predominantly dispersive* or *dispersive*.

We have discussed the modified equation and numerical dissipation and dispersion on the example of a hyperbolic equation. The analysis can also be applied to parabolic systems. This can sometimes produce quite interesting results (one example is given in Chapter 7 in the discussion of the simple explicit method for the heat equation). In general, however, the composition of the truncation error is less important for parabolic system than for hyperbolic ones.

The reason is the different nature of the physical phenomena represented by parabolic and hyperbolic equations. The hyperbolic equations often describe wave propagation in media with low natural dissipation and diffusion. Good examples are the supersonic aerodynamic or acoustic flows. Viscosity and heat diffusivity do not play important roles in these phenomena, except within the boundary layers and shock waves in high-speed flows. Therefore, the waves propagate in an almost ideal, nonviscous and nondiffusive manner. For such waves, the numerical dispersion and dissipation, if left unchecked, can result in strongly distorted solutions. Obviously, evaluation and control of both dispersive and dissipative errors are not just desirable but necessary.

In the systems described by parabolic equations, strong natural dissipation is typically present as a part of the described physical process. In general, schemes of second or higher orders in space are used for parabolic equations. For such schemes, assuming that the grid steps are sufficiently small to accurately resolve the space and time gradients of the solution, the physical diffusion is likely to dominate over the numerical diffusion. The dispersive errors are either small or avoided completely by the schemes used for parabolic equations.

4.3.3 Boundary and Initial Conditions

The finite difference approximation of the PDE is applied only at internal grid points of the computational domain. At the grid points lying at the boundaries, the scheme should approximate the boundary and initial conditions. Usual finite difference formulas can be employed with the obvious limitation that they can only use the values of the unknown function at the points within the domain, that is on only one side of the boundary.

There is one important requirement. The order of approximation of the boundary conditions should not be lower than the order of the scheme used to approximate the PDE. The reason is that the order of the approximation of the entire solution is equal to the lowest order of approximation of any of its parts. If a lower-order scheme is used for the boundary conditions, the larger truncation error propagates into the solution domain and compromises otherwise higher accuracy of computations.

Let us consider the example of the modified heat equation (4.32) with the Dirichlet and Neumann boundary conditions

$$u(0,t) = a(t), \quad \frac{\partial u}{\partial x}(L,t) = b(t), \text{ at } t_0 < t < t_{\text{end}},$$

where a and b are known functions of time, and the initial condition

$$u(x,t_0) = g(x) \quad \text{at} \quad 0 \le x \le L.$$

Let the grid points x_0 and x_N correspond to the boundaries $x = 0$ and $x = L$, and let the time layer t^0 correspond to the initial moment of time $t = t_0$.

Approximation of the Dirichlet boundary condition is easy. We assume that

$$u_0^n = a(t^n) = a^n, \quad n = 1, \ldots, M, \tag{4.44}$$

which is an exact (no error is introduced) representation. For the Neumann condition, care must be taken to approximate the partial derivative by a formula of second order so as not to compromise the second order accuracy of the entire scheme. If, for example, we use the backward difference

$$\frac{\partial u}{\partial x}(L,t) = \frac{u_N^n - u_{N-1}^n}{\Delta x} + O(\Delta x),$$

the error $O(\Delta x)$ will quickly propagate into the interior of the domain and the entire solution will have the truncation error T.E. $= O(\Delta t, \Delta x)$. The proper course of action is to apply one of the one-sided finite difference formulas of the second order, for example, (4.14), and write

$$\frac{3u_N^n - 4u_{N-1}^n + u_{N-2}^n}{2\Delta x} = b(t^n) = b^n, \quad n = 1, \ldots, M. \tag{4.45}$$

This can be used to compute u_N^n after the values of u_i^n at the interior grid points have already been found:

$$u_N^n = \frac{1}{3}\left(2\Delta x b^n + 4u_{N-1}^n - u_{N-2}^n\right).$$

At last, the initial condition is represented exactly by

$$u_i^0 = g(x_i), \quad i = 1, \ldots, N. \tag{4.46}$$

There exists a technique of imposing boundary conditions that does not require the grid points lying at the boundary. The *ghost* points located symmetrically on the outer side of the boundary are used instead. Two examples are presented in Figure 4.6. In one, shown in Figure 4.6a, the Dirichlet boundary condition $u|_{\partial\Omega} = a$ is approximated with the second order of accuracy by assigning the value of u at the ghost point x_{N+1} to $u_{N+1}^n = a - (u_N^n - a)$. The second example shown in Figure 4.6b illustrates how one can approximate the Neumann condition $\partial u/\partial x|_{\partial\Omega} = b$ by choosing the $u_{N+1}^n = u_N^n + 2b\Delta x$ so that the slope of the line connecting u_N^n and u_{N+1}^n is equal to b.

4.3.4 Consistency of Numerical Approximation

We have seen that the truncation error introduced by the finite difference approximation of a PDE problem is a combination of the errors of approximation of all derivative and nonderivative terms and boundary conditions at all grid points and time layers.

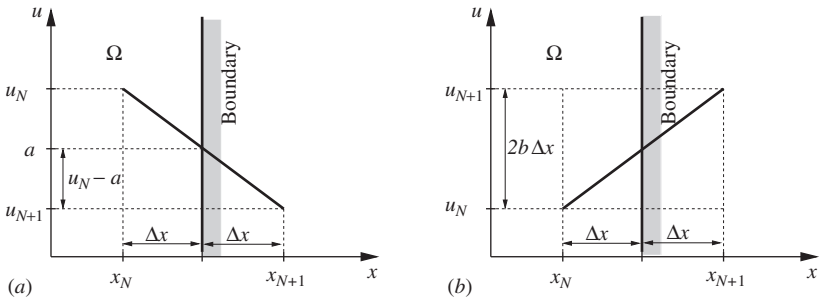


Figure 4.6 Setting boundary conditions using ghost points: *(a)* Dirichlet boundary condition $u|_{\partial\Omega} = a$; *(b)* Neumann boundary condition $\partial u/\partial x|_{\partial\Omega} = b$.

The necessary (but not sufficient) condition for the accurate finite difference solution is the generalization of the consistency condition introduced earlier for partial derivatives. The truncation errors of the finite difference approximation of the equation and boundary and initial conditions should vanish as all the grid steps approach zero:

$$\lim_{\Delta x,\Delta y,\Delta z,\Delta t\to 0} \text{T.E.} = 0. \tag{4.47}$$

If this is the case, the finite difference scheme is called *consistent*. Obviously, the consistency is achieved if the truncation error is at least of the first order in every coordinate and time at all points and time layers of the computational grid. *The scheme has to be at least of the first order to be consistent*.

4.3.5 System of Difference Equations

The approximations of the PDE written for all internal grid points and approximations of boundary and initial conditions form a system of algebraic equations called *difference equations* or, in more general terms, *discretization equations*. The system has to be solved to find the values of the unknown functions at the grid points.

As a rule, all the equations of the discretization system are coupled with each other. The system is irreducible to smaller independent systems and should be solved as a whole. This can be a difficult and computationally intensive task. It is not uncommon for a CFD analysis to use grids consisting of millions of nodes. We should also take into account that several variables usually have to be calculated at every node, each variable requiring a separate equation. For example, computing a flow

of an incompressible fluid with heat transfer would require at least five variables: three velocity components, pressure, and temperature.

The need to solve huge systems of algebraic equations has always been a critical aspect, even a bottleneck of CFD analysis. The available computing power limits the size of the system that can be solved in reasonable time and, thus, the type of the flow and heat transfer processes that can be accurately analyzed.

Significant efforts have been invested over the years to develop effective methods of solving large systems of discretization equations. The most important of them are discussed in this book. The first, rather simple idea appears in the next section.

4.3.6 Implicit and Explicit Methods

Let us consider a finite difference scheme applied to a *time-dependent problem*. The natural and commonly used approach to solution of discretization equations for such problems is illustrated in Figure 4.7. We start with the initial conditions at the time layer t^0, use the discretization equations approximating the PDE and boundary conditions to find the unknowns at the time layer t^1, then at t^2, and so on, moving or marching ahead one layer at a time. Each such time step requires solution of a system that consists of equations for all space grid points. Albeit still large, such systems are much smaller than the cumulative system that includes all the time layers.

With respect to the marching procedure, there are two types of the discretization schemes: *explicit* and *implicit*. The types are presented here

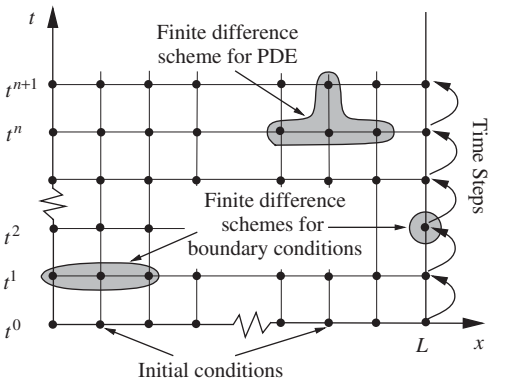


Figure 4.7 Illustration of the marching procedure.

using the example of a finite difference solution of the modified heat equation (4.32). It will be clear from the discussion that the concepts equally apply to other equations and other discretization methods, for example, finite element or spectral.

Explicit Schemes: Let us first consider the already introduced scheme

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = a^2 \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} + f_i^n. \tag{4.48}$$

The difference molecule is shown in Figure 4.4a. The difference equation includes only one value of u from the $(n+1)$st time layer. The transition from layer t^n to layer t^{n+1} is easy. One has to rewrite (4.48) as

$$u_i^{n+1} = u_i^n + \Delta t \left(a^2 \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} + f_i^n \right), \tag{4.49}$$

that is, to rewrite the difference equation so as to *explicitly* express the unknown quantity u_i^{n+1} at the layer t^{n+1} through the quantities u_i^n at the layer t^n known from the previous time step. The schemes for which this is possible are called *explicit* schemes.

Implicit Schemes: Another scheme for the heat equation can be developed if we, for example, approximate the PDE at the point (x_i, t^{n+1}) using backward difference of the first order for the time derivative and central difference of the second order for the space derivative. The result is the scheme

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = a^2 \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2} + f_i^{n+1} \tag{4.50}$$

with truncation error T.E. $= O(\Delta t, (\Delta x)^2)$. The difference molecule is shown in Figure 4.8. If we now move all the terms needed to be calculated at the new time layer into the left-hand side, we obtain

$$u_i^{n+1} - a^2 \frac{\Delta t}{(\Delta x)^2} \left(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1} \right) = u_i^n + \Delta t \, f_i^{n+1}. \tag{4.51}$$

We see that a time step of the marching procedure cannot be completed as easily as before. No explicit formula exists that expresses the unknown u_i^{n+1} through the known values such as u_i^n, u_{i-1}^n, u_{i+1}^n. Coupled equations (4.51) for $i = 1, \ldots, N-1$ together with the equations approximating the

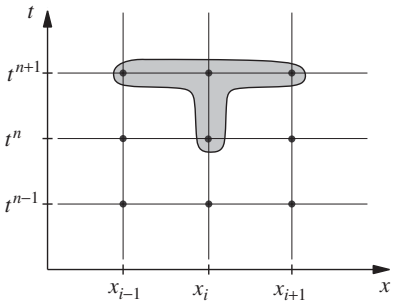


Figure 4.8 Difference molecule of the implicit scheme for the heat equation (4.50).

boundary conditions at $i = 0$ and $i = N$ must be solved as a system of linear algebraic equations. Scheme (4.51) is an example of an *implicit* scheme.

One time step of an implicit scheme requires larger amount of computational work than one step of an explicit scheme, since a system of algebraic equations has to be solved. The implicit methods are, nevertheless, in wide use for the reason of their better stability properties. The issues are further discussed in section 6.3. For now, we only mention that larger time steps can be used with an implicit scheme, so the solution can be computed in a smaller number of steps.

We also have to mention that efficient numerical methods have been developed for solution of large algebraic systems. One of them, the Thomas algorithm suitable for 1D problems, is presented in Chapter 7, while the general discussion is postponed until Chapter 8.

4.4 DEVELOPMENT OF FINITE DIFFERENCE SCHEMES

How do we develop finite difference schemes? The simplest and, probably, most efficient way is to look up the desired scheme in a CFD book. If this solution is satisfactory, the reader can skip this section without much harm for future understanding. If this is not the case, one of the two methods considered here can be used. There is the third method based on the principles of finite volume approximation. Recognizing its importance for modern CFD, we describe it separately and in more detail in Chapter 5.

The general task is formulated as follows: *To develop a finite difference approximation of a given order of accuracy for a partial derivative using a given set of grid points*.

4.4.1 Taylor Series Expansions

This method is a generalization of the procedure applied in section 4.2 to develop our first finite difference formulas.

As a demonstration example, we will use the one-dimensional heat equation

$$\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2} + f(x,t) \tag{4.52}$$

and develop a scheme of the second order in space and first order in time. The scheme will be explicit and based on the finite difference molecule shown in Figure 4.4a.

For the space derivative, we have to develop a formula that uses u_{i-1}^n, u_i^n, and u_{i+1}^n and approximates $\partial^2 u/\partial x^2$ at the point (x_i, t^n). The truncation error should be $O((\Delta x)^2)$. We start with the Taylor series expansions for u_{i-1}^n and u_{i+1}^n around (x_i, t^n):

$$u_{i-1}^n = u_i^n - \left.\frac{\partial u}{\partial x}\right|_i^n \Delta x + \frac{1}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|_i^n (\Delta x)^2 - \frac{1}{6}\left.\frac{\partial^3 u}{\partial x^3}\right|_i^n (\Delta x)^3 + O\left((\Delta x)^4\right)$$

$$u_{i+1}^n = u_i^n + \left.\frac{\partial u}{\partial x}\right|_i^n \Delta x + \frac{1}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|_i^n (\Delta x)^2 + \frac{1}{6}\left.\frac{\partial^3 u}{\partial x^3}\right|_i^n (\Delta x)^3 + O\left((\Delta x)^4\right).$$

The series are truncated at the fourth-order term. In general, the truncation limit is determined experimentally, but it should be not lower than the sum of the order of approximated derivative and the desired order of the truncation error.

The equations are solved for $\partial^2 u/\partial x^2$ in terms of the values of u at the grid points. Dividing by $(\Delta x)^2$ we get

$$u_{i-1}^n (\Delta x)^{-2} = u_i^n (\Delta x)^{-2} - \left.\frac{\partial u}{\partial x}\right|_i^n (\Delta x)^{-1} + \frac{1}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|_i^n - \frac{1}{6}\left.\frac{\partial^3 u}{\partial x^3}\right|_i^n \Delta x + O\left((\Delta x)^2\right)$$

$$u_{i+1}^n (\Delta x)^{-2} = u_i^n (\Delta x)^{-2} + \left.\frac{\partial u}{\partial x}\right|_i^n (\Delta x)^{-1} + \frac{1}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|_i^n + \frac{1}{6}\left.\frac{\partial^3 u}{\partial x^3}\right|_i^n \Delta x + O\left((\Delta x)^2\right).$$

We now sum the first and second equations multiplied by arbitrary constants a and b. After rearranging, we find

$$\frac{a+b}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|_i^n = (\Delta x)^{-2}\left[a u_{i-1}^n + b u_{i+1}^n - (a+b)u_i^n\right] + (\Delta x)^{-1} \times \left[\left.\frac{\partial u}{\partial x}\right|_i^n (a-b)\right] + \Delta x\left[\frac{1}{6}\left.\frac{\partial^3 u}{\partial x^3}\right|_i^n (a-b)\right] + O\left((\Delta x)^2\right). \tag{4.53}$$

It is obvious now that the needed finite difference formula can be obtained if we find the values of a and b, which provide zero coefficients at $\partial u/\partial x$ and $\partial^3 u/\partial x^3$ and the coefficient equal to 1 at $\partial^2 u/\partial x^2$. This can be expressed as a system

$$\begin{cases} a+b=2 \\ a-b=0 \end{cases} \tag{4.54}$$

In the general case, we would have different cancelation conditions for terms with $\partial u/\partial x$ and $\partial^3 u/\partial x^3$. The system of three linear algebraic equations for two unknowns could be inconsistent (have no solution). In that case, the initial task would have to be reconsidered by allowing use of more than three grid points. In our particular case, however, the cancelation conditions are identical as a result of the symmetric shape of the difference molecule and the use of constant Δx. The solution is given by $a=1$, $b=1$. The resulting finite difference approximation

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{(\Delta x)^2} + O\left((\Delta x)^2\right), \tag{4.55}$$

is already known to us as the central difference of second order.

For the time derivative $\partial u/\partial t$, we use u_i^n and u_i^{n+1} to develop a formula of the first order in Δt. The procedure is simple. From the Taylor expansion

$$u_i^{n+1} = u_i^n + \left.\frac{\partial u}{\partial t}\right|_i^n \Delta t + O\left((\Delta t)^2\right) \tag{4.56}$$

we find that

$$\frac{\partial u}{\partial t} = \frac{u_i^{n+1} - u_i^n}{\Delta t} + O(\Delta t), \tag{4.57}$$

which completes the derivation. The truncation error $O\left((\Delta x)^2, \Delta t\right)$ was determined in the process.

There is no simple universal rule for the number and distribution of the grid points that have to be included into the difference molecule in order to achieve the desired accuracy. As a rule of thumb, the number is equal to the order of the derivative plus the order of the truncation error. For example, a derivative of the third order in x can be approximated by a formula of third order of accuracy on a molecule that includes six points. Quite often, however, formulas are developed using fewer (as in our example for $\partial^2 u/\partial x^2$) or more points.

As another example, we consider a function $u(x, y)$ and derive a one-sided formula that approximates $\partial u/\partial x$ on three grid points with the truncation error $O\left((\Delta x)^2\right)$. Such formulas are useful for Neumann boundary conditions. The approximation is done at (x_i, y_j) using $u_{i-2,j}$, $u_{i-1,j}$, and $u_{i,j}$. Again, we start with the Taylor series expansions for $u_{i-2,j}$ and $u_{i-1,j}$ around (x_i, y_j):

$$u_{i-2,j} = u_{i,j} - 2\left.\frac{\partial u}{\partial x}\right|_{i,j}\Delta x + 2\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j}(\Delta x)^2 + O\left((\Delta x)^3\right) \tag{4.58}$$

$$u_{i-1,j} = u_{i,j} - \left.\frac{\partial u}{\partial x}\right|_{i,j}\Delta x + \frac{1}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j}(\Delta x)^2 + O\left((\Delta x)^3\right). \tag{4.59}$$

We form the linear combination $a(4.58) + b(4.59)$ with arbitrary constants a and b

$$au_{i-2,j} + bu_{i-1,j} = (a+b)u_{i,j} + (-2a-b)\left.\frac{\partial u}{\partial x}\right|_{i,j}\Delta x + \frac{(4a+b)}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j}(\Delta x)^2 + O\left((\Delta x)^3\right) \tag{4.60}$$

and require that the coefficient at $\partial u/\partial x$ is 1 and the coefficient at $\partial^2 u/\partial x^2$ is zero. This results in a system

$$\begin{cases} -2a - b = 1 \\ 4a + b = 0 \end{cases},$$

which has the solution $a = 1/2$ and $b = -2$. Substitution into (4.60) and rearranging produces the required formula

$$\left.\frac{\partial u}{\partial x}\right|_{i,j} = \frac{u_{i-2,j} - 4u_{i-1,j} + 3u_{i,j}}{2\Delta x} + O\left((\Delta x)^2\right). \tag{4.61}$$

Nonuniform Grids: For nonuniform grids (variable grid steps), the same technique can be applied, although this typically results in more complex formulas. As we discuss in Chapter 12, there is a more efficient way to deal with nonuniform grids based on the use of mapping (coordinate transformation).

4.4.2 Polynomial Fitting

The basic idea of the method is to assume that, locally, the unknown function is approximated by a polynomial of a certain order. The polynomial is "fitted" to the function's behavior by determining its coefficients so that the polynomial is equal to the function exactly at the grid points of a finite difference molecule.

As an example, we consider the Neumann boundary condition

$$\left.\frac{\partial u}{\partial x}\right|_{x=0} = q \quad \text{at} \quad 0 < t < 1. \tag{4.62}$$

To preserve the accuracy of a finite difference scheme, the boundary condition should be approximated with, at least, the same order of accuracy as the partial differential equation itself. Let us assume that the scheme has the truncation error $\sim O\left((\Delta x)^4\right)$, so we have to develop a one-sided finite difference approximation for $(\partial u/\partial x)|_{x=0}$ of the fourth order of accuracy. We use the same uniform grid as before and assume that the boundary is at the grid point $x_0 = 0$. The neighboring points are $x_1 = \Delta x$, $x_2 = 2\Delta x$, and so on.

First, we locally represent $u(x,t)$ as a polynomial of third degree:

$$u(x,t^n) = a + bx + cx^2 + dx^3. \tag{4.63}$$

Obviously, the boundary condition (4.62) applied to (4.63) means

$$b = q. \tag{4.64}$$

What remains is to find a, c, and d such that the polynomial fits the values of u at the grid points adjacent to the boundary. The number of points to be taken must be equal to the number of the still-unknown coefficients. Calculating (4.63) at the three nearest points, we obtain the system

$$\begin{cases} u_1^n = a + q\Delta x + c(\Delta x)^2 + d(\Delta x)^3 \\ u_2^n = a + 2q\Delta x + 4c(\Delta x)^2 + 8d(\Delta x)^3 \\ u_3^n = a + 3q\Delta x + 9c(\Delta x)^2 + 27d(\Delta x)^3 \end{cases},$$

which can be solved for a, c, and d. Doing that and calculating the polynomial at the boundary x_0, we find the approximation of (4.62):

$$u_0^n = \frac{1}{11}\left(18u_1^n - 9u_2^n + 2u_3^n - 6q\,\Delta x\right) + O\left((\Delta x)^4\right). \tag{4.65}$$

Note that the order of the truncation error does not follow directly from the derivation. It has to be determined separately in a Taylor series analysis.

REFERENCES AND SUGGESTED READING

Fletcher, C. A. J. *Computational Techniques for Fluid Dynamics*. Vol. 1, *Fundamental and General Techniques*. Berlin: Springer-Verlag, 1991.

Tannehill, J. C. D. A. Anderson, and R. H. Pletcher. *Computational Fluid Mechanics and Heat Transfer*. Philadelphia: Taylor & Francis, 1997.

PROBLEMS

1. Consider the function $u = \sin x$. Apply the forward difference (4.4), backward difference (4.9), and central difference (4.10) to evaluate the derivative du/dx at $x = 1.0$. Use the uniform grids with steps $\Delta x = 1.0$, 0.5, 0.1, and 0.01. Compare the results with the exact value of the derivative. Determine which grid steps give accurate results (e.g., with the error less than 0.01) for each scheme. Discuss why the central scheme is more accurate than the two others.

2. Write the schemes similar to (4.4), (4.9), and (4.10) for $(\partial u/\partial y)|_{i,j}$.

3. Write the schemes similar to (4.17), (4.18), and (4.19) for $(\partial^2 u/\partial y^2)|_{i,j}$.

4. Write the central finite difference formulas of the second order for $\partial u/\partial x$ and $\partial^2 u/\partial x^2$ at the grid point (x_{i+1}, y_j, t^{n+1}).

5. Verify the consistency and order of the schemes (4.13) and (4.14) using the Taylor series expansions.

6. Verify the consistency and order of the scheme (4.20) using the Taylor series expansions.

7. Consider the generic transport equation $\phi_t + u\phi_x = \mu\phi_{xx}$, where u is a known function of x and t and μ is a constant coefficient. Assume that the computational grid is uniform with steps Δx and Δt. Write the finite difference schemes satisfying the following requirements:

   a) Explicit scheme of the first order in time and second order in space. Use central differences for the space derivatives.

b) Follow the requirements in (a), but the scheme is implicit.

c) Scheme of the first order in time. Use implicit central difference approximation for the diffusion term $\mu\phi_{xx}$ and explicit backward difference approximation for the convection term $u\phi_x$.

8. A finite difference scheme was developed to solve the heat equation. Testing in comparison with a known exact solution showed that the error is of the same order of magnitude as the solution itself and does not tend to zero as the grid steps decrease. Assuming there was no coding errors in the computer program, what was the reason for such behavior?

9. The modified equations of certain finite difference schemes are given below. In each case, determine the order of approximation and find whether the dominant error is due to numerical dissipation or numerical dispersion.

$$u_t + cu_x = \left(\frac{1}{2}c^2\Delta t\right)u_{xx} - \left(\frac{1}{6}c(\Delta x)^2 + \frac{1}{3}c^3(\Delta t)^2\right)u_{xxx} + \cdots$$

$$u_t + cu_x = \frac{1}{6}c(\Delta x)^2(\nu^2 - 1)u_{xxx} - \frac{1}{120}c(\Delta x)^4(9\nu^4 - 10\nu^2 + 1)u_{xxxxx} + \cdots.$$

$$u_t - a^2u_{xx} = \left[\frac{1}{2}a^4\Delta t + \frac{(a\Delta x)^2}{12}\right]u_{xxxx} + \left[\frac{1}{3}a^6(\Delta t)^2 + \frac{1}{12}a^4\Delta t(\Delta x)^2 + \frac{1}{360}a^2(\Delta x)^4\right]u_{xxxxxx} + \cdots$$

10. The Neumann boundary condition $\partial u/\partial x = a$ at $x = 0$ has to be implemented in a finite difference scheme for the heat equation. The grid is uniform with step Δx. Among the finite difference formulas discussed in this chapter, select one to approximate the boundary condition if the scheme's order of approximation in x is

a) First

b) Second

11. Using the method of Taylor series expansion or polynomial fitting, develop the following finite difference schemes:

a) (4.13)
b) (4.17)
c) (4.25).

Programming Exercises Calculate the finite difference solution of the equation (4.32) with $a = 0.5$ and $f(x,t) = \sin 5x$ in the domain $0 < x < \pi$, $0 < t < 50$ with initial condition $u(x,0) = x(\pi - x)$ and boundary conditions $u(0,t) = 0$, $u(\pi,t) = 0$. Use the fully explicit scheme (4.48). Try two grids, one with 100 space points and 100 time layers and another with the number of time layers increased to 100,000. Compare the profile of u at $t = t^{\text{end}} = 20$ with the exact solution of the steady-state problem $u_{\text{exact}} = (25a^2)^{-1} \sin 5x$. Do not be surprised by the difference between the two numerical solutions. The matter is further discussed in Chapter 6.

5

FINITE VOLUME METHOD

5.1 INTRODUCTION AND INTEGRAL FORMULATION

This chapter is entirely devoted to one class of discretization schemes, the finite volume method. Such attention is fully warranted by the special place this method occupies within the field of applied CFD. Many general-purpose codes for fluid flows, including the majority of commercial CFD programs, are based on the finite volume approach. The reasons for popularity will become clear in the discussion that follows. Right now, we only mention the main two: convenience of use with unstructured grids and the global conservation property.

The finite volume schemes are principally different from the classical finite difference schemes in the way they are derived. Instead of discretizing the partial differential equations, we start with the physical conservation laws in the integral form, such as those presented in section 2.7. Discretization is applied directly to the integral equations written for small control (finite) volumes. In this regard, the method resembles the finite element technique. The distinctive character of the finite volume method is sometimes taken as far as claiming it to be a completely separate approach unrelated to the finite difference approximation. We adhere to a more moderate view, according to which the finite volume method is, albeit special, still a type of the general finite difference technique.

In the following discussion, the method will be illustrated using the formal conservation equation introduced in section 2.7:

$$\frac{d}{dt}\int_{\Omega}\Phi d\Omega = -\int_{S}\Phi \boldsymbol{V}\cdot\boldsymbol{n}dS + \int_{S}\chi\nabla\Phi\cdot\boldsymbol{n}dS + \int_{\Omega}Qd\Omega. \qquad (5.1)$$

It is written for an arbitrary control volume Ω with surface S and outward-facing unit normal to the surface $\boldsymbol{n}$ (see Figure 2.4), and expresses the 'conservation' of the scalar field Φ within Ω. Each term in (5.1) has counterparts in the integral equations of conservation of mass, momentum, and energy. The left-hand side represents the rate of change of the total amount of Φ within Ω. Each integral in the right-hand side represents a certain typical way by which this amount can be changed. The surface integral $-\int_S \Phi \boldsymbol{V} \cdot \boldsymbol{n} dS$ shows the *convective flux* through the boundary of the control volume S. It appears due to the transport of Φ by velocity $\boldsymbol{V}$. The integral $\int_S \chi \nabla \Phi \cdot \boldsymbol{n} dS$ has the form typical for cross-boundary transport by diffusion mechanisms, such as viscous friction or heat conduction. χ is a constant or variable coefficient of diffusion. This term gives the *diffusive flux* through the boundary S. At last, the volume integral $\int_\Omega Q d\Omega$ represents the rate of change of the amount of Φ caused by internal sources of intensity Q per unit volume.

Let us identify the control volume Ω. In the finite volume method, the computational domain is divided into small, nonoverlapping subdomains called *cells*. A finite volume scheme is derived by applying integral balance equation, such as (5.1), to every cell. The next step discussed in sections 5.2 and 5.3 is to approximate the integrals, so that each integral equation is replaced by an algebraic discretization equation. Taken for all cells, the discretization equations form a system that has to be solved in the same way as in the case of a standard finite difference scheme.

Special consideration has to be given to the cells adjacent to the boundaries of the computational domain. Surface integrals need to be changed on the part of $\partial\Omega$ coinciding with the boundary so as to incorporate the boundary conditions. The procedure is discussed in section 5.4.

Note that the following discussion concerns only *spatial discretization*. The time discretization in the finite volume schemes is not different from the time discretization applied to the finite difference schemes or other discretization schemes in general.

5.1.1 Finite Volume Grid

Examples of finite volume grids are shown in Figure 5.1. Although they do not include a three-dimensional grid, it should be understood that the principles of finite volume approximation are valid in the general three-dimensional case.

The simplest finite volume grid is for one-dimensional problems. The control volumes are intervals. For example, Figure 5.1a presents a finite volume grid with cells $\Omega_i = [x_{i-1/2}, x_{i+1/2}]$. The outward-facing normal

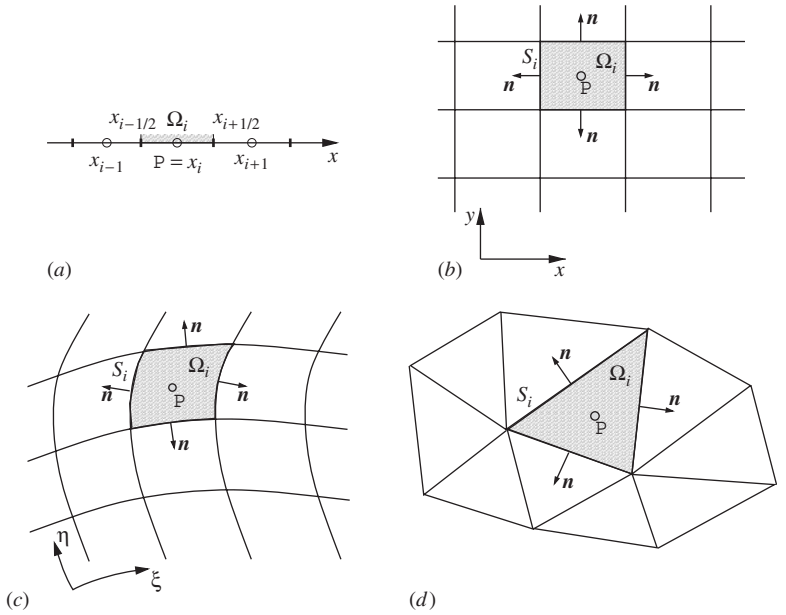


Figure 5.1 Examples of finite volume grids. *(a)* One-dimensional grid; *(b)* Two-dimensional Cartesian structured grid; *(c)* Two-dimensional curvilinear structured grid; *(d)* Two-dimensional unstructured grid.

vector $\boldsymbol{n}$ is in the negative x-direction on the left-hand boundary $x_{i-1/2}$ and in the positive x-direction on the right-hand boundary $x_{i+1/2}$.

In the two-dimensional and three-dimensional cases, we have a choice between structured and unstructured grids. In the structured grids, the cells are quadrilateral (in the two-dimensional case) or hexahedral (in the three-dimensional case) and arranged in a structured pattern along the lines of a Cartesian (as in Figure 5.1b) or curvilinear (as in Figure 5.1c) coordinate system. In unstructured grids, the cells may have various shapes, such as prisms, tetrahedra, hexahedra, or pyramids, in three dimensions and plane figures, such as triangles, quadrilaterals, or other convex polygons, in two dimensions. As an example, Figure 5.1d shows a grid with triangular cells.

The finite volume schemes can be designed entirely in terms of the cell-related quantities, such as the cell-averaged variables $|\Omega|^{-1}\int_\Omega \Phi d\Omega$, where $|\Omega|$ is the volume of the cell. It is, however, customary and convenient to introduce a *grid point* within each cell and write the schemes in terms of approximate values of variables at these points. One commonly used approach is the cell-centered arrangement illustrated in Figure 5.1. In each example, one grid point is shown and marked by the letter P. The grid point location coincides with the cell's center, so that the value of Φ

or another variable taken at the grid point serves as a good approximation of the mean value of this variable in the cell.

There are other possible arrangements. In some of them, the grid points are positioned so that the faces of the control cell are located midway between the two neighboring grid points. In others, the so-called cell-vertex schemes, the grid points are at the vertices of the cell boundaries. Our discussion will be limited to cell-centered arrangements. A broader and more detailed description can be found in the books focused on the subject of finite volume method—for example, in the books listed at the end of the chapter.

5.1.2 Global Conservation Property

The finite volume schemes possess one important property. Let us first illustrate it on the example of a one-dimensional problem. The conservation equation (5.1) is solved in the interval $0 \le x \le L$. The boundary conditions are set in the form of prescribed net fluxes q_0 at $x = 0$ and q_L at $x = L$. As discussed in section 5.4, such form of boundary conditions is convenient to use with the finite volume methods. The conservation of Φ in the entire solution domain can be expressed as

$$\frac{d}{dt}\int_0^L \Phi dx = \int_0^L Q dx + q_0 - q_L. \tag{5.2}$$

Note that this is an exact relation, according to which the total amount of Φ in the domain changes due to the internal sources and the fluxes through the boundaries.

The problem is solved using the finite volume grid consisting of N cells $[x_{i-1/2}, x_{i+1/2}]$ with $i = 1, \ldots, N$, $x_{1/2} = 0$, and $x_{N+1/2} = L$ (see Figure 5.2a). The conservation equation (5.1) is expressed for the one-dimensional cell Ω_i as

$$\frac{d}{dt}\int_{x_{i-1/2}}^{x_{i+1/2}} \Phi dx = -\Phi V|_{i+1/2} + \Phi V|_{i-1/2} + \chi \left.\frac{\partial \Phi}{\partial x}\right|_{i+1/2} - \chi \left.\frac{\partial \Phi}{\partial x}\right|_{i-1/2} + \int_{x_{i-1/2}}^{x_{i+1/2}} Q dx. \tag{5.3}$$

The equation has to be modified for the cells $\Omega_1 = (0, x_{3/2})$ and $\Omega_N = (x_{N-1/2}, L)$ adjacent to the boundaries of the computational domain. The

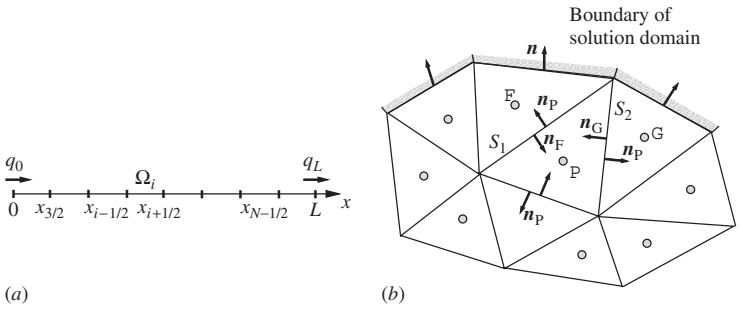


Figure 5.2 Illustration of the global conservation property of finite volume methods.

total (convective plus diffusive) fluxes are replaced by q_0 at $x = 0$ and by q_L at $x = L$. The integral equations for the boundary cells are

$$\frac{d}{dt}\int_0^{x_{3/2}} \Phi dx = -\Phi V|_{3/2} + \chi \left.\frac{\partial \Phi}{\partial x}\right|_{3/2} + \int_0^{x_{3/2}} Q dx + q_0 \qquad (5.4)$$

and

$$\frac{d}{dt}\int_{x_{N-1/2}}^{L} \Phi dx = \Phi V|_{N-1/2} - \chi \left.\frac{\partial \Phi}{\partial x}\right|_{N-1/2} + \int_{x_{N-1/2}}^{L} Q dx - q_L. \qquad (5.5)$$

Let us see how the global conservation relation (5.2) is reproduced by the finite volume scheme. We add the equations for all cells ((5.3) with $i = 2, \ldots, N-1$, (5.4), and (5.5)) together. The left-hand terms add up to $(d/dt)\int_0^L \Phi dx$ (remember that the cells do not overlap and cover the entire domain). The internal source terms give $\int_0^L Q dx$. It is easy to see that the flux terms at the internal cell-to-cell boundaries cancel out, since each of them appears with opposite signs in the equations for two neighboring cells. The only fluxes that remain are the fluxes q_0 and q_L through the boundaries of the computational domain.

We see that the summation of the finite volume equations over all cells results in (5.2) without any additional discretization-related terms. The principle of conservation is reproduced *exactly* and *for the entire computational domain*. This is the global conservation property shared by all finite volume schemes.

One can easily verify that the global conservation property is valid for two-dimensional and three-dimensional finite volume grids. A two-dimensional illustration is given in Figure 5.2b. Two neighboring cells marked by the grid points P and F have the common face S_1. The values

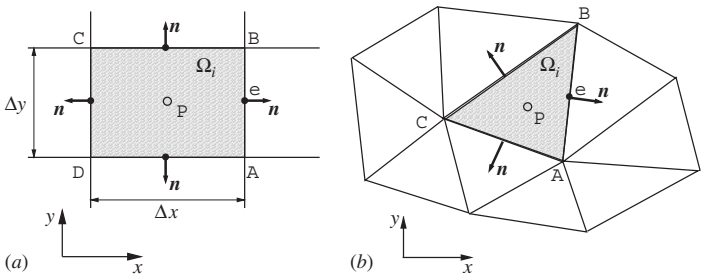


Figure 5.3 Illustration of the approximation of volume and surface integrals.

of the solution variables, such as Φ or $\boldsymbol{V}$, on S_1 are the same for both the cells. At the same time, the outward-facing unit normals denoted as $\boldsymbol{n}_{\mathrm{P}}$ and $\boldsymbol{n}_{\mathrm{F}}$ have opposite directions. The contributions of the surface integrals over S_1 in the integral equations for the cell P and the cell F are, therefore, equal in magnitude and opposite in sign. They cancel each other, when the equations are added together. Similarly, adding the equation for the cell marked G results in the cancelation of the surface integrals over the face S_2. If the sum over all cells is taken, only the surface integrals corresponding to the faces lying on the boundary of the computational domain remain and we obtain the exact replication of the global conservation principle

$$\frac{d}{dt}\int_{\Psi}\Phi d\Omega = -\int_{\partial\Psi}(\boldsymbol{q}\cdot\boldsymbol{n})dS + \int_{\Psi}Qd\Omega, \tag{5.6}$$

where Ψ is the computational domain, $\partial\Psi$ is its boundary, and $\boldsymbol{q}$ is the flux of Φ through the boundary.

5.2 APPROXIMATION OF INTEGRALS

The algebraic discretization equations that constitute a finite volume scheme are derived by approximating the integral equations written for the finite volume cells. The main elements of the approximation are discussed in this section. The discussion is general and valid for all kinds of finite volume cells. As illustrations, two-dimensional structured Cartesian and unstructured grids shown in Figure 5.3 are used.

5.2.1 Volume Integrals

The simplest way to approximate a volume integral is to replace it by the product of the cell's volume (in the three dimensional case) or area (in

two dimensions) $|\Omega|$ and the mean value of the integrand $\bar{\Phi}$ approximated through the grid point values. If the cell-centered arrangement of the grid points is used, we can replace the mean by the value at the central grid point (see Figure 5.3).

$$\int_{\Omega} \Phi d\Omega = \bar{\Phi}|\Omega| \approx \Phi_{\mathrm{P}}|\Omega|. \tag{5.7}$$

This approximation generates the truncation error, which has the second order of magnitude in terms of the size of the cell. For example, for the structured grid in Figure 5.3a, the dimensions of the cell are $(\Delta x, \Delta y)$, and the truncation error is $O((\Delta x)^2, (\Delta y)^2)$. If a finite volume scheme of the second order of accuracy is designed, (5.7) is sufficient and should be used. If, however, a scheme of a higher order is desired, $\bar{\Phi}$ has to be replaced by a more accurate approximation that uses the values of the integrand at other points within the cell, such as the vertex points A, B, C, and D or the midpoints of the cell faces, such as e. If these points do not belong to the grid (as is the case in a cell-centered arrangement), the values of the solution variables at them have to be interpolated from the neighboring grid points. The order of accuracy of the interpolation should, of course, not be lower than the desired order of the scheme. Interpolation techniques are discussed in section 5.3.

5.2.2 Surface Integrals

The surface $\partial\Omega$ of the cell Ω consists of several faces, which are curves in the two-dimensional and surfaces in the three-dimensional case. Their shape and number vary with the design of the grid. For example, in the two-dimensional quadrilateral cell of the structured grid shown in Figure 5.3a, the faces are the four intervals AB, BC, CD, and DA, while the cell in Figure 5.3b has three faces: AB, BC, and CA. Every surface integral in the cell equation (5.1) breaks down as a sum of integrals over the faces, which are computed separately. We will use the face AB for the demonstration. The procedure can be easily generalized to other faces, other shapes of the cell, and to the three-dimensional case.

The key component of the surface integral approximation is the midpoint rule. The integral is evaluated as a product of the area of the face and the mean value of the integrand. The mean is approximated by the value of the integrand at the midpoint of the face. In our example, this is expressed for an arbitrary integrand f as

$$\int_{\mathrm{AB}} f dS = \bar{f} S_{\mathrm{AB}} \approx f_{\mathrm{e}} S_{\mathrm{AB}}, \tag{5.8}$$

where S_{AB} is the area (length if the cell is two-dimensional) of the face, and the overline stands for the mean value on the face. The approximation generates the error $\sim O((S_{\mathrm{AB}})^2)$, which is the error of the second order in terms of the size of the cell.

Applied to the convective and diffusive flux integrals in (5.1), the approximations are

$$\int_{\mathrm{AB}} \Phi \boldsymbol{V} \cdot \boldsymbol{n} dS = \overline{\Phi \boldsymbol{V} \cdot \boldsymbol{n}} S_{\mathrm{AB}} \approx (\Phi \boldsymbol{V} \cdot \boldsymbol{n})_{\mathrm{e}} S_{\mathrm{AB}} \tag{5.9}$$

and

$$\int_{\mathrm{AB}} \chi \nabla \Phi \cdot \boldsymbol{n} dS = \overline{\chi \nabla \Phi \cdot \boldsymbol{n}} S_{\mathrm{AB}} \approx (\chi \nabla \Phi \cdot \boldsymbol{n})_{\mathrm{e}} S_{\mathrm{AB}}, \tag{5.10}$$

where e is the midpoint of the face AB.

The surface integrals contain projections of the solution vectors, such as $\boldsymbol{V}$ and $\nabla \Phi$, on the direction of the normal vector $\boldsymbol{n}$. If the grid is structured and the cell boundaries follow the lines of a Cartesian or curvilinear coordinate system, taking the projection is simple. For example, for the face AB in Figure 5.3a, we have $\boldsymbol{n}$ in the direction of the positive x-axis, and the projections are

$$(\Phi \boldsymbol{V} \cdot \boldsymbol{n})_{\mathrm{e}} = (\Phi V_x)_{\mathrm{e}}, \quad (\chi \nabla \Phi \cdot \boldsymbol{n})_{\mathrm{e}} = (\chi \partial \Phi / \partial x)_{\mathrm{e}}. \tag{5.11}$$

In many cases, the normal vector is not aligned with an axis of a global coordinate system. This is, in particular, true for unstructured grids as illustrated in Figure 5.3b. The evaluation of the projections requires both (or all three in three-dimensional problems) components of the solution vectors. For example, using the global Cartesian coordinate system shown in Figure 5.3b, we can write:

$$(\Phi \boldsymbol{V} \cdot \boldsymbol{n})_{\mathrm{e}} = \Phi_{\mathrm{e}} (V_x n_x + V_y n_y)_{\mathrm{e}}, \tag{5.12}$$

$$(\chi \nabla \Phi \cdot \boldsymbol{n})_{\mathrm{e}} = \chi_{\mathrm{e}} ((\partial \Phi / \partial x) n_x + (\partial \Phi / \partial y) n_y)_{\mathrm{e}}. \tag{5.13}$$

The orientation of the normal vector $\boldsymbol{n}$ is defined by the orientation of the corresponding cell face. This can be formalized by introducing the vector $\boldsymbol{S}$ connecting the end vertices of the face and defining $\boldsymbol{n}$ as the vector that is perpendicular to $\boldsymbol{S}$ and has unit length. In the two-dimensional examples in Figure 5.3, $\boldsymbol{S} = \boldsymbol{AB} = S_x \boldsymbol{i} + S_y \boldsymbol{j}$, where $\boldsymbol{i}$ and $\boldsymbol{j}$ are the base vectors of the Cartesian coordinate system. The outward-facing normal vector of unit length is

$$\boldsymbol{n} = \frac{1}{S_{\mathrm{AB}}} (S_y \boldsymbol{i} - S_x \boldsymbol{j}). \tag{5.14}$$

This allows us to rewrite the approximations (5.9) and (5.10) in a compact form

$$\int_{\mathrm{AB}} \Phi \boldsymbol{V} \cdot \boldsymbol{n} dS \approx \Phi_{\mathtt{e}} (V_{x\,\mathtt{e}} S_y - V_{y\,\mathtt{e}} S_x) \tag{5.15}$$

and

$$\int_{\mathrm{AB}} \chi \nabla \Phi \cdot \boldsymbol{n} dS \approx \chi_{\mathtt{e}} ((\partial \Phi / \partial x)_{\mathtt{e}} S_y - (\partial \Phi / \partial y)_{\mathtt{e}} S_x). \tag{5.16}$$

5.3 METHODS OF INTERPOLATION

As we have discussed in the previous section, the approximation of surface integrals requires knowledge of solution variables and their derivatives at the midpoints of the cell faces. These points are usually not part of the grid, and the values have to be obtained by *interpolation* from the grid points. Among the numerous possible schemes, we will only consider several, which are simple and commonly used. Further information can be found in the specialized texts on finite volume methods.

For simplicity, the methods of interpolation will be introduced using the two-dimensional Cartesian grid shown in Figure 5.4a. More specifically, we will show how the values of Φ and $\nabla \Phi \cdot \boldsymbol{n}$ at the face midpoint $\mathtt{e}$ can be approximated using the values of Φ at the grid points $\mathtt{P}$, $\mathtt{E}$, $\mathtt{W}$, $\mathtt{N}$, $\mathtt{S}$, $\mathtt{EE}$ and so on of a cell-centered grid. As even simpler examples, schemes for the one-dimensional linear convection and heat equations will be derived using the one-dimensional finite volume grid shown in Figure 5.4b. The principles of interpolation on nonorthogonal and unstructured grids will be briefly discussed in section 5.3.4.

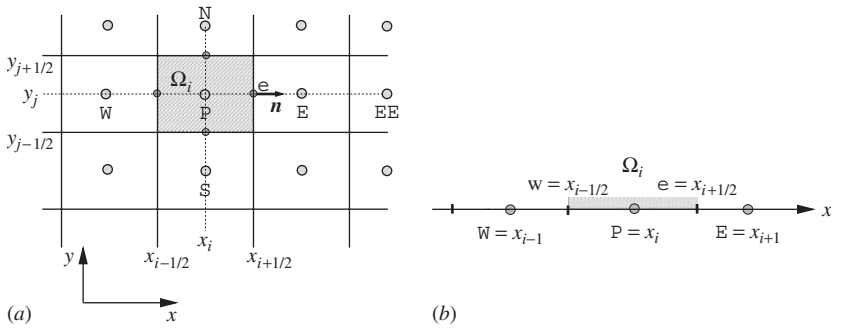


Figure 5.4 Two-dimensional Cartesian and one-dimensional grids used to illustrate interpolation methods.

5.3.1 Upwind Interpolation

The simplest method of approximation of Φ_e is to use the value at a neighboring grid point. In our example, this means approximating by either Φ_P or Φ_E. In the *upwind interpolation*, the choice, P or E, is dictated by the direction of the flow

$$\Phi_e = \begin{cases} \Phi_P & \text{if} \quad (\boldsymbol{V} \cdot \boldsymbol{n})_e > 0 \\ \Phi_E & \text{if} \quad (\boldsymbol{V} \cdot \boldsymbol{n})_e < 0 \end{cases}. \tag{5.17}$$

The value at the nearest upwind (upstream) grid point is taken. The choice seems natural since the upwind value is convected by the flow toward the point e. In agreement with this logic, the upwind schemes are usually considered in connection with hyperbolic problems, in which the convection velocity $\boldsymbol{V}$ determines the direction and speed of propagation of information in the solution.

When applied to hyperbolic problems, the schemes based on the upwind interpolation demonstrate valuable properties. In particular, the schemes satisfy the boundedness criterion, meaning that spurious oscillations never evolve in the solutions for propagating waves of sharp fronts. Another useful property is that the time integration of such schemes is numerically stable if the time step is sufficiently small. (The numerical stability is defined and discussed in Chapter 6.)

On the other hand, the procedure of upwind interpolation has one very significant drawback. It is of only the first order of accuracy. The truncation error of the upwind-based schemes contains strong numerical dissipation (see section 4.3.2 for definition and discussion). The sharp variations in the solution are artificially smeared out. This becomes a serious issue in predominantly hyperbolic problems, where there is no natural strong diffusion, and the sharp variations easily develop and persist. Very fine grids are needed to obtain accurate solutions of such problems using the upwind schemes.

Let us now derive an upwind interpolation scheme for the linear convection equation

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0, \quad c > 0. \tag{5.18}$$

The example seems appropriate, since the solutions of the linear convection equation are characterized by purely hyperbolic behavior. Integrating (5.18) over the cell Ω_i (see Figure 5.4b), we obtain

$$\frac{d}{dt}\int_{x_i-1/2}^{x_i+1/2} u dx + cu_e - cu_w = 0.$$

This integral form shows that the linear convection equation can be considered as a one-dimensional version of the general conservation equation (5.1) with constant velocity c in the positive x-direction and zero diffusion flux and volume sources. The volume integral is approximated according to (5.7) as

$$\int_{x_i-1/2}^{x_i+1/2} u dx \approx u_{\mathrm{P}} \Delta x.$$

The upwind interpolation is used for the midpoint values u_{e} and u_{w}. At the cell boundary $\mathrm{e} = x_{i+1/2}$, the normal $\boldsymbol{n}$ is in the positive x-direction, and (5.17) gives

$$u_{\mathrm{e}} \approx u_{\mathrm{P}}.$$

At the boundary $\mathrm{w} = x_{i-1/2}$, the normal is in the direction of negative x. We should take the value at the nearest grid point west of the cell:

$$u_{\mathrm{w}} \approx u_{\mathrm{W}}.$$

The finite volume approximation of (5.18) is

$$\frac{d}{dt}(u_{\mathrm{P}} \Delta x) + c u_{\mathrm{P}} - c u_{\mathrm{W}} = 0.$$

The last step is to introduce the time layers $t^n = t^0 + n\Delta t$ and apply the explicit time discretization of the first order. The result is

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c \frac{u_i^n - u_{i-1}^n}{\Delta x} = 0. \tag{5.19}$$

The scheme is identical to the upwind finite difference scheme (4.41).

5.3.2 Linear Interpolation

Another commonly used approximation is based on linear interpolation between two neighboring grid points. In terms of our example illustrated in Figure 5.4, this means

$$\Phi_{\mathrm{e}} = \gamma \Phi_{\mathrm{P}} + (1 - \gamma) \Phi_{\mathrm{E}}, \tag{5.20}$$

where $\gamma = |\mathrm{eE}|/|\mathrm{PE}|$ is the interpolation factor. This interpolation is of the second order of accuracy (see section 4.2.7). If the cells adjacent to the

face e are of the same size, the point e is exactly in the middle between P and E, $\gamma = 0.5$, and (5.20) turns into the simple averaging formula

$$\Phi_{\mathrm{e}} = \frac{\Phi_{\mathrm{P}} + \Phi_{\mathrm{E}}}{2}. \tag{5.21}$$

The linear interpolation can also be applied to approximate the normal derivative $(\nabla\Phi \cdot \boldsymbol{n})_{\mathrm{e}}$. In our example, $\boldsymbol{n}$ is in the positive x-direction, so the derivative can be evaluated as $\partial\Phi/\partial x$. Using the underlying assumption that the function behaves linearly around e, we find

$$\left(\frac{\partial\Phi}{\partial n}\right)_{\mathrm{e}} \approx \frac{\Phi_{\mathrm{E}} - \Phi_{\mathrm{P}}}{|\mathrm{PE}|} = \frac{\Phi_{\mathrm{E}} - \Phi_{\mathrm{P}}}{x_{\mathrm{E}} - x_{\mathrm{P}}}. \tag{5.22}$$

Let us analyze the order of approximation of this formula. Using the Taylor series expansion of Φ_{P} and Φ_{E} around the point e we find that the truncation error is

$$\begin{aligned}\text{T.E.} = {} & \frac{(x_{\mathrm{e}} - x_{\mathrm{P}})^2 - (x_{\mathrm{e}} - x_{\mathrm{E}})^2}{2(x_{\mathrm{E}} - x_{\mathrm{P}})}\left(\frac{\partial^2\Phi}{\partial x^2}\right)_{\mathrm{e}} \\ & - \frac{(x_{\mathrm{e}} - x_{\mathrm{P}})^3 + (x_{\mathrm{E}} - x_{\mathrm{e}})^3}{6(x_{\mathrm{E}} - x_{\mathrm{P}})}\left(\frac{\partial^3\Phi}{\partial x^3}\right)_{\mathrm{e}} + \cdots\end{aligned} \tag{5.23}$$

The first term in the right-hand side is of the first order with respect to the typical grid step, which we can estimate as the distance |EP| between the neighboring grid points. The interpolation scheme is formally of the first order. If the cells adjacent to the face e are of the same size, and e is in the middle of the interval PE, the first term is zero and the scheme acquires the second order of accuracy. In this case, (5.22) becomes the familiar central difference approximation of the first derivative. For this reason, the linear interpolation method is also called the *central difference* (CD) interpolation.

In the general case, when the cells are nonuniform, the first-order term in the truncation error is nonzero but becomes small when |Pe| and |eE| are close to each other. The CD scheme can be considered a scheme of nearly the second order on nonuniform grids if the variation of the grid size is not very strong.

In order to illustrate the application of the linear interpolation, we will develop a finite volume scheme for the purely parabolic one-dimensional version of (5.1):

$$\frac{\partial u}{\partial t} = a^2\frac{\partial^2 u}{\partial x^2}, \tag{5.24}$$

which is, of course, the heat equation. Integrating over the finite volume cell Ω_i of the grid in Figure 5.4b, we obtain

$$\frac{d}{dt}\int_{x_i-1/2}^{x_{i+1/2}} u dx = a^2\left(\frac{\partial u}{\partial x}\right)_{\mathrm{e}} - a^2\left(\frac{\partial u}{\partial x}\right)_{\mathrm{w}}.$$

The volume integral is approximated by $u_{\mathrm{P}}\Delta x$, while the x-derivatives are approximated using the linear interpolation as

$$\left(\frac{\partial u}{\partial x}\right)_{\mathrm{e}} \approx \frac{u_{\mathrm{E}} - u_{\mathrm{P}}}{\Delta x}, \quad \left(\frac{\partial u}{\partial x}\right)_{\mathrm{w}} \approx \frac{u_{\mathrm{P}} - u_{\mathrm{W}}}{\Delta x}.$$

Adding the explicit first order time discretization, we obtain

$$\frac{u_i^{n+1} - u_i^n}{\Delta t}\Delta x = a^2\frac{u_{i+1}^n - u_i^n}{\Delta x} - a^2\frac{u_i^n - u_{i-1}^n}{\Delta x}.$$

The formula can be rearranged so that it becomes identical to the familiar finite difference scheme for the heat equation (see, e.g. (4.33)):

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = a^2\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2}. \tag{5.25}$$

The fact that the finite volume schemes for the one-dimensional heat and linear convection equations coincide with the simple schemes developed following the finite difference approach is not at all surprising. The reason is the one-dimensionality of the problems and the obvious analogy between the finite volume grid shown in Figure 5.4b and the uniform finite difference grid used in Chapter 4. The unique character of the finite volume schemes usually appear when multidimensional problems are solved using unstructured grids.

5.3.3 Upwind Interpolation of Higher Order

A popular higher-order scheme for interpolation of Φ_{e} is the *Quadratic Upstream Interpolation of Convective Kinematics*, or QUICK. It has the third order of accuracy and is obtained by fitting a parabola through the values of the interpolated function at two grid points upstream and one point downstream of e. The resulting formulas are quite complex in the general case of nonorthogonal grids. In our example, however, the grid points are arranged along the coordinate lines, and the QUICK interpolation can be expressed by a simple formula. Let $(\boldsymbol{V}\cdot\boldsymbol{n})_{\mathrm{e}} > 0$, which

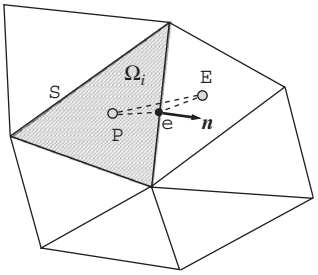


Figure 5.5 Effect of grid nonorthogonality on interpolation techniques.

means that the convective transport is in the direction of grid point E (see Figure 5.4a). Fitting a parabola through the values of Φ at the upstream points P and W and the downstream point E, we obtain

$$\Phi_e = \Phi_P + c_1(\Phi_E - \Phi_P) + c_2(\Phi_P - \Phi_W), \tag{5.26}$$

where

$$c_1 = \frac{(x_e - x_P)(x_e - x_W)}{(x_E - x_P)(x_E - x_W)}, \quad c_2 = \frac{(x_e - x_P)(x_E - x_e)}{(x_P - x_W)(x_E - x_W)}.$$

Other interpolation schemes can be used, some having the order of accuracy higher than three. Details can be found in the specialized literature. The schemes based on the high order interpolation often become excessively complex when used with nonuniform unstructured grids, and, therefore, are rarely applied in such cases.

5.3.4 Interpolation on Nonorthogonal Grids

The interpolation techniques become complex when the geometry of the problem requires curvilinear or unstructured grids. The complexity increases quite dramatically with the order of the approximation. For this reason, the finite volume schemes of the second order remain the most widely applied. The schemes of the orders higher than third are almost never used.

The main difficulty is illustrated in Figure 5.5. The line connecting the neighboring grid points P and E does not pass through the face midpoint e. Among the interpolation schemes just discussed, only the first-order upwind interpolation can be applied without modification. The formula (5.17) remains valid and accurate up to the discretization error of the order of the size of the cell.

As soon as the interpolation uses more than one grid point, the misalignment becomes a problem. For example, the linear interpolation (5.20) of Φ_e acquires an additional error and becomes formally an approximation of the first order. Various improvements of the method have been developed, description of which is available in the books listed at the end of the chapter. We mention one popular technique, according to which the piecewise-linear character of the line PeE is simply ignored. The linear interpolation formula (5.20) is applied in its original form. The additional first-order error is present, but it is small if the angle between eP and eE is close to 180°. Luckily, this situation is rather common in CFD, where it is a matter of good practice to maintain the neighboring cells at close size and aspect ratio.

Another problem appears when we attempt to approximate diffusive fluxes. The simple central difference formula (5.22) does not generate a second-order approximation of $(\partial\Phi/\partial n)_e$ even when |Pe| and |eE| are close, since the line EP is, in general, nonparallel to the normal $\boldsymbol{n}$ (see Figure 5.5). Similarly, since EP is nonparallel to the axes of the Cartesian coordinate system, we cannot use central differences to find second-order approximations of $(\partial\Phi/\partial x)_e$ and $(\partial\Phi/\partial y)_e$ and use them in (5.16).

Various methods have been developed to deal with the problem. One interesting approach is to evaluate the gradient at the cell-centered grid points and interpolate the results to the face midpoints. The grid-point gradients can be effectively estimated using the divergence theorem. For example, let us approximate $(\partial\Phi/\partial x)_P$. We apply the second-order approximation

$$\left(\frac{\partial\Phi}{\partial x}\right)_P \approx \frac{\int_\Omega(\partial\Phi/\partial x)d\Omega}{|\Omega|} \tag{5.27}$$

and transform the integral in the right-hand side using the divergence theorem:

$$\int_\Omega(\partial\Phi/\partial x)d\Omega = \int_\Omega \mathrm{div}(\Phi\boldsymbol{i})d\Omega = \int_S(\Phi\boldsymbol{i}\cdot\boldsymbol{n})dS = \int_S \Phi n_x dS$$
$$= \sum_j \int_{S_j} \Phi n_x dS,$$

where j is the index that marks the cell's faces. The surface integral over every face is replaced by the second order approximation

$$\int_{S_j} \Phi n_x dS \approx \Phi_{e_j} n_x S_j,$$

where Φ_{e_j} is the value of Φ at the corresponding midpoint. The final formula is

$$\left(\frac{\partial \Phi}{\partial x}\right)_{\mathrm{P}} \approx \frac{\sum_j \Phi_{e_j} n_x S_j}{|\Omega|}. \tag{5.28}$$

Note that, typically, the midpoint values Φ_{e_j} have to be calculated anyway to approximate the convective fluxes.

5.4 BOUNDARY CONDITIONS

Near the boundary of the computational domain, the boundary conditions should be incorporated into the integral balance equations for the cells and, thus, into the finite volume discretization. The special treatment concerns only the surface integrals over the faces lying on the boundary. The cumulative (convective plus diffusive) flux should be determined on the basis of the boundary conditions.

Let us consider the two-dimensional example shown in Figure 5.6. One face lying on the boundary is AB. We have to replace the surface integrals $-\int_{\mathrm{AB}} \Phi \boldsymbol{V} \cdot \boldsymbol{n} dS + \int_{\mathrm{AB}} \chi \nabla \Phi \cdot \boldsymbol{n} dS$ by an integral that gives the flux due to the boundary conditions. For the Neumann condition, when the normal component of the boundary flux $\boldsymbol{q}$ is prescribed, this can be done in a straightforward manner. We simply replace the surface integrals by

$$-\int_{\mathrm{AB}} (\boldsymbol{q} \cdot \boldsymbol{n}) dS = -\int_{\mathrm{AB}} q_n dS \approx -q_{n\mathrm{e}} S_{\mathrm{AB}}. \tag{5.29}$$

For the Dirichlet and Robin conditions, the flux is unknown and has to be approximated using the boundary conditions and values of Φ at interior points. For example, in the case of the Dirichlet condition, when Φ at the

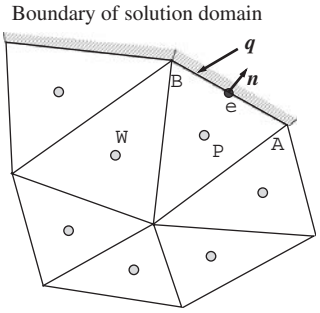


Figure 5.6 Treatment of boundary conditions in finite volume methods.

face AB is prescribed, we assume that the flux is provided by a diffusive mechanism activated by the gradient of Φ at the boundary. The boundary flux is

$$\int_{\mathrm{AB}} (\boldsymbol{q} \cdot \boldsymbol{n}) dS = -\int_{\mathrm{AB}} \chi \nabla \Phi \cdot \boldsymbol{n} dS \approx -\chi \left(\frac{\partial \Phi}{\partial n} \right)_{\mathrm{e}} S_{\mathrm{AB}}. \tag{5.30}$$

To approximate the gradient of Φ we can use the scheme of the first order

$$\left(\frac{\partial \Phi}{\partial n} \right)_{\mathrm{e}} \approx \frac{\Phi_{\mathrm{e}} - \Phi_{\mathrm{P}}}{|\mathrm{Pe}|} \tag{5.31}$$

or the interpolation of higher order, which uses values of Φ at more than one interior grid points.

REFERENCES AND SUGGESTED READING

Blazek, J. *Computational Fluid Dynamics: Principles and Applications*. Amsterdam: Elsevier, 2005.

Versteeg, H., and W. Malalasekra. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*, Upper Saddle River, NJ: Prentice Hall, 2007.

PROBLEMS

1. Transform the following equations into the integral form similar to (5.1). For each equation, identify, if present, the rate of change term, volumetric source term, convective flux term, and diffusive flux term.

a) Linear convection equation $u_t + cu_x = 0$, where $c > 0$ is a constant

b) One-dimensional heat equation $u_t = a^2 u_{xx} + f(x)$

c) Three-dimensional heat equation $u_t = \nabla^2 u + g(\boldsymbol{x})$ (*Hint:* Use the identity $\nabla^2 u = \mathrm{div}(\nabla u)$ and the divergence theorem.)

d) Three-dimensional Poisson equation $\nabla^2 u = g(\boldsymbol{x})$

e) One-dimensional Burgers equation $u_t + uu_x = \mu u_{xx}$, where $\mu > 0$ is a constant

2. A finite volume scheme is developed using a two-dimensional structured Cartesian grid with constant steps Δx and Δy (see Figure 5.4a).

Write the following approximations using the values of the function u at the grid points, such as P, E, EE, and W:

a) Approximation of the second order for the volume integral $\int_{\Omega_i} u d\Omega$

b) Upwind approximation for the surface integral $\int_{S_e} u\boldsymbol{V} \cdot \boldsymbol{n} dS$, where S_e is the face containing the point e, $\boldsymbol{V}$ is the constant velocity $\boldsymbol{V} = (1, 0.5)$, and $\boldsymbol{n}$ is the outward-facing unit normal to S_e

c) Central difference approximation of the second order for the surface integral $\int_{S_e} (\partial u/\partial x) dS$

3. The Burgers equation $u_t + uu_x = \mu u_{xx}$ is solved by the finite volume method on a one-dimensional grid with constant step Δx (see Figure 5.4b). Develop the schemes based on the following principles:

a) Upwind interpolation for convective flux, linear interpolation for diffusive flux

b) Linear interpolation for convective and diffusive fluxes

4. The conservation equation (5.1) is solved by the finite volume method on a two-dimensional structured Cartesian grid with constant steps Δx and Δy (see Figure 5.4a). Develop the schemes based on the following principles:

a) Upwind interpolation for convective flux, linear interpolation for diffusive flux

b) Linear interpolation for convective and diffusive fluxes.

6

STABILITY OF TRANSIENT SOLUTIONS

6.1 INTRODUCTION AND DEFINITION OF STABILITY

Numerical stability is an essential ingredient of a successful computational solution of any marching problem. The absence of this ingredient renders the solution completely useless, as illustrated by the following example.

Example We return to our first finite difference scheme introduced at the end of Chapter 3 and further discussed in Chapter 4. The PDE problem for the inhomogeneous heat equation with Dirichlet boundary conditions

$$\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2} + \sin 5x, \quad u(0,t) = 0, \quad u(L,t) = 0, \quad u(x,0) = x(\pi - x) \tag{6.1}$$

is solved in the domain $0 < x < L = \pi$, $0 < t < 50$ with $a = 0.5$. The uniform grid $x_i = i\,\Delta x$, $t^n = n\,\Delta t$ with $i = 0, \ldots, N = 100$, $n = 0, \ldots, M$ is used. The steps are $\Delta x = L/N$ and $\Delta t = 0.01$ or $\Delta t = 10^{-4}$. The finite difference scheme is explicit and based on the first-order forward difference in time and second-order central difference in space:

$$u_i^{n+1} = u_i^n + \Delta t \left(a^2 \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} + f_i^n \right)$$
$$\text{at} \quad i = 1, \ldots, N-1, \; n = 1, \ldots, M-1. \tag{6.2}$$

The boundary and initial conditions are discretized as

$$u_0^n = 0, \quad u_N^n = 0, \quad \text{at} \quad n = 1, \ldots, M$$
$$u_i^0 = x_i(\pi - x_i) \quad \text{at} \quad i = 0, \ldots, N.$$

The problem has an analytical solution, according to which $u(x,t)$ differs from the equilibrium solution by less than 10^{-5} at $t = 50$. We accept this error and consider the equilibrium solution at $t = 50$ as the exact solution of the problem. The solution can be easily found by assuming $\partial u/\partial t = 0$, which leads to

$$a^2 \frac{\partial^2 u_{\text{exact}}}{\partial x^2} = -\sin 5x,$$

so

$$u_{\text{exact}}(x) = \frac{1}{25a^2} \sin 5x = 0.16 \sin 5x. \tag{6.3}$$

We can evaluate the accuracy of the numerical solution by calculating absolute errors

$$\epsilon_{\text{abs}}(x_i) = \left| u_{\text{exact}}(x_i) - u_i^M \right|, \quad i = 0, \ldots, N \tag{6.4}$$

and relative errors

$$\epsilon_{\text{rel}}(x_i) = \frac{\epsilon_{\text{abs}}(x_i)}{u_{\text{mean}}}, \quad \text{where} \quad u_{\text{mean}} = \left(\frac{1}{N+1} \sum_{i=0}^{N} (u_i^M)^2 \right)^{1/2}. \tag{6.5}$$

The results are presented in Figure 6.1. The computed u is shown as a function of time at $x = x_{50} = L/2$ in Figure 6.1a,c and as a function of x at a fixed time in Figure 6.1b,d.

Let us first consider the solution obtained with $\Delta t = 0.01$. The results presented in Figure 6.1a,b are obviously incorrect. Soon after the start, the solution begins to grow rapidly and reaches the upper limit set by the computer's memory. In the state calculated at $t = 2$ (see Figure 6.1b), we see wild oscillations that have nothing in common with the exact solution (6.3). The apparent amplitude of 10^{36} is, in fact, the limit allowed by the plotting software.

By contrast, when the solution is repeated with the same Δx but the time step reduced to $\Delta t = 10^{-4}$, the behavior is much more reasonable (see Figure 6.1c,d). The numerical solution closely follows the analytical exact solution. At $t = 50$, the relative error (6.5) is less than 10^{-4}.

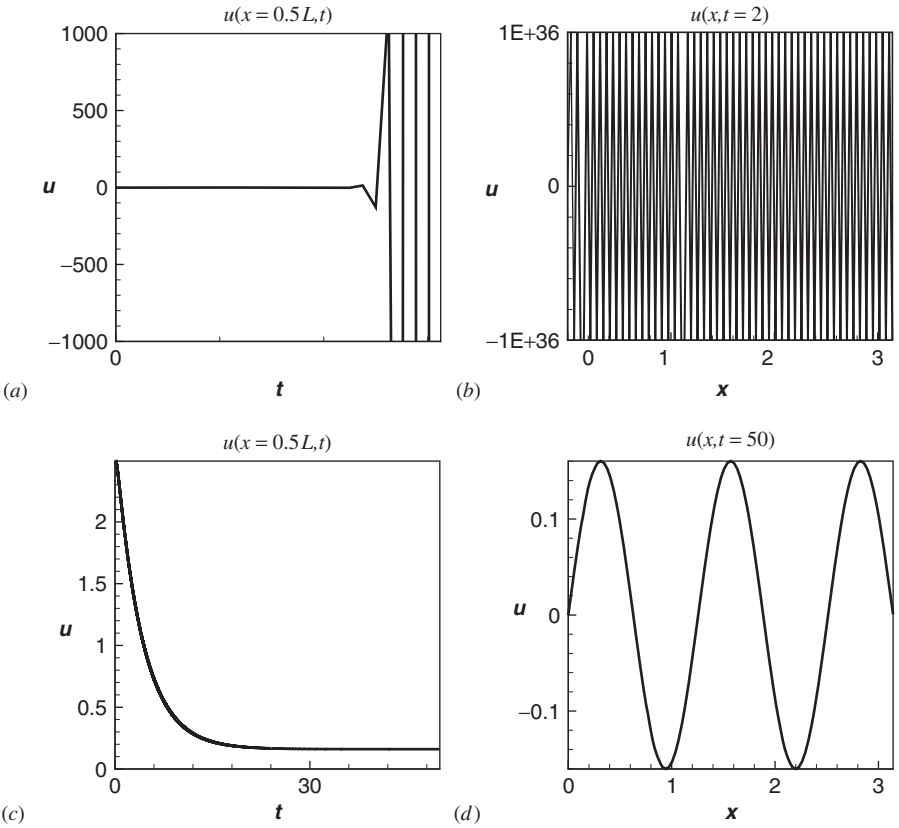


Figure 6.1 Solution of problem (6.1) using finite difference scheme (6.2). Grid step is $\Delta x = 0.01\pi$. Time step is $\Delta t = 0.01$ in *(a), (b)* and $\Delta t = 10^{-4}$ in *(c), (d)*.

Why does the size of the time step have such a dramatic impact on the solution? The simple explanation that we observe loss of accuracy because of larger truncation error does not stand the critique. The truncation error of time discretization at $\Delta t = 0.01$ is $O(\Delta t) \sim 10^{-2}$, which is very far from 10^{36}. The real answer is the unbounded growth of the round-off errors of computer calculations, as discussed in the rest of this chapter.

6.1.1 Discretization and Round-off Error

Let us consider more carefully the errors that are generated in the process of numerical solution of a PDE problem. The following discussion is general in the sense that it applies to all kinds of discretization and all time-dependent PDE, although we will use the finite difference approximation (6.2) of the heat equation (6.1) as an example.

There are two kinds of numerical error. First, the numerical approximation differs from the PDE and boundary conditions by the truncation error. We learned in the previous chapters that its amplitude depends on the approximation scheme and the size of the grid steps. For example, the finite difference scheme (6.2) represents the heat equation with the truncation error of the magnitude $O(\Delta t, (\Delta x)^2)$. The truncation results in the error of the numerical solution called the *discretization error*. It is defined as the difference between the exact solution of the system of algebraic equations generated by the numerical scheme and the exact analytical solution of the PDE problem.

Another source of error is the inability of a computer to solve the algebraic equations, such as (6.2), exactly. There are always *round-off errors* associated with the fact that any computer performs arithmetic operations using a finite number of digits. The number varies depending on the computational platform and programming instructions, but it is always finite. The corresponding round-off errors are normally very small. If, however, they accumulate as a marching procedure advances in time, the effect can be quite dramatic.

Let us introduce the notation:

ua is the exact analytical solution of the PDE problem such as (6.1).

ud is the exact solution of the system of discretization equations such as (6.2).

un is the actually computed solution of the system of discretization equations.

The discretization error is $\boldsymbol{ud} - \boldsymbol{ua}$, and the round-off error is $\boldsymbol{\epsilon} = \boldsymbol{un} - \boldsymbol{ud}$. The actually computed solution differs from the exact solution of the discretized problem as

$$\boldsymbol{un} = \boldsymbol{ud} + \boldsymbol{\epsilon}. \tag{6.6}$$

The discretization error has the same order of magnitude as the truncation error of the scheme. In our example, this means $O(\Delta t, (\Delta x)^2)$. We can conclude that the discretization error cannot be the reason of enormous inaccuracy illustrated in Figure 6.1a,b. The next step is to analyze the round-off error.

6.1.2 Definition

Let us consider one time step of a marching procedure. It advances the solution from the time layer t^n to the time layer t^{n+1}. We disregard the details for the moment and view the procedure as a *black box* illustrated in Figure 6.2. It takes the numerical solution at t^n as an input and generates the solution at t^{n+1} as an output.

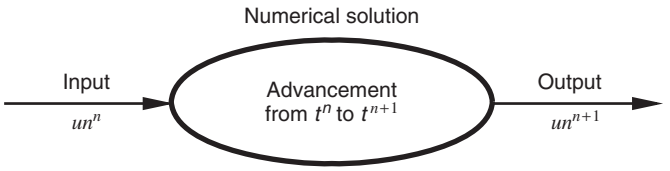


Figure 6.2 A time step of a marching scheme considered as a black box.

The main question is, what happens to the round-off errors as they pass through the black box? If they amplify, their growth and accumulation are likely to bring failure to the solution. This situation is called *instability*. If they decay or stay limited, their effect can be neglected. In this case, we have *stability*.

The formal definition of stability can be done in several ways. One possibility is to define *weak* stability. According to this definition, a marching scheme is unstable if the amplitude of the round-off error ϵ increases at, at least, one grid point as the solution passes from the time layer t^n to the time layer t^{n+1}. Otherwise, the scheme is called stable. Let us use the notation ϵ_i^n for the round-off error at time layer t^n and grid point x_i. The scheme is stable if

$$\left| \frac{\epsilon_i^{n+1}}{\epsilon_i^n} \right| \leq 1 \tag{6.7}$$

at every point x_i and is unstable if there is a point where the condition is violated.

Another definition is that of *strong* stability, which requires that the overall round-off error in the entire space domain does not grow with time. This can be expressed in terms of the vector norm of the error $\|\boldsymbol{\epsilon}^n\| = \left(N^{-1} \sum_{i=1}^{N} (\epsilon_i^n)^2\right)^{1/2}$ as

$$\frac{\|\boldsymbol{\epsilon}^{n+1}\|}{\|\boldsymbol{\epsilon}^n\|} \leq 1. \tag{6.8}$$

6.2 STABILITY ANALYSIS

6.2.1 Neumann Method

Among the methods used to analyze the stability, the Neumann method based on Fourier expansions is the most widely used. The basic idea is to assume the actual numerical solution in the form (6.6), feed it as the input stream into the black box in Figure 6.2, and use the Fourier expansion to analyze the behavior of ϵ.

We will employ the scheme (6.2) for demonstration. First, note that both the actually computed (rounded-off) ***un*** and the exact solution of the numerical problem ***ud*** satisfy the finite difference equation (6.2). Writing (6.6) at every grid point

$$un_i^n = ud_i^n + \epsilon_i^n, \tag{6.9}$$

and substituting into (6.2) we obtain

$$\frac{ud_i^{n+1} + \epsilon_i^{n+1} - ud_i^n - \epsilon_i^n}{\Delta t} = a^2 \frac{ud_{i+1}^n + \epsilon_{i+1}^n - 2ud_i^n - 2\epsilon_i^n + ud_{i-1}^n + \epsilon_{i-1}^n}{(\Delta x)^2} + f_i^n. \tag{6.10}$$

The exact solution ***ud*** also satisfies (6.2):

$$\frac{ud_i^{n+1} - ud_i^n}{\Delta t} = a^2 \frac{ud_{i+1}^n - 2ud_i^n + ud_{i-1}^n}{(\Delta x)^2} + f_i^n. \tag{6.11}$$

Subtraction of (6.11) from (6.10) yields the equation for the round-off error

$$\frac{\epsilon_i^{n+1} - \epsilon_i^n}{\Delta t} = a^2 \frac{\epsilon_{i+1}^n - 2\epsilon_i^n + \epsilon_{i-1}^n}{(\Delta x)^2}. \tag{6.12}$$

The particular form (6.12) of the error equation is for the particular PDE (heat equation (6.1)) and finite difference scheme (simple explicit scheme (6.2)) used in our example. The principal procedure, substitution of ***un*** into the finite difference equation and subtraction of the equation for ***ud*** is, however, universal. In general, the final equation may contain ***ud*** and variable coefficients, but contains only $\boldsymbol{\epsilon}$ if the finite difference scheme is linear.

Note that the inhomogeneous term in the right-hand side of (6.1) does not appear in the equation for the error (6.12) and, thus, does not affect the stability. Our analysis would be the same if we considered any other inhomogeneous term or the homogeneous heat equation. In general, the inhomogeneous sourcelike terms can be neglected in the stability analysis if they are completely independent of the solution u. They, however, should be taken into account in the opposite case.

The equation (6.12) can be considered as a finite difference representation of the linear PDE

$$\frac{\partial \epsilon}{\partial t} = a^2 \frac{\partial^2 \epsilon}{\partial x^2}, \tag{6.13}$$

which can be solved analytically using the method of separation of variables. We will follow this approach, neglect the effect of boundary conditions, and write the general solution as

$$\epsilon(x,t) = \sum_m b_m(t) e^{\imath k_m x} + \overline{c.c}, \tag{6.14}$$

where $\overline{c.c}$ stands for complex conjugate of the first term in the right-hand side. Formula (6.14) allows some interpretation. The round-off error at a given time layer t^n is an irregular function of x like that shown in Figure 6.3. The series (6.14) is the decomposition of this function into Fourier harmonics. We know from the theory of Fourier series that such a decomposition is possible for any piecewise continuous function and that the wavenumbers are

$$k_m = \frac{2\pi m}{L}, \quad m = 1, 2, 3, \ldots. \tag{6.15}$$

One correction of the classical Fourier theory is needed. In the numerical solution we deal not with a continuous function $\epsilon(x,t)$ but, rather, with its values at discrete points x_i separated from each other by the distance Δx. As illustrated in Figure 6.4, the harmonics with the wavelength smaller than $2\Delta x$ cannot be identified on the grid. We have to limit the series (6.14) by the maximum discernable wavenumber

$$k_{\max} = \frac{2\pi}{2\Delta x} = \frac{\pi}{\Delta x} = \frac{2\pi}{L}\frac{N}{2}, \tag{6.16}$$

where N is the number of the grid points. The set of the wavenumbers in (6.15) is, therefore, limited by $m = N/2$. The corrected decomposition formula is

$$\epsilon(x,t) = \sum_{m=1}^{N/2} b_m(t) e^{\imath k_m x} + \overline{c.c}, \quad k_m = \frac{2\pi m}{L}. \tag{6.17}$$

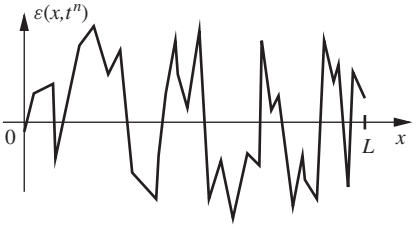


Figure 6.3 Round-off error as a function of x.

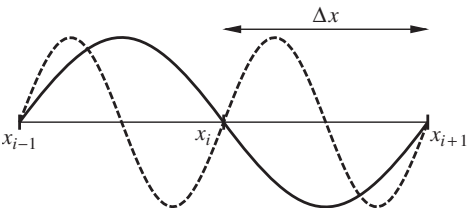


Figure 6.4 Solid curve illustrates the shortest Fourier wave discernable on a grid with step Δx. It has the wavelength $2\Delta x$. The waves with smaller wavelengths cannot be identified on the grid. For example, presence of the harmonic with the wavelength Δx shown by the dashed curve remains unrecognized since the harmonic has the same values at the grid points as the long-wave harmonic.

The growth or decay of the round-off error is determined by the behavior of the coefficients $b_m(t)$. If, for at least one Fourier mode $\epsilon_m(x,t) = b_m(t)e^{\imath k_m x}$, the amplitude grows with time, the entire error grows, and the scheme is unstable. In the case when the PDE and the numerical approximation equation are linear, the behavior of $b_m(t)$ can be determined analytically, since the error equations, such as (6.12) and (6.13), are also linear. Substitution of (6.17) into (6.13) shows that the PDE equation can be separated into $N/2$ equations, one for every Fourier mode and its conjugate. Furthermore, each of these separate equations has an exponential solution

$$\epsilon_m(x,t) = e^{c_m t}e^{\imath k_m x}, \tag{6.18}$$

where c_m is a yet unknown complex coefficient. We will analyze the behavior of c_m using the finite difference approximation (6.12) of the error equation.

The stability criterion (6.7) says that the scheme is stable if there are no grid points where ϵ is amplified. We translate this as a requirement that none of the Fourier modes ϵ_m is amplified and see immediately that the coefficient c_m provides all the necessary information. The amplification of the mode ϵ_m is the same at all grid points and is given by

$$\left|\frac{\epsilon_i^{n+1}}{\epsilon_i^n}\right| = \left|\frac{\epsilon^{c_m(t^n+\Delta t)}e^{\imath k_m x_i}}{\epsilon^{c_m t^n}e^{\imath k_m x_i}}\right| = \left|e^{c_m \Delta t}\right|. \tag{6.19}$$

This quantity is called the *amplification factor*:

$$G_m = \left|e^{c_m \Delta t}\right|. \tag{6.20}$$

The stability criterion can be reformulated as follows: *The scheme is stable if the condition*

$$G_m = \left| e^{c_m \Delta t} \right| \leq 1, \tag{6.21}$$

is satisfied for all $m = 1, \ldots, N/2$.

We will now return to our example and do some algebra. At the grid points, the solution (6.18) is

$$\epsilon_i^n = e^{c_m t^n} e^{\imath k_m x_i}, \quad \epsilon_i^{n+1} = e^{c_m (t^n + \Delta t)} e^{\imath k_m x_i} \tag{6.22}$$

$$\epsilon_{i-1}^n = e^{c_m t^n} e^{\imath k_m (x_i - \Delta x)}, \quad \epsilon_{i+1}^n = e^{c_m t^n} e^{\imath k_m (x_i + \Delta x)}, \tag{6.23}$$

and the finite difference equation for the mth mode of the round-off error is

$$\frac{e^{c_m (t^n + \Delta t)} e^{\imath k_m x_i} - e^{c_m t^n} e^{\imath k_m x_i}}{\Delta t} = a^2 \frac{e^{c_m t^n} e^{\imath k_m (x_i - \Delta x)} - 2 e^{c_m t^n} e^{\imath k_m x_i} + e^{c_m t^n} e^{\imath k_m (x_i + \Delta x)}}{(\Delta x)^2}. \tag{6.24}$$

Dividing by $e^{c_m t^n} e^{\imath k_m x_i}$ we obtain

$$\frac{e^{c_m \Delta t} - 1}{\Delta t} = a^2 \frac{e^{-\imath k_m \Delta x} - 2 + e^{\imath k_m \Delta x}}{(\Delta x)^2}, \tag{6.25}$$

which can be regrouped as

$$e^{c_m \Delta t} = 1 + \frac{a^2 \Delta t}{(\Delta x)^2} \left(e^{-\imath k_m \Delta x} - 2 + e^{\imath k_m \Delta x} \right). \tag{6.26}$$

We use the identity

$$\cos(k_m \Delta x) = \frac{e^{\imath k_m \Delta x} + e^{-\imath k_m \Delta x}}{2}, \tag{6.27}$$

so (6.26) becomes

$$e^{c_m \Delta t} = 1 + \frac{2a^2 \Delta t}{(\Delta x)^2} (\cos(k_m \Delta x) - 1) = 1 - \frac{4a^2 \Delta t}{(\Delta x)^2} \sin^2 \left(\frac{k_m \Delta x}{2} \right). \tag{6.28}$$

It is convenient to introduce the fictitious angle $\beta = k_m \Delta x$ and use it instead of the wavenumber index m. With the wavenumber k_m running

from $2\pi/L$ to $N\pi/L$ and $\Delta x = L/N$ (see (6.15) and (6.16)), the angle is always within the limits

$$\frac{2\pi}{N} \leq \beta \leq \pi. \tag{6.29}$$

The stability criterion is formulated in terms of β as

$$G(\beta) \leq 1 \text{ for all } \beta. \tag{6.30}$$

The rest is easy. We use (6.21) and (6.20) with the abbreviation

$$r = \frac{a^2 \Delta t}{(\Delta x)^2} \tag{6.31}$$

to express the stability criterion as

$$\left|1 - 4r \sin^2\left(\frac{\beta}{2}\right)\right| \leq 1, \text{ or } \; -1 \leq 1 - 4r \sin^2\left(\frac{\beta}{2}\right) \leq 1,$$

$$\text{or} \quad \begin{cases} 4r \sin^2(\beta/2) & \geq & 0 \\ 4r \sin^2(\beta/2) & \leq & 2 \end{cases}$$

Since r is always positive and $0 \leq \sin^2(\beta/2) \leq 1$, the first condition is satisfied automatically. The second condition gives the *stability criterion for the simple explicit scheme (6.2) for the heat equation:*

$$r = \frac{a^2 \Delta t}{(\Delta x)^2} \leq \frac{1}{2}. \tag{6.32}$$

We are now prepared to answer the question of what was wrong with the first of the numerical solutions shown in Figure 6.1. At $\Delta x = 0.01\pi$, $a = 0.5$, and $\Delta t = 0.01$, we have $r \approx 2.54$, which is far above the stability limit. What is seen in the top two plots of Figure 6.1 is the typical example of a numerically unstable solution. For the second simulation, we use $\Delta t = 10^{-4}$, which corresponds to $r \approx 0.025$, The solution is stable, and accurate results are obtained as shown in the other two plots of Figure 6.1. We can evaluate the maximum time step at which the scheme is stable as

$$\Delta t_{\max} = \frac{(\Delta x)^2}{2a^2} \approx 1.97 \cdot 10^{-3}.$$

It is not always possible to find an analytical estimate of the upper bound of $G(\beta)$, as we did in (6.32). Numerical evaluation may be needed. Furthermore, the information on the amplification of the round-off error at different wavelengths—that is, the information on the behavior of the function $G(\beta)$—can be useful for understanding the properties of a finite difference scheme. It is, therefore, convenient and customary to plot the entire $G(\beta)$ in Cartesian or, as illustrated in Figure 6.5, in polar coordinates. In the latter case, β and G serve as the polar angle and radius, respectively. A scheme is deemed stable if the entire curve $G(\beta)$ at $0 < \beta < \pi$ lies within the unit radius circle $G = 1$. For example, curves in Figure 6.5 show that the simple explicit scheme (6.2) applied to the one-dimensional heat equation is stable at $r = 0.4$ and unstable at $r = 0.6$.

The stability results (6.28) and (6.32) are only valid for the particular example when the scheme (6.2) is applied to the heat equation, but the procedure we used to arrive at these results is universal and can be used for other schemes and other equations. Let us reiterate the main steps:

- The equation for round-off error is derived by substituting the expression (6.9) into the finite difference scheme.
- Fourier decomposition (6.14) is assumed for the round-off error.
- The error equation is solved for separate Fourier modes to find the amplification factor $G(\beta)$.
- The stability criterion such as, for example (6.32), is derived.

The comment is in order concerning whether we can apply the methods and results of the Neumann stability analysis developed for simple model

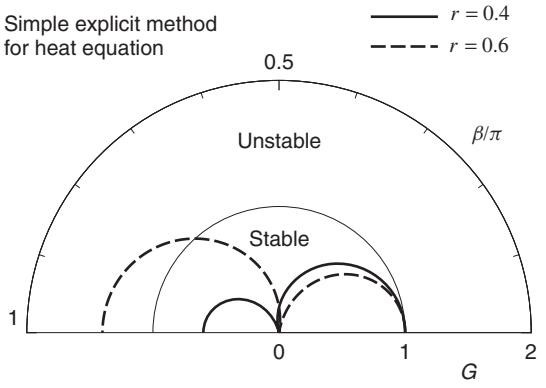


Figure 6.5 Amplification factor G for simple explicit scheme applied to heat equation (6.1).

problems, such as (6.1), to significantly more complex equations such as the full Navier-Stokes system. The mathematically rigorous answer is no. Most importantly, the model equations are linear with constant coefficients. Only for such systems the exponential solution (6.18) exists. On the contrary, the realistic equations of fluid flows and convective heat transfer are almost always nonlinear and, often, include variable coefficients. We will return to this question in the following chapters and show that the stability of time integration of complex equation can be analyzed in approximate sense using the criteria derived for model systems.

Let us consider another example—the fully implicit scheme for the heat equation (6.1). The scheme was introduced in Chapter 4. It reads

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = a^2 \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2} + f_i^{n+1}. \tag{6.33}$$

Following the procedure of the Neumann stability analysis (see (6.9)–(6.12)), we derive the equation for the round-off error

$$\frac{\epsilon_i^{n+1} - \epsilon_i^n}{\Delta t} = a^2 \frac{\epsilon_{i+1}^{n+1} - 2\epsilon_i^{n+1} + \epsilon_{i-1}^{n+1}}{(\Delta x)^2}. \tag{6.34}$$

Substituting (6.22) and (6.23) and dividing by $e^{c_m t^n} e^{\imath k_m x_i}$ we obtain

$$\frac{e^{c_m \Delta t} - 1}{\Delta t} = a^2 e^{c_m \Delta t} \frac{e^{-\imath k_m \Delta x} - 2 + e^{\imath k_m \Delta x}}{(\Delta x)^2}, \tag{6.35}$$

which can be rewritten as

$$e^{c_m \Delta t} \left[1 - r \left(e^{-\imath k_m \Delta x} - 2 + e^{\imath k_m \Delta x} \right) \right] = 1, \tag{6.36}$$

or, with (6.27),

$$e^{c_m \Delta t} \left[1 + 2r - 2r \cos \beta \right] = 1. \tag{6.37}$$

Using the trigonometric identity we find the amplification factor as

$$G(\beta) = \frac{1}{1 + 4r \sin^2(\beta/2)}. \tag{6.38}$$

The plot of $G(\beta)$ is shown in Figure 6.6. It illustrates the fact that *the stability condition* $G(\beta) \leq 1$ *is satisfied for any* $r \geq 0$. This conclusion is typical for implicit schemes. Many of them (but not all!) are *unconditionally stable* (i.e., stable for any choice of the time and space discretization steps).

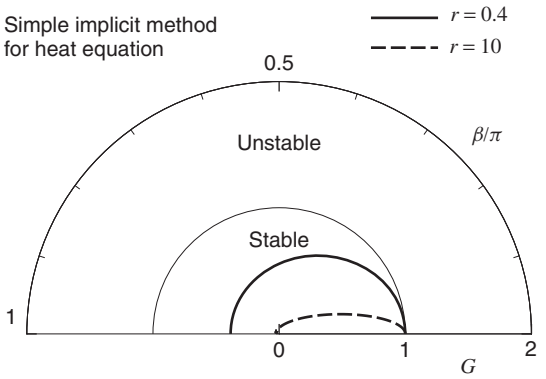


Figure 6.6 Amplification factor G for simple implicit scheme applied to heat equation (6.1).

6.2.2 Matrix Method

The matrix method is another approach to the stability analysis. It can be used for two-layer schemes, i.e. for the schemes with equations containing values of u at not more than two consecutive time layers. The equation for the round-off error, such as, for example, (6.12) can be expressed in the matrix form:

$$\boldsymbol{\epsilon}^{n+1} = \mathbf{A} \cdot \boldsymbol{\epsilon}^n \quad \text{or} \quad \boldsymbol{\epsilon}^{n+1} = \underbrace{\mathbf{A} \cdot \mathbf{A} \cdots \mathbf{A}}_{n+1} \cdot \boldsymbol{\epsilon}^0 = \mathbf{A}^{n+1} \cdot \boldsymbol{\epsilon}^0, \tag{6.39}$$

where $\mathbf{A}$ is a square matrix $N \times N$. In our example of a simple explicit scheme (6.2) applied to the heat equation, the matrix has zero elements except for the three main diagonals:

$$\mathbf{A} = \begin{pmatrix} (1-2r) & r & 0 & \dots & \dots & 0 \\ r & (1-2r) & r & 0 & \dots & 0 \\ 0 & r & (1-2r) & r & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & 0 & r & (1-2r) & r \\ 0 & \dots & \dots & 0 & r & (1-2r) \end{pmatrix},$$

where $r = a^2 \Delta t / (\Delta x)^2$. In the cases when the boundary conditions affect the stability, their approximation is included into the matrix.

It can be shown that the vector norm $\|\boldsymbol{\epsilon}^{n+1}\|$ of the round-off error evolving according to (6.39) remains bounded if the eigenvalues of $\mathbf{A}$ are

all different and have absolute values less or equal than 1:

$$|\lambda_m| \leq 1, \ \forall m = 1, \ldots, N. \tag{6.40}$$

In our example, the tridiagonal form of the matrix simplifies the task of finding the eigenvalues. There is a direct formula

$$\lambda_m = 1 - 4r \sin^2 \left[\frac{m\pi}{2(N+1)} \right].$$

In more general cases, when the analytical formulas do not exist, the eigenvalues can be found using one of the approximate numerical methods. The stability criterion (6.40) leads to

$$-1 \leq 1 - 4r \sin^2 \left[\frac{m\pi}{2(N+1)} \right] \leq 1.$$

The right-hand condition is always satisfied, while the left-hand condition gives

$$r \sin^2 \left[\frac{m\pi}{2(N+1)} \right] \leq \frac{1}{2}.$$

The inequality is true for any m if

$$r \leq \frac{1}{2}. \tag{6.41}$$

The Neumann and matrix methods deal with the same subject, the round-off error, and have the same purpose, to establish conditions under which the error remains bounded. There are some differences, which are briefly discussed here.

The stability criteria produced by the two methods can, in general, be different. This is not a serious concern for us because they coincide in almost all practically important cases.

There is a difference in the range of applicability. The matrix method just described can only be applied to two-layer schemes. The limitation is quite strong since, as we will see in the next chapters, there are many powerful and popular multilevel schemes, in which the finite difference equations connect values of u at three or more time layers. On the contrary, the Neumann method can be applied, at least theoretically, to schemes with any number of layers.

The matrix method has an advantage that it can include the boundary conditions of Dirichlet and Neumann type into the stability analysis. The Neumann method disregards the boundary conditions.

6.3 IMPLICIT VERSUS EXPLICIT SCHEMES — STABILITY AND EFFICIENCY CONSIDERATIONS

We saw on the example of the two schemes for the heat equation that the implicit and explicit methods have very different stability characteristics. For the explicit schemes, the time-step limitation can be quite severe, requiring small Δt and, thus, large number of time steps. For example, a solution using the simple explicit method (6.2) with $\Delta x = 0.01\pi$ is stable only if $\Delta t < 1.97 \cdot 10^{-3}$, which means that about 2.5×10^4 time steps are needed to cover the interval $0 < t < 50$. On the contrary, many implicit methods, such as the simple implicit method (6.33), are unconditionally stable. The solution can be completed in much fewer time steps, say 10^3 or even 10^2.

At first glance, the implicit methods are much more efficient and have to be invariably used. A more careful consideration, though, shows that the situation is complex and case-specific.

Two important factors have to be considered. The first is that, although we can choose the time step of an implicit scheme as large as desired, caution must be exercised. If Δt is too large, the numerical solution, albeit stable, can suffer from large truncation errors. For example, the simple implicit method (6.33) has T.E. $= O(\Delta t, (\Delta x)^2)$ and selecting, say, $\Delta t = 1$ would not be a good idea, if one is looking for an accurate representation of the time evolution of u.

The situation is different when we are interested only in the final equilibrium state and not in the transient process leading to it. In this case, using an implicit method with large Δt is justified. As illustrated in Figure 6.7, the numerical solution obtained with a large time step does not accurately represent the evolution of the system. The final state, however, is a solution of a steady-state equation. The accuracy of numerical approximation depends on Δx and order of spatial approximation, but not on Δt.

The second factor is the computational cost. As opposite to the explicit schemes, where the time advancement is a relatively simple and computationally inexpensive task (see (6.2)), the implicit approach requires us to solve a system of coupled linear algebraic equations at every time layer. The total number of equations is equal to the number of unknowns multiplied by the number the space grid points, which can be quite large.

Table 6.1 summarizes the advantages and disadvantages of the implicit and explicit approaches. They were illustrated in this section on the example of just two schemes for the heat equation, but apply to the solution of all marching problems in general.

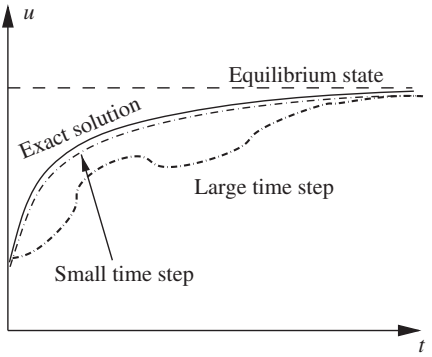


Figure 6.7 Effect of time step on solution by an implicit unconditionally stable scheme. Small Δt guarantees accurate representation of the entire solution, including the transient part. Accuracy is low for the transient part if Δt is large, but the final equilibrium state is reached faster.

Table 6.1 Comparison between Implicit and Explicit Approaches to Solution of Marching Problems

	Explicit Schemes	Implicit Schemes
Advantages	• Small amount of computations is needed for one time step. • Accurate solution is generated if stability criteria are satisfied and sufficiently small Δx, Δt are used. • Easy to program.	• No or low stability constraints. Large Δt can be used to achieve the equilibrium state in shorter time. • Accurate solution is generated if sufficiently small Δx, Δt are used.
Disadvantages	• Stability constraints. Often, very small Δt must be taken, which may lead to large total amount of computations.	• Larger amount of computations is needed for one time step, which may lead to large total amount of computations. • More difficult to program.

REFERENCES AND SUGGESTED READING

Fletcher, C. A. J. *Computational Techniques for Fluid Dynamics*. Vol. 1, *Fundamental and General Techniques*. Berlin: Springer-Verlag, 1991.

Tannehill, J. C., D. A. Anderson, and R. H. Pletcher. *Computational Fluid Mechanics and Heat Transfer*. Philadelphia: Taylor & Francis, 1997.

PROBLEMS

1. Can the numerical instability be avoided by using higher precision (larger number of decimal digits) in the computations, thus reducing the round-off error? If not, what would be the effect of higher precision?

2. Verify your answers to problem 1 in a simple computational experiment. Use the simple explicit scheme (6.2) to compute the solution of the example problem discussed in the beginning of the chapter. Take $r = 0.6$ and run computations with different levels of precision, for example, with simple and double precision in FORTRAN.

3. Consider the heat equation (6.1) with $a = 0.1$ solved in the interval $0 < x < 0.1$ using the simple explicit method (6.2). The number of grid points in the x-direction is $N = 101$ including the two points at the boundaries of the interval. What is the maximum time step Δt that allows us to avoid instability?

4. Consider the same question as in the previous problem, but for the simple implicit method.

5. Use the Neumann analysis to determine stability properties of the scheme

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_i^n - u_{i-1}^n}{\Delta x} = 0$$

applied to the linear convection equation $u_t + cu_x = 0$, where c is a positive constant.

6. Answer the same question as in the previous problem but for the scheme

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} = 0.$$

7

APPLICATION TO MODEL EQUATIONS

In this chapter we present a few selected schemes for one-dimensional unsteady model equations. Heat and linear convection equations are considered as the simplest examples of parabolic and hyperbolic systems. We also solve the Burgers equation to demonstrate the difficulties brought by nonlinearity and the possible approach to resolving them. Our goal is not to provide an extensive review of numerous schemes developed over the decades of existence of CFD. On the contrary, the number of the discussed schemes will be limited to the minimum needed to illustrate the distinctive features, pitfalls, and successive strategies typical for each type of PDE. The one-dimensional model equations are particularly helpful in this regard, since they provide the advantages of simplicity and existence of exact analytical solutions. As the last comment, since the equations are one-dimensional, there is no or little difference between the results of the finite difference and finite volume approaches. The schemes discussed in this chapter can be developed following either of these techniques of spatial discretization.

7.1 LINEAR CONVECTION EQUATION

The one-dimensional wave equation, a representative of the hyperbolic equations of second order, is

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}. \tag{7.1}$$

The solution can be described as a combination of two waves propagating in the opposite directions:

$$u(x,t) = F(x+ct) + G(x-ct), \tag{7.2}$$

where F and G are functions defined by initial and boundary conditions. Although (7.1) is simple already, the principal features of the finite difference schemes for hyperbolic equations can be investigated on the example of an even simpler linear convection equation:

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0, \quad c > 0. \tag{7.3}$$

Our interest in this equation is also justified by the direct similarity between its structure and the structure of the material derivative $Du/Dt \equiv \partial u/\partial t + (\boldsymbol{V}\cdot\nabla)u$ present in the governing equations of fluid mechanics and heat transfer.

The linear convection equation has the solution in the form of a single wave

$$u(x,t) = F(x-ct) \tag{7.4}$$

propagating in the positive x-direction with the constant speed c (see Figure 7.1). The solution can be easily found, if we know the initial condition

$$u(x,0) = F(x) \tag{7.5}$$

and the boundary condition at the low-x boundary of the computational domain. Note that, since the equation is of the first order, only one boundary condition is needed. It has to be on the left end in agreement with the wave propagation to the right.

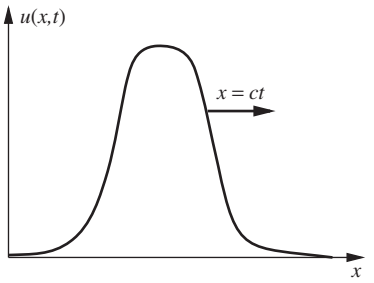


Figure 7.1 A solution of the linear convection equation (7.3).

7.1.1 Simple Explicit Schemes

We start with the simplest possible schemes. Equation (7.3) is approximated at the grid point (x_i, t^n) using the first-order forward difference for the time derivative and the first-order forward, backward, or second-order central formula for the space derivative. This results in three different schemes illustrated in Figure 7.2:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_{i+1}^n - u_i^n}{\Delta x} = 0, \quad \text{T.E.} = O(\Delta t, \Delta x), \tag{7.6}$$

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_i^n - u_{i-1}^n}{\Delta x} = 0, \quad \text{T.E.} = O(\Delta t, \Delta x), \tag{7.7}$$

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} = 0, \quad \text{T.E.} = O(\Delta t, (\Delta x)^2). \tag{7.8}$$

The same schemes can be developed with the finite volume approach. The upwind scheme (7.7) was, in fact, derived in section 5.3.1 using the upwind interpolation (5.17) of the convective fluxes $u_{i+1/2}$ and $u_{i-1/2}$ at the boundaries of the cell $\Omega_i = [x_{i-1/2}, x_{i+1/2}]$. Using the linear interpolation (5.21), we would obtain the scheme (7.8). In order to obtain the scheme (7.6), we would have to use the *downwind* interpolation $u_{i+1/2} \approx u_{i+1}$, $u_{i-1/2} \approx u_i$.

Let us analyze the stability of the derived schemes using the Neumann algorithm (see section 6.2.1). Since the PDE under consideration is linear, the same difference equations (7.6)–(7.8) hold for the round-off error ϵ. Dividing the equations by ϵ_i^n, we obtain, for the ratio $e^{a\Delta t} = \epsilon_i^{n+1}/\epsilon_i^n$:

$$\frac{e^{a\Delta t} - 1}{\Delta t} + c\frac{e^{\imath\beta} - 1}{\Delta x} = 0, \tag{7.9}$$

$$\frac{e^{a\Delta t} - 1}{\Delta t} + c\frac{1 - e^{-\imath\beta}}{\Delta x} = 0, \tag{7.10}$$

$$\frac{e^{a\Delta t} - 1}{\Delta t} + c\frac{e^{\imath\beta} - e^{-\imath\beta}}{2\Delta x} = 0, \tag{7.11}$$

where $2\pi/N \le \beta \le \pi$ stands for $k_m \Delta x$, and k_m is the wavenumber of the Fourier harmonics constituting the error.

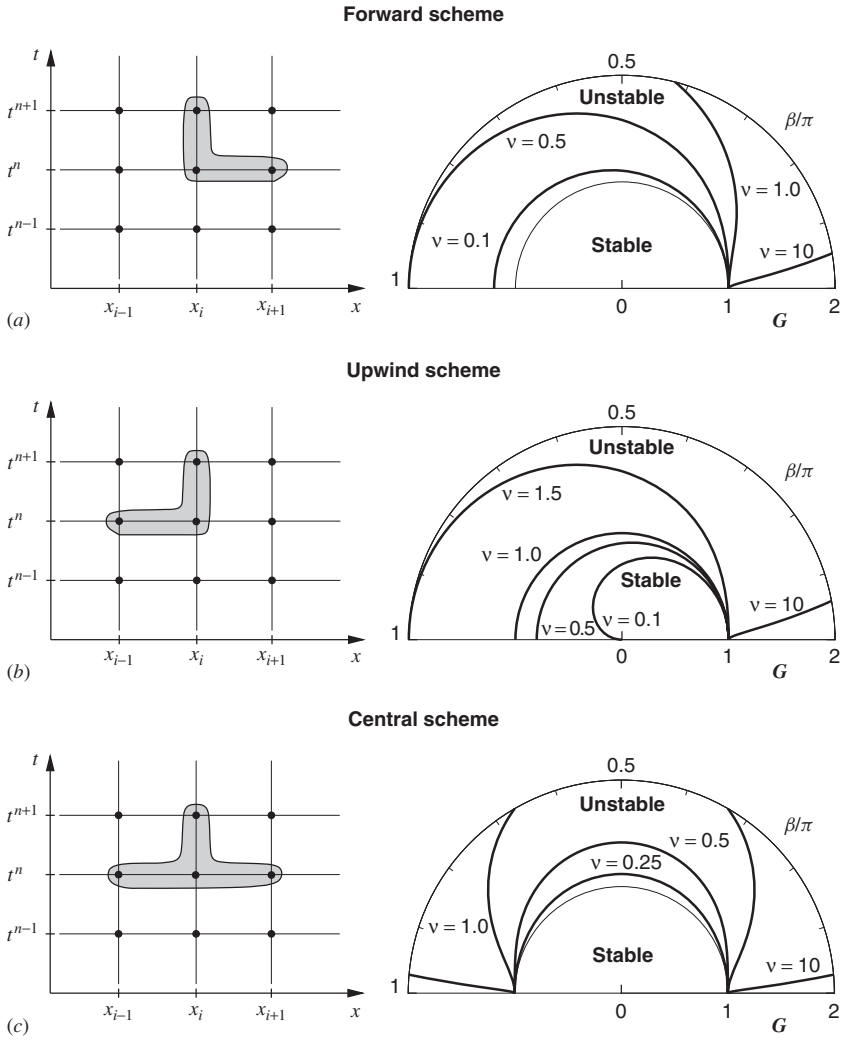


Figure 7.2 Finite difference molecules and amplification factors of the simple explicit schemes applied to the wave equation (7.3): *(a)* forward scheme (7.6); *(b)* upwind scheme (7.7); *(c)* central scheme (7.8).

The amplification factors $G = |e^{a\,\Delta t}|$ are, respectively,

$$G = \left|1 - \nu(e^{\imath\beta} - 1)\right| = \left|1 + \nu - \nu e^{\imath\beta}\right|, \tag{7.12}$$

$$G = \left|1 - \nu(1 - e^{-\imath\beta})\right| = \left|1 - \nu + \nu e^{-\imath\beta}\right|, \tag{7.13}$$

$$G = \left|1 - \nu(e^{\imath\beta} - e^{-\imath\beta})\right| = |1 - \nu\imath \sin\beta|, \tag{7.14}$$

where we use the *Courant coefficient*

$$\nu = c\frac{\Delta t}{\Delta x}. \tag{7.15}$$

The Courant coefficient plays a special role in the stability of schemes for hyperbolic equations, similarly to the coefficient r for parabolic equations (see (6.31)). The amplification factors (7.12)–(7.14) are plotted in Figure 7.2 as functions of β. We see that the forward and central schemes (7.6) and (7.8) are *unconditionally unstable* (unstable for any choice of time and space discretization steps), which makes them worthless. By contrast, the scheme (7.7) is stable if ν satisfies the Courant-Friedrichs-Lewy (often abbreviated to CFL) stability condition:

$$0 \le \nu = c\frac{\Delta t}{\Delta x} \le 1. \tag{7.16}$$

The scheme (7.7) is called the first-order *upstream* or *upwind* method.

The modified equation for the upwind scheme is

$$u_t + cu_x = \frac{c\Delta x}{2}(1-\nu)u_{xx} - \frac{c(\Delta x)^2}{6}(2\nu^2 - 3\nu + 1)u_{xxx} + O((\Delta x)^3, (\Delta x)^2\Delta t, \Delta x(\Delta t)^2, (\Delta t)^3). \tag{7.17}$$

The leading term of the truncation error is a numerical dissipation term. The fact that this term is of the first order means that the numerical dissipation is strong, unless a very fine grid is used. This is a serious problem, not only for the particular scheme (7.7), but, in general, for the schemes based on the upwind interpolation.

The last comment concerns a remarkable simplification of the upwind scheme, which is achieved if $\nu = 1$ is chosen. As one can see in (7.17), coefficients at dominating dissipation and dispersion errors become zero. In fact, the entire truncation error can be shown to vanish, and the scheme reduces to the so-called *shift method*

$$u_i^{n+1} = u_{i-1}^n, \tag{7.18}$$

which is an exact solution obtained by the method of characteristics.

7.1.2 Other Schemes

Simple Implicit Scheme: The simple implicit scheme for (7.3) is obtained by approximating the equation at (x_i, t^{n+1}) using backward

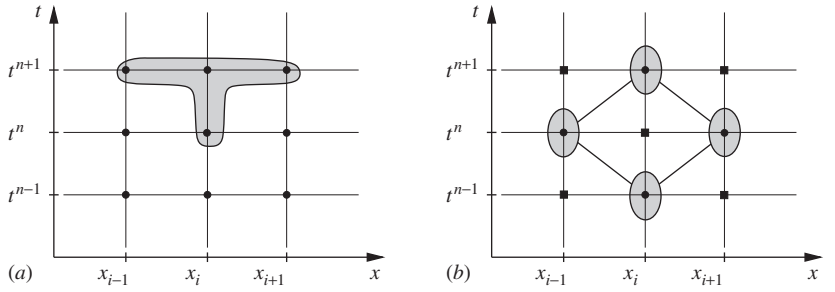


Figure 7.3 Finite difference molecules of schemes applied to the wave equation (7.3). *(a)* Simple implicit scheme (7.19); *(b)* Leapfrog scheme (7.21).

difference for the time derivative and central difference for the space derivative:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_{i+1}^{n+1} - u_{i-1}^{n+1}}{2\Delta x} = 0. \tag{7.19}$$

The finite difference molecule is shown in Figure 7.3a. This method is unconditionally stable and has the truncation error $O(\Delta t, (\Delta x)^2)$. The modified equation

$$u_t + cu_x = \left(\frac{c^2\Delta t}{2}\right)u_{xx} - \left(\frac{c(\Delta x)^2}{6} + \frac{1}{3}c^3(\Delta t)^2\right)u_{xxx} + \cdots \tag{7.20}$$

shows that, as typical for the first-order methods, the truncation error is dominated by numerical dissipation. The dissipation is particularly strong if one relies on the unconditional stability and uses large time steps Δt.

Leapfrog Scheme: The methods just discussed are of the first order in time. Their accuracy is low, and the solutions are distorted by strong numerical dissipation, unless very small grid steps are used. As the first example of the second-order schemes, we present the leapfrog method. This method is also the first scheme we consider that involves the values of u on more than two time layers (see Figure 7.3b). For the linear convection equation (7.3), the leapfrog scheme is

$$\frac{u_i^{n+1} - u_i^{n-1}}{2\Delta t} + c\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} = 0. \tag{7.21}$$

The formula can be designed by approximating the space and time derivatives at (x_i, t^n) by central differences. The method is explicit with the

truncation error $O((\Delta t)^2, (\Delta x)^2)$. It is stable at $\nu \leq 1$. The modified equation is

$$u_t + cu_x = \frac{c(\Delta x)^2}{6}(\nu^2 - 1)u_{xxx} - \frac{c(\Delta x)^4}{120}(9\nu^4 - 10\nu^2 + 1)u_{xxxxx} + \cdots. \tag{7.22}$$

It can be seen that numerical dispersion dominates the truncation error. This is common for the methods of the second order in time. A remarkable feature of the leapfrog scheme is that the right-hand side of the modified equation does not contain the even-order derivatives of u at all. The method generates no numerical dissipation! This is both good and bad news. Some amount of numerical dissipation is needed to suppress the round-off errors. The leapfrog method is a nice illustration of this statement. In the absence of numerical dissipation, the method is *neutrally stable*. The amplification factor is

$$G = \left| \pm(1 - \nu^2 \sin^2 \beta)^{1/2} - i\nu \sin \beta \right|. \tag{7.23}$$

It is larger than 1 if $\nu > 1$. An interesting behavior is observed when $\nu \leq 1$. The amplification factor is $G \equiv 1$ for all such ν. The round-off errors introduced at every time step neither grow nor decay as the solution advances.

Being, historically, one of the first second-order methods, the leapfrog scheme was quite popular during the early years of CFD (1960s and 1970s). It has lost popularity to other methods, partially because of one serious drawback associated with the leapfrog nature of the solution. It can be seen in the equation (7.21) and Figure 7.3b that the scheme connects u_i^{n+1} with u_{i-1}^n, u_{i+1}^n, and u_i^{n-1}, but not with u_i^n, u_{i-1}^{n-1}, or u_{i+1}^{n-1}. These other values are connected with each other and with u_i^{n-2} by the finite difference equation

$$\frac{u_i^n - u_i^{n-2}}{2\Delta t} + c\frac{u_{i+1}^{n-1} - u_{i-1}^{n-1}}{2\Delta x} = 0,$$

which is the approximation of the PDE at the grid point (x_i, t^{n-1}). The system of discretization equations consists of two uncoupled subsystems, one with equations at "square" grid points in Figure 7.3b and another with equations at "circular" points. If no extra precaution is taken, the solutions to the two subsystems can deviate from each other (split) with time.

Lax-Wendroff Scheme: This explicit method is given by the formula

$$u_i^{n+1} = u_i^n - \frac{\nu}{2}(u_{i+1}^n - u_{i-1}^n) + \frac{\nu^2}{2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n), \tag{7.24}$$

where $\nu = c\Delta t/\Delta x$ is the Courant coefficient. At first glance, the scheme does not look at all as an approximation of the equation (7.3). This is, however, a consistent approximation with T.E. $= O((\Delta x)^2, (\Delta t)^2)$. This can be verified by the Taylor expansion method. Except for the extra term in the right-hand side, the scheme is identical to the unstable simple central difference scheme (7.8). This extra term is designed so as to cancel the dominating numerical dissipation term in the modified equation for the scheme (7.8) and, as it happens, to stabilize the scheme. The modified equation for the Lax-Wendroff scheme is

$$u_t + cu_x = -\frac{c(\Delta x)^2}{6}(1-\nu^2)u_{xxx} - \frac{c(\Delta x)^3}{8}\nu(1-\nu^2)u_{xxxx} + \cdots. \tag{7.25}$$

The truncation error is predominantly dispersive. The scheme is stable if $|\nu| \leq 1$. It reduces to the shift scheme (7.18) when $\nu = 1$.

MacCormack Scheme: This method, developed by MacCormack in 1969 is important for us as an example of the two-step predictor-corrector methods. For the wave equation (7.3), the scheme is

$$\begin{aligned} \text{Predictor:} \quad & u_i^* = u_i^n - \nu(u_{i+1}^n - u_i^n) \\ \text{Corrector:} \quad & u_i^{n+1} = \frac{1}{2}\left[u_i^n + u_i^* - \nu\left(u_i^* - u_{i-1}^*\right)\right]. \end{aligned} \tag{7.26}$$

Although the particular form of the method can be different for different equations, the principal approach is always the same: We calculate an intermediate *predicted* solution and, then, "correct" it to achieve desired accuracy and stability. For the MacCormack method, the truncation error is $O((\Delta x)^2, (\Delta t)^2)$ and the stability criterion is $\nu \leq 1$.

7.2 ONE-DIMENSIONAL HEAT EQUATION

As before, we employ the one-dimensional heat equation

$$\frac{\partial u}{\partial t} = a^2\frac{\partial^2 u}{\partial x^2} \tag{7.27}$$

as the simplest representative of the parabolic family. An inhomogeneous part $f(x,t)$ can be added to the right-hand side without the loss of generality. We have already solved this equation in sections 4.3.6 and 6.2.1

using the simple explicit and implicit methods. In this section, we extend the analysis by adding another scheme and investigating the modified equations.

7.2.1 Simple Explicit Scheme

The simple explicit method

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = a^2 \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} \tag{7.28}$$

has the truncation error $O(\Delta t, (\Delta x)^2)$ and is stable when the stability parameter satisfies

$$r = \frac{a^2 \Delta t}{(\Delta x)^2} \leq \frac{1}{2}. \tag{7.29}$$

Additional information can be extracted from the modified equation

$$u_t - a^2 u_{xx} = \left[-\frac{a^4 \Delta t}{2} + \frac{(a\Delta x)^2}{12} \right] u_{xxxx} + \left[\frac{a^6 (\Delta t)^2}{3} - \frac{a^4 \Delta t (\Delta x)^2}{12} + \frac{a^2 (\Delta x)^4}{360} \right] u_{xxxxxx} + \cdots. \tag{7.30}$$

There are no odd-derivative terms in (7.30). This means that the truncation error has no dispersive part. This feature is common for the finite difference schemes of second order in space applied to parabolic equations.

Another interesting feature of the modified equation (7.30) is that we can greatly reduce the dissipation error and improve the accuracy of the scheme by choosing Δt and Δx such that $r = 1/6$. The first term of the truncation error vanishes, and the error becomes $O((\Delta t)^2, (\Delta x)^4)$. To illustrate the effect, we solve the heat equation for

$$0 < x < \pi,\ a = 2,\ u(0,t) = u(\pi,t) = 0,\ u(x,0) = \sin(5x). \tag{7.31}$$

The solution can be compared with the exact solution $u_{\text{exact}} = \sin(5x)\exp(-(5a)^2 t)$. The calculations are performed at deliberately low space resolution with just $N = 15$ grid points. Two time steps are used, one corresponding to $r = 0.4$ and another to $r = 1/6$. It is clearly visible in Figure 7.4 that the solution with $r = 1/6$ has much higher accuracy.

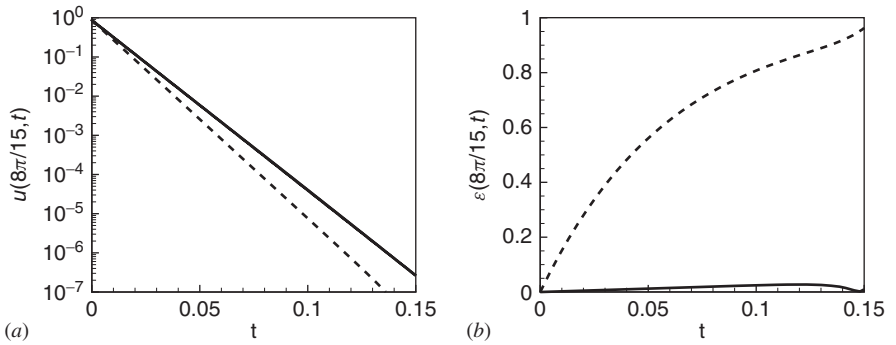


Figure 7.4 Solution of the heat equation problem (7.27), (7.31) by simple explicit method. Space resolution is $\Delta x = \pi/15$. The time step is such that $r = 1/6$ (solid lines) or $r = 0.4$ (dashed lines). *(a)* Solution at $x = x_8 = 8\pi/15$ as a function of time. The curve for the exact solution practically coincides with the curve for $r = 1/6$. *(b)* Relative error $\epsilon = |(u_{\text{calculated}} - u_{\text{exact}})/u_{\text{exact}}|$ at $x = 8\pi/15$.

7.2.2 Simple Implicit Scheme

The simple implicit method is

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = a^2 \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2}. \tag{7.32}$$

The method is unconditionally stable and has the truncation error $O(\Delta t, (\Delta x)^2)$. Analysis of the modified equation

$$u_t - a^2 u_{xx} = \left[\frac{a^4 \Delta t}{2} + \frac{(a\Delta x)^2}{12}\right] u_{xxxx} + \left[\frac{a^6 (\Delta t)^2}{3} + \frac{a^4 \Delta t (\Delta x)^2}{12} + \frac{a^2 (\Delta x)^4}{360}\right] u_{xxxxxx} + \cdots \tag{7.33}$$

shows that the truncation error is purely dissipative.

Let us compare (7.33) with the modified equation (7.30) for the simple explicit method. The amplitude of the dominating part of the truncation error is determined by the coefficient at u_{xxxx}. In the explicit scheme, the terms in this coefficient are of different signs so they cancel each other, either completely at $r = 1/6$ or partially at other values of r. No such cancelation occurs in the implicit scheme. This means that, at the same Δx and Δt, the explicit method is more accurate than the implicit one.

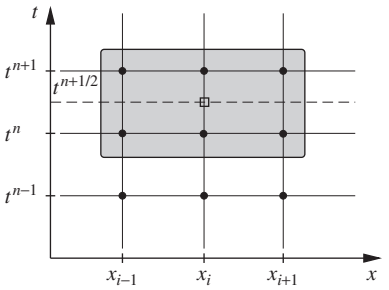


Figure 7.5 Difference molecule for the Crank-Nicolson scheme (7.34).

7.2.3 Crank-Nicolson Scheme

Implicit and explicit approaches can be mixed together in one scheme, sometimes providing remarkably good results. An important example is the Crank-Nicolson scheme:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = a^2 \frac{1}{2} \frac{(u_{i+1}^n - 2u_i^n + u_{i-1}^n) + (u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1})}{(\Delta x)^2}. \tag{7.34}$$

The scheme is illustrated by the difference molecule in Figure 7.5. We can obtain this scheme by approximating the PDE at the point $(x_i, t^{n+1/2})$ located at the imaginary half-integer time layer $t^{n+1/2} = (t^n + t^{n+1})/2$. The central difference formula (4.10) with the grid step $\Delta t/2$ is applied to the time derivative $\partial u/\partial t$ at $t^{n+1/2}$:

$$\left.\frac{\partial u}{\partial t}\right|_i^{n+1/2} = \frac{u_i^{n+1} - u_i^n}{\Delta t} + O((\Delta t)^2). \tag{7.35}$$

For the space derivative, we use the second-order central difference (4.19) applied to the result of the interpolation $u_i^{n+1/2} = \left(u_i^n + u_i^{n+1}\right)/2$:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_i^{n+1/2} = \frac{1}{2}\left(\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} + \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2}\right).$$

As discussed in section 4.2.7, the interpolation procedure generates the error of the order of $O\left((\Delta t)^2\right)$. The cumulative truncation error of (7.34) is T.E. $= O\left((\Delta t)^2, (\Delta x)^2\right)$. The time discretization by the Crank-Nicòlson scheme (7.34) is more accurate than the discretizations by the fully explicit and implicit schemes (7.28) and (7.32).

Applying the Neumann stability analysis, we find that the amplification factor of (7.34) is

$$G = \left| \frac{1 - r(1 - \cos\beta)}{1 + r(1 - \cos\beta)} \right|. \tag{7.36}$$

With positive r, this means that the scheme is unconditionally stable.

7.3 BURGERS AND GENERIC TRANSPORT EQUATIONS

We consider the Burgers equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \mu\frac{\partial^2 u}{\partial x^2} \tag{7.37}$$

and the generic transport equation

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} = \mu\frac{\partial^2 \phi}{\partial x^2} \tag{7.38}$$

introduced in section 3.1.1. Each of them combines three important features of the Navier-Stokes equations: unsteadiness, nonlinear convection, and dissipation (diffusion). The equations are hyperbolic at $\mu = 0$. They are formally parabolic at $\mu > 0$, although they may be viewed as of mixed hyperbolic-parabolic type. In the latter view, the hyperbolic character is given by the combination of the first and second terms, while the parabolic character is due to the combination of the first and third terms.

We will consider one scheme for the Burgers equation (7.37). It can be easily modified for the generic transport equation. The scheme uses central differences of the second order for the space derivatives and forward difference of the first order for the time derivative:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + u_i^n \frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} = \mu\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2}. \tag{7.39}$$

The scheme is explicit with the truncation error T.E. $= O(\Delta t, (\Delta x)^2)$.

The stability analysis of (7.39) is nontrivial, primarily because the equation is nonlinear. The standard Neumann stability analysis cannot, strictly speaking, be applied because it assumes existence of exponential solutions of the equations for the round-off error, which is true only for analysis of linear PDE with constant coefficients. The common approach

in the presence of nonlinearity is to "freeze" the coefficient, such as u in (7.37) and (7.38); that is, to act as if it were a constant. For the scheme (7.39), this means conducting the stability analysis for the linearized discretization equation with u_i^n replaced by a constant c. Such analysis leads to two stability constraints:

$$c^2\Delta t \le 2\mu, \text{ which can be rewritten as } \nu^2 \le 2r, \text{ and } r \le 1/2, \quad (7.40)$$

where, as before, $\nu = c\Delta t/\Delta x$ and $r = \mu\Delta t/(\Delta x)^2$.

This must be true *for any* u. This means that we must evaluate the maximum amplitude of the solution

$$u_{\max} = \max_{\Omega} |u|$$

and rewrite the stability criteria as

$$(u_{\max})^2\Delta t \le 2\mu, \text{ and } r \le 1/2. \quad (7.41)$$

The same criteria should be satisfied by the scheme (7.39) applied to the generic transport equation.

Another aspect of the stability analysis of (7.39) and other more complex equations, such as the Navier-Stokes system, concerns the possibility of using the stability criteria derived for the model linear convection and heat equations (7.3) and (7.27). One may be tempted to consider (7.39) as a combination of the central difference scheme (7.8) and the simple explicit scheme (7.28) applied, respectively, to the convective and diffusive parts of the Burgers equation. It seems logical that the stability conditions for (7.39) should be a result of combining the criteria developed for the corresponding model schemes (7.8) and (7.28). Quite often, the logic works. We can assume that a scheme is stable if the criteria derived for the schemes used for the convective and diffusive parts are satisfied and, then, verify the assumption in test calculations.

Sometimes, however, the approach results in excessively restrictive conditions. For example, applying it to (7.39) would lead to the conclusion that the scheme is unconditionally unstable since the scheme (7.8) used to approximate the convective part is unconditionally unstable (see section 7.1.1). A more accurate stability analysis of (7.39), however, shows that the scheme is stabilized by the presence of the diffusion term, provided the diffusion coefficient is large enough, as specified by the first condition in (7.40).

7.4 METHOD OF LINES APPROACH

The schemes considered so far were based on simultaneous discretization of time and space derivatives. This approach has the advantage that the scheme is formulated in its final form from the very beginning. Its order of approximation and stability properties can be fully analyzed. Furthermore, some schemes can be optimized in such a way that certain terms of truncation error cancel each other, so that the accuracy is improved (the optimization of the simple explicit scheme for the heat equation at $r = 1/6$ is a good example).

There is an alternative approach called *method of lines*, according to which the operations of spatial and time discretization are separated. The method of lines can be applied to any PDE or a system of PDEs, which can be represented in the form that includes one time derivative of the first order in each equation. The spatial discretization is performed first, and the resulting system of equations is written down as a system of ordinary differential equations

$$\boldsymbol{u}_t = \boldsymbol{R}[\boldsymbol{u}], \tag{7.42}$$

where $\boldsymbol{u}(t)$ is the vector of spatially discretized variables (grid points values in the case of finite difference schemes or amplitudes of trial functions in the case of spectral methods) and $\boldsymbol{R}[\boldsymbol{u}]$ is a differential operator that includes approximations of spatial derivatives of u and other terms.

The system (7.42) is solved using one of the time-integration methods developed for the ordinary differential equations. We have already used this approach when we illustrated the spectral methods in section 3.3.1.

The numerical stability to round-off errors is determined by the type of the time-integration scheme and by the properties of the operator $\boldsymbol{R}[\boldsymbol{u}]$. Obtaining the detailed information including the amplification factor and exact stability criteria typically requires analysis of a complete discretized system, although some tendencies can often be predicted on the basis of the time-integration scheme alone.

Many methods of time integration of the general equation (7.42) are available. We will consider two families of multilevel schemes: Adams and Runge-Kutta methods. They can be effectively applied, in the way just outlined, to PDE of both hyperbolic and parabolic types.

7.4.1 Adams Methods

The Adams methods are derived by polynomial fitting over several consecutive time layers. They can be explicit (Adams-Bashforth) or implicit

(Adams-Moulton) methods. The general formulas are

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \sum_{j=0}^{q} A_j \boldsymbol{R}^{n-j} \quad \text{Adams-Bashforth,} \tag{7.43}$$

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \sum_{j=-1}^{q-1} A_j \boldsymbol{R}^{n-j} \quad \text{Adams-Moulton,} \tag{7.44}$$

where $\boldsymbol{u}^n$ is the spatially discretized vector of solution at the time layer t^n and $\boldsymbol{R}^{n-j} = \boldsymbol{R}\left[\boldsymbol{u}^{n-j}\right]$. The scheme coefficients A_j are determined by assuming polynomial behavior of $\boldsymbol{u}(t)$. The number q of terms in the right-hand side determines the order of accuracy. For example, at $q = 0$, we obtain the well-known explicit and implicit Euler methods with T.E. $\sim O(\Delta t)$:

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \boldsymbol{R}^n, \tag{7.45}$$

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \boldsymbol{R}^{n+1}. \tag{7.46}$$

Formulas with $q = 1$ and $q = 2$ give schemes of the second and third order, respectively:

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \frac{\Delta t}{2}\left(3\boldsymbol{R}^n - \boldsymbol{R}^{n-1}\right) + O\left((\Delta t)^2\right), \tag{7.47}$$

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \frac{\Delta t}{2}\left(\boldsymbol{R}^{n+1} + \boldsymbol{R}^n\right) + O\left((\Delta t)^2\right), \tag{7.48}$$

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \frac{\Delta t}{12}\left(23\boldsymbol{R}^n - 16\boldsymbol{R}^{n-1} + 5\boldsymbol{R}^{n-2}\right) + O\left((\Delta t)^3\right), \tag{7.49}$$

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \frac{\Delta t}{12}\left(5\boldsymbol{R}^{n+1} + 8\boldsymbol{R}^n - \boldsymbol{R}^{n-1}\right) + O\left((\Delta t)^3\right). \tag{7.50}$$

7.4.2 Runge-Kutta Methods

Generally, there exists an infinite number of the multistep Runge-Kutta schemes, which can be formulated with an arbitrary order of accuracy. For example, one method of the second order is

$$\text{Step 1:} \quad \boldsymbol{u}^{(1)} = \boldsymbol{u}^n + \frac{\Delta t}{2}\boldsymbol{R}^n$$

$$\text{Step 2:} \quad \boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \boldsymbol{R}^{(1)}, \tag{7.51}$$

where $\boldsymbol{R}^n = \boldsymbol{R}[\boldsymbol{u}^n]$ and $\boldsymbol{R}^{(1)} = \boldsymbol{R}\left[\boldsymbol{u}^{(1)}\right]$ are the spatial operators calculated for the solution at previous time layer $\boldsymbol{u}^n$ and the intermediate solution $\boldsymbol{u}^{(1)}$. A popular method of the fourth order is

$$\begin{aligned}
&\text{Step 1:} \quad \boldsymbol{u}^{(1)} = \boldsymbol{u}^n + \frac{\Delta t}{2}\boldsymbol{R}^n \\
&\text{Step 2:} \quad \boldsymbol{u}^{(2)} = \boldsymbol{u}^n + \frac{\Delta t}{2}\boldsymbol{R}^{(1)} \\
&\text{Step 3:} \quad \boldsymbol{u}^{(3)} = \boldsymbol{u}^n + \Delta t\boldsymbol{R}^{(2)} \\
&\text{Step 4:} \quad \boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \frac{\Delta t}{6}\left(\boldsymbol{R}^n + 2\boldsymbol{R}^{(1)} + 2\boldsymbol{R}^{(2)} + \boldsymbol{R}^{(3)}\right).
\end{aligned} \tag{7.52}$$

As an example, we can solve the linear convection equation (7.3) using the Runge-Kutta scheme of the second order (7.51) and applying central differences for spatial discretization:

$$\begin{aligned}
\mathrm{R}^n &= -c\frac{u^n_{i+1} - u^n_{i-1}}{2\Delta x} + O((\Delta x)^2), \\
\mathrm{R}^{(1)} &= -c\frac{u^{(1)}_{i+1} - u^{(1)}_{i-1}}{2\Delta x} + O((\Delta x)^2).
\end{aligned}$$

The resulting scheme has T.E. $= O((\Delta x)^2, (\Delta t)^2)$.

The main advantages of the Runge-Kutta methods are their flexibility (a scheme of any order in time can be developed) and good stability properties. They are not unconditionally stable, but the largest allowed time step is typically large in comparison with steps required by other explicit methods. The most serious disadvantage is the necessity to calculate the operator $\boldsymbol{R}[\boldsymbol{u}]$ several times on every time step.

7.5 IMPLICIT SCHEMES: SOLUTION OF TRIDIAGONAL SYSTEMS BY THOMAS ALGORITHM

Any implicit scheme requires solution of a system of linear algebraic equations at every time step. Typically, this is a computationally challenging task, since the system is very large. Its size in a three-dimensional CFD analysis can be $\sim 10^6$ or even larger. There exists a variety of methods for solving such systems. Some of them are discussed in Chapter 8. Here, we consider the special case when the solution of a one-dimensional PDE is calculated. We assume that the time discretization involves only two subsequent time layers and the space discretization is done using a

finite difference scheme based on not more than three neighboring points. The resulting system of linear equations has a *tridiagonal* matrix and can be solved using the very efficient form of the Gauss elimination called *double sweep* or *Thomas* algorithm. The technique is credited to L. H. Thomas, who published it in 1949, although the idea is simple and, probably, occurred independently to other specialists dealing with the same problem.

Let us illustrate the method on the example of the simple implicit scheme

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = a^2 \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2} + f_i^{n+1}$$

written for the modified heat equation

$$\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2} + f(x,t).$$

We assume that the solution is sought in the interval $0 \le x \le L$ on a uniform grid $x_i = i\Delta x$, $i = 0, \ldots, N$, with $x_0 = 0$ and $x_N = L$. To demonstrate some variations of the method, we impose the Dirichlet boundary condition $u(0,t) = g_0$ at $x = 0$ and the Neumann boundary condition $(\partial u/\partial x)(L,t) = g_1$ at $x = L$.

The finite difference formula for the interior points can be rewritten as

$$a_i u_{i+1}^{n+1} + d_i u_i^{n+1} + b_i u_{i-1}^{n+1} = c_i, \quad i = 1, 2, \ldots, N-1, \tag{7.53}$$

where we used the abbreviations for the constant coefficients:

$$a_i = -a^2 \Delta t/\Delta x^2,\ b_i = -a^2 \Delta t/\Delta x^2,\ d_i = 1 + 2a^2 \Delta t/\Delta x^2,$$
$$c_i = u_i^n + \Delta t f_i^{n+1}.$$

The boundary condition at $x = 0$ can be expressed as

$$a_0 u_1^{n+1} + d_0 u_0^{n+1} = c_0, \text{ with } a_0 = 0,\ d_0 = 1, \text{ and } c_0 = g_0. \tag{7.54}$$

The boundary condition at $x = L$ must be approximated by a one-sided finite difference formula. It is important to maintain the second order of accuracy, so as not to compromise the accuracy of the entire solution. We employ the formula (4.14), which results in

$$\frac{3}{2} u_N^{n+1} - 2u_{N-1}^{n+1} + \frac{1}{2} u_{N-2}^{n+1} = \Delta x g_1.$$

For future use, we have to combine this equation with the equation (7.53) written for the point x_{N-1}

$$a_{N-1}u_N^{n+1} + d_{N-1}u_{N-1}^{n+1} + b_{N-1}u_{N-2}^{n+1} = c_{N-1}$$

and use the system of two equations to exclude the unknown u_{N-2}^{n+1} and obtain a linear relation between u_{N-1}^{n+1} and u_N^{n+1}

$$b_N u_{N-1}^{n+1} + d_N u_N^{n+1} = c_N, \tag{7.55}$$

where $b_N = d_{N-1} + 4b_{N-1}$, $d_N = a_{N-1} - 3b_{N-1}$, and $c_N = c_{N-1} - 2\Delta x b_{N-1} g_1$.

The entire system of equations can be written as

$$\begin{pmatrix} d_0 & a_0 & 0 & \dots & \dots & \dots & 0 \\ b_1 & d_1 & a_1 & 0 & \dots & \dots & 0 \\ 0 & b_2 & d_2 & a_2 & 0 & \dots & 0 \\ 0 & 0 & * & * & * & 0 & 0 \\ 0 & 0 & 0 & * & * & * & 0 \\ 0 & \dots & \dots & 0 & b_{N-1} & d_{N-1} & a_{N-1} \\ 0 & \dots & \dots & \dots & 0 & b_N & d_N \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ \vdots \\ c_{N-1} \\ c_N \end{pmatrix}, \tag{7.56}$$

where u_i is substituted for u_i^{n+1} for brevity.

We see that the coefficient matrix in the left-hand side is, indeed, tridiagonal. Its elements are zeros except for the main diagonal and adjacent sub- and super-diagonals.

The Thomas algorithm includes two stages, or *sweeps*. During the first stage, the *forward sweep*, the elementary row operations are applied to transform the coefficient matrix into upper triangular form. As known from linear algebra, an elementary row operation applied simultaneously to the coefficient matrix and the right-hand side does not affect the solution of a matric equation. First, we multiply the row 0 by $-b_1/d_0$ and add to the row 1 in order to remove b_1:

$$\begin{aligned} d_1' &= d_1 - \frac{b_1}{d_0} a_0 \\ b_1' &= b_1 - \frac{b_1}{d_0} d_0 = 0 \\ c_1' &= c_1 - \frac{b_1}{d_0} c_0. \end{aligned} \tag{7.57}$$

Similar operation with the transformed row 1 is then used to remove b_2 in row 2. The procedure continues with rows 3, 4, and so on. The general formula is

$$\begin{aligned} d_i' &= d_i - \frac{b_i}{d_{i-1}'} a_{i-1} \\ b_i' &= b_i - \frac{b_i}{d_{i-1}'} d_{i-1}' = 0 \\ c_i' &= c_i - \frac{b_i}{d_{i-1}'} c_{i-1}', \quad i = 1, 2, \ldots, N. \end{aligned} \tag{7.58}$$

Atthe end of the forward sweep, the matrix equation becomes

$$\begin{pmatrix} d_0 & a_0 & 0 & \ldots & \ldots & \ldots & 0 \\ 0 & d_1' & a_1 & 0 & \ldots & \ldots & 0 \\ 0 & 0 & d_2' & a_2 & 0 & \ldots & 0 \\ 0 & 0 & 0 & * & * & 0 & 0 \\ 0 & 0 & 0 & 0 & * & * & 0 \\ 0 & \ldots & \ldots & 0 & 0 & d_{N-1}' & a_{N-1} \\ 0 & \ldots & \ldots & \ldots & 0 & 0 & d_N' \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1' \\ c_2' \\ \vdots \\ \vdots \\ c_{N-1}' \\ c_N' \end{pmatrix}. \tag{7.59}$$

It now has an upper triangular coefficient matrix in the left-hand side.

In the *backward sweep*, the transformed system is used to find u_i. First, we determine

$$u_N = c_N'/d_N'. \tag{7.60}$$

Then, the value of u_N and the $(N-1)$st equation are used to find u_{N-1}:

$$u_{N-1} = (c_{N-1}' - a_{N-1}u_N)/d_{N-1}'.$$

The procedure continues backward to find the entire solution according to

$$\begin{aligned} u_i &= (c_i' - a_i u_{i+1})/d_i', \quad i = N-1, N-2, \ldots, 1, \\ u_0 &= (c_0 - a_0 u_1)/d_1'. \end{aligned} \tag{7.61}$$

The Thomas algorithm effectively utilizes the tridiagonal structure of the matrix and completes the solution of the linear system (7.56) in only $O(N)$ arithmetic operations. This is in sharp contrast with the standard Gauss elimination procedure that would require $O(N^3)$ operations. Obviously, the Thomas algorithm should be used wherever possible.

REFERENCES AND SUGGESTED READING

Fletcher, C. A. J. *Computational Techniques for Fluid Dynamics*. Vol. 1, *Fundamental and General Techniques*. Berlin: Springer-Verlag, 1991.

Press, W. P., S. A. Teukolsky, W. T. Vettering, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*. Cambridge, UK: Cambridge University Press, 2001.

Tannehill, J. C., D. A. Anderson, and R. H. Pletcher. *Computational Fluid Mechanics and Heat Transfer*: Philadelphia Taylor & Francis, 1997.

PROBLEMS

1. Compare the schemes introduced in section 7.1 for the linear convection equation:

a) Simple explicit schemes (7.6–7.8)
b) Simple implicit scheme (7.19)
c) Leapfrog scheme (7.21)
d) Lax-Wendroff scheme (7.24)
e) MacCormack scheme (7.26).

For every scheme, describe its stability properties (write the stability criterion, where appropriate) and order of approximation. Determine, where data are available, whether the numerical dissipation or numerical dispersion dominates the truncation error. Summarize the discussion stating, for every scheme, how appropriate is it in your opinion for solution of the linear convection equation. Does the scheme accurately reproduce the traveling wave solution shown in Figure 7.1? What are the main advantages and disadvantages of the scheme?

2. Repeat Problem 1 for the following schemes applied to the one-dimensional heat equation:

a) Simple explicit scheme (7.28)
b) Simple implicit scheme (7.32)
c) Crank-Nicolson scheme (7.34).

3. Modify the scheme (7.39) for the Burgers equation by changing the central difference approximation of the convective term to the upwind approximation. Assume that u is always positive. How does this modification change the order of approximation and the character of the truncation error?

4. One-dimensional heat equation is solved using the method of lines scheme based on the central difference for space derivative and the Adams-Bashforth scheme of second order (7.47) for time discretization. Write the full set of discretization equations solved on one time step.

5. Repeat Problem 4 for the scheme based on the Runge-Kutta method of second order (7.51).

6. Consider the application of the Thomas algorithm to the simple implicit scheme used to solve the one-dimensional heat equation (section 7.5). Modify the algorithm for other sets of boundary conditions:

a) Dirichlet conditions on both ends: $u(0,t) = g_0$, $u(L,t) = g_1$

b) Neumann conditions on both ends: $(\partial u/\partial x)(0,t) = g_0$, $(\partial u/\partial x)(L,t) = g_1$.

Programming Exercises Develop the following algorithms. Test in comparison with exact solutions, where available:

1. Thomas algorithm for solution of a linear system with tridiagonal matrix.
2. Upwind and leapfrog solutions of the linear convection equation $\partial u/\partial t + c\partial u/\partial x = 0$ at $0 \le x \le 10$ and $0 \le t \le 2$, with $c = 1$, the boundary condition $u(0,t) = 0$, and the initial condition $u(x,0) = \sin(x)$ if $x \le \pi$ and $u(x,0) = 0$ if $x > \pi$. Use the exact solution of the form (7.4) for verification.
3. Simple explicit, simple implicit, and Crank-Nicolson solutions of the one-dimensional heat equation $\partial u/\partial t = 4\left(\partial^2 u/\partial x^2\right)$ at $0 \le x \le 1$ and $0 \le t \le 0.1$ with boundary conditions $u(0,t) = u(1,t) = 0$ and initial condition $u(x,0) = \sin(5\pi x)$. Use the exact solution $u(x,t) = \sin(5\pi x)\exp((-10\pi)^2 t)$ for verification.

Part II

METHODS

8

STEADY-STATE PROBLEMS

In this chapter, we consider methods that can be used to solve a system of linear algebraic equations (a matrix equation)

$$\mathbf{A} \cdot \boldsymbol{v} = \boldsymbol{c} \tag{8.1}$$

or a system of nonlinear algebraic equations

$$\boldsymbol{F}(\boldsymbol{v}) = 0. \tag{8.2}$$

8.1 PROBLEMS REDUCIBLE TO MATRIX EQUATIONS

The matrix equation (8.1) is very common, almost unavoidable in CFD analysis. In the case of nonlinear PDE, solution of a matrix equation appears as a part of the iteration procedure that involves linearization. This is discussed in section 8.4.2. If PDE are linear, (8.1) follows directly from discretization. Several examples are considered in this section.

8.1.1 Elliptic PDE

As we discussed earlier, a typical elliptic equation describes steady-state physical processes or processes with zero adjustment time, such as the final temperature distribution in a body or pressure distribution in a flow of an incompressible fluid. The simplest model equations representing the class of elliptic PDE are the Laplace equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{8.3}$$

and the Poisson equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y). \tag{8.4}$$

From the computational viewpoint, we deal with a steady-state equation that has to be solved in a spatial domain Ω subject to certain boundary conditions at the boundary S (see Figure 8.1). The boundary conditions are typically of Dirichlet, Neumann, or Robin types. Due to the elliptic nature of the problem, the system of discretization equations always connects together the values of u at all grid points within Ω and at the boundary.

Five-Point Formula: For simplicity, we use a uniform rectangular grid $x_i = i\,\Delta x$, $y_j = j\,\Delta y$ with constant grid steps Δx and Δy. The common approach is to approximate the partial derivatives of (8.3) and (8.4) at internal points (x_i, y_j) using central differences of second order. The finite difference molecule consists of five grid points (x_{i-1}, y_j), (x_{i+1}, y_j), (x_i, y_j), (x_i, y_{j-1}), and (x_i, y_{j+1}) (see Figure 8.1). For example, the discretization of (8.4) is

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2} = f_{i,j}. \tag{8.5}$$

To complete the system of equations, we have to add finite difference approximations of boundary conditions at all grid points lying at the boundary S.

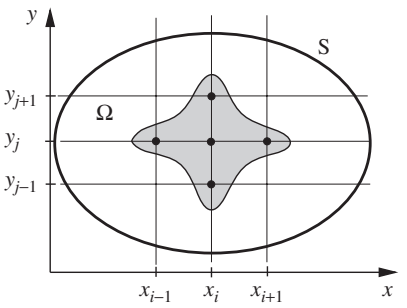


Figure 8.1 Features of the finite difference solution of Laplace and Poisson equations: computational domain Ω, its boundary S, rectangular grid, and difference molecule for the five-point formula.

Let us rewrite the system of finite difference equations in the compact matrix form (8.1). $\boldsymbol{v}$ is now the solution vector consisting of unknowns $u_{i,j}$ ordered in a single file, $\boldsymbol{c}$ is the similarly ordered vector of the right-hand sides, and $\mathbf{A}$ is the coefficient matrix. Vectors can be ordered in different ways. For example, in a rectangular domain with $i = 1, \ldots, N_x$, $j = 1, \ldots, N_y$, we can choose

$$v_1 = u_{1,1}, v_2 = u_{2,1}, \ldots, v_{N_x} = u_{N_x,1}, v_{N_x+1} = u_{1,2}, \ldots,$$
$$c_1 = f_{1,1}, c_2 = f_{2,1}, \ldots, c_{N_x} = f_{N_x,1}, c_{N_x+1} = f_{1,2}, \ldots,$$

which is expressed by the formulas

$$v_{(j-1)N_x+i} = u_{i,j},\ c_{(j-1)N_x+i} = f_{i,j},\ i = 1, \ldots, N_x,\ j = 1, \ldots, N_y. \quad (8.6)$$

The finite difference equation (8.5) becomes one of the equations of the matrix system (8.1):

$$a_{\ell,\ell-N_x} v_{\ell-N_x} + a_{\ell,\ell-1} v_{\ell-1} + a_{\ell,\ell} v_\ell + a_{\ell,\ell+1} v_{\ell+1} + a_{\ell,\ell+N_x} v_{\ell+N_x} = c_\ell, \quad (8.7)$$

where $\ell = (j-1)N_x + i$ is the number of the grid point (x_i, y_j), for which the approximation is written, and the coefficients of the ℓ^{th} row of matrix $\mathbf{A}$ are zero except for

$$a_{\ell,\ell-N_x} = a_{\ell,\ell+N_x} = \frac{1}{(\Delta y)^2},$$
$$a_{\ell,\ell-1} = a_{\ell,\ell+1} = \frac{1}{(\Delta x)^2}, \quad (8.8)$$
$$a_{\ell,\ell} = \left(-\frac{2}{(\Delta x)^2} - \frac{2}{(\Delta y)^2}\right).$$

The structure of the system is illustrated in Figure 8.2. We see that the matrix $\mathbf{A}$ consists mostly of zeros. The nonzero elements are $a_{\ell,\ell}$ on the main diagonal, $a_{\ell,\ell-1}$ and $a_{\ell,\ell+1}$ on the sub- and super-diagonals, and $a_{\ell,\ell-N_x}$ and $a_{\ell,\ell+N_x}$ on the diagonals distanced by N_x from the main.

Five-point discretization can be applied to other two-dimensional elliptic equations of second order. For example, let us approximate the PDE

$$b(x,y)\frac{\partial^2 u}{\partial x^2} + c(x,y)\frac{\partial^2 u}{\partial y^2} + d(x,y)\frac{\partial u}{\partial x} + f(x,y)u = g(x,y), \quad (8.9)$$

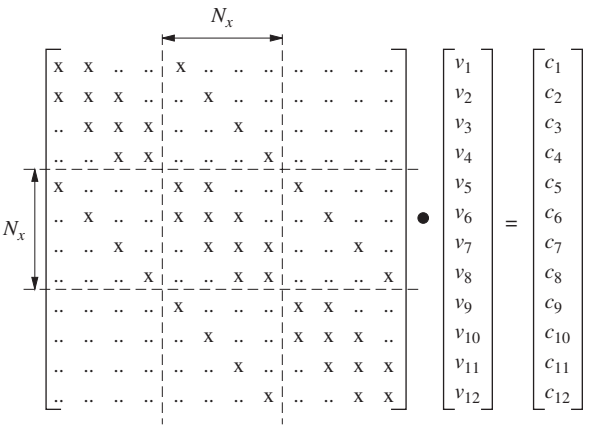


Figure 8.2 Structure of the matrix equation for the five-point discretization (8.5) of two-dimensional Poisson equation. For simplicity, we show the equation for an unnaturally crude grid with $N_x = 4$ and $N_y = 3$. In the coefficient matrix, only the elements marked by x are nonzero. The boundary condition are not taken into account. Doing so would replace some nonzero elements in the rows corresponding to the boundary grid points by zeros.

where b, c, d, f, and g are known variable coefficients and the ellipticity condition $ac > 0$ is satisfied throughout the domain Ω. Using central differences for derivatives, we obtain

$$b_{i,j}\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + c_{i,j}\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} +$$

$$d_{i,j}\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + f_{i,j}u_{i,j} = g_{i,j}. \tag{8.10}$$

The difference molecule and the structure of the matrix equation evidently remain the same, as in Figures 8.1 and 8.2. The only change is in the expressions for the nonzero elements of matrix $\mathbf{A}$.

Three-dimensional Case: The approach based on the approximation of derivatives by central differences can be easily extended to the three-dimensional case (or to the case of arbitrary dimension, should the need arise). As an example, we develop the three-dimensional version of the central difference formula (8.5) for the Poisson equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = f(x, y, z). \tag{8.11}$$

Again, we assume for simplicity that the computational grid is uniform and rectangular. In the z-direction, the grid step is Δz and the grid points are $z_k = k\Delta z$. The approximation of u at (x_i, y_j, z_k) is denoted as $u_{i,j,k}$. The finite difference scheme of the second order in all three coordinates is

$$\frac{u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}}{\Delta x^2} + \frac{u_{i,j+1,k} - 2u_{i,j,k} + u_{i,j-1,k}}{\Delta y^2} + \frac{u_{i,j,k+1} - 2u_{i,j,k} + u_{i,j,k-1}}{\Delta z^2} = f_{i,j,k}. \tag{8.12}$$

Obviously, we can order the variables $u_{i,j,k}$ and the right-hand sides into one-dimensional vectors and express (8.12) as the matrix equation (8.1).

8.1.2 Implicit Integration of Nonsteady Equations

We know from section 7.5 that a matrix equation has to be solved at every time step if an implicit scheme is used for solution of a nonsteady one-dimensional PDE. The situation remains the same in the case of a multidimensional problem. For example, let us consider two-dimensional heat conduction described by the heat equation

$$\frac{\partial u}{\partial t} = \kappa \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + \dot{Q}, \tag{8.13}$$

and apply the Crank-Nicolson method with central differences for spatial derivatives:

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} = \frac{k}{2} \left(\frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{(\Delta x)^2} + \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{(\Delta y)^2} + \frac{u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}}{(\Delta x)^2} + \frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{(\Delta y)^2} \right) + \dot{Q}_{i,j}. \tag{8.14}$$

This can be rewritten as

$$u_{i-1,j}^{n+1} \left[-\frac{\kappa}{2(\Delta x)^2} \right] + u_{i+1,j}^{n+1} \left[-\frac{\kappa}{2(\Delta x)^2} \right] + u_{i,j-1}^{n+1} \left[-\frac{\kappa}{2(\Delta y)^2} \right] + u_{i,j+1}^{n+1} \left[-\frac{\kappa}{2(\Delta y)^2} \right] + u_{i,j}^{n+1} \left[\frac{1}{\Delta t} + \frac{\kappa}{(\Delta x)^2} + \frac{\kappa}{(\Delta y)^2} \right] = c_{i,j}, \tag{8.15}$$

where indices i and j cover all internal points of the computational grid and

$$c_{i,j} = u^n_{i-1,j}\left[\frac{\kappa}{2(\Delta x)^2}\right] + u^n_{i+1,j}\left[\frac{\kappa}{2(\Delta x)^2}\right] + u^n_{i,j-1}\left[\frac{\kappa}{2(\Delta y)^2}\right] + u^n_{i,j+1}\left[\frac{\kappa}{2(\Delta y)^2}\right] + u^n_{i,j}\left[\frac{1}{\Delta t} - \frac{\kappa}{(\Delta x)^2} - \frac{\kappa}{(\Delta y)^2}\right] + \dot{Q}_{i,j} \quad (8.16)$$

is the coefficient known from the previous time step. Equations for boundary conditions have to be added to the system, but we disregard them for simplicity. Ordering the variables $u^{n+1}_{i,j}$ and the right-hand sides into one-dimensional arrays, for example, as in (8.6), we obtain the matrix equation (8.1) for the unknowns $u^{n+1}_{i,j}$.

Another example is the solution of the Navier-Stokes equations for nonsteady flows of incompressible fluids. The widely used projection algorithms, which are discussed in Chapter 10, require solution of a Poisson equation for pressure at every time step. The equation is mathematically equivalent to (8.11), and its finite difference approximation can be expressed as (8.1).

We have already considered in section 7.5 the special case of one-dimensional transient problems, where the matrix **A** is tridiagonal and (8.1) can be easily solved using the Thomas algorithm. Unfortunately, this case is an exception. In the majority of CFD problems, **A** is *not* tridiagonal, as illustrated by the example in Figure 8.2. More complex and computationally expensive methods have to be used. Some of these methods are reviewed in this chapter.

It is important to mention that the more efficient methods tend to be those utilizing the special structure of matrix **A**. In finite difference and finite volume schemes, **A** is practically never *dense*—that is, densely filled with nonzero elements. Much more often, we deal with a *sparse* matrix, the majority of elements of which are zeros. Furthermore, the nonzero elements can be clustered around the main diagonal, forming either bands or square blocks. Such matrices are called band-diagonal or block-diagonal, respectively.

8.2 DIRECT METHODS

From elementary linear algebra, we know two direct methods of solution of matrix equations: the Cramer's rule and the standard Gauss elimination. Unfortunately, these methods, while working quite well for small matrices, are practically useless for CFD purposes. The reason is the huge amount of computations they require. Let the order of matrix **A**—that is, the total

number of unknowns—be N. This can be a fairly large number in CFD. For example, in a three-dimensional problem with 100 grid points in each direction, N would be 10^6 multiplied by the number of variables—for example, four (three velocity components and pressure) in the case of simple incompressible flows. The situation with the Cramer's rule solution is particularly hopeless, since it would require calculation of N determinants, which means $\sim (N+1)!$ operations. The Gauss elimination procedure is more efficient, but still requires about $2N^3/3$ multiplications and additions.

Cramer's rule cannot be effectively modified for large matrices and is, therefore, never used. By contrast, modified Gauss elimination procedures have found some limited application.

8.2.1 Band-Diagonal and Block-Diagonal Matrices

The Gauss elimination can be greatly accelerated if the matrix $\mathbf{A}$ has band-diagonal or block-diagonal structure. Importantly for the band-diagonal matrices, the elimination operations can be arranged so that the zeros outside the filled band remain unchanged during the elimination. One can disregard these elements completely and conduct operations only on the nonzero elements. This results in the total count of arithmetic operations $\sim N$. The Thomas algorithm is an example of such algorithm suitable for tridiagonal matrices. Similar, albeit more complex, methods exist for matrices with five or more filled diagonals. Detailed description can be found in books on methods of numerical algebra, including the books listed at the end of this chapter.

The Thomas algorithm can be easily generalized to the case of a block-tridiagonal matrix, which is a matrix that has the same tridiagonal structure as (7.56), but with the number elements replaced by square submatrices (blocks) of size $M > 1$. We split the vectors of unknowns and right-hand sides into subvectors of length M as $\boldsymbol{v} = (\boldsymbol{V}_1, \ldots, \boldsymbol{V}_N)^{\mathrm{T}}$ and $\boldsymbol{c} = (\boldsymbol{C}_1, \ldots, \boldsymbol{C}_N)^{\mathrm{T}}$ and write the matrix equation (8.1) as

$$\begin{pmatrix} \mathbf{D}_0 & \mathbf{A}_0 & 0 & \ldots & \ldots & \ldots & 0 \\ \mathbf{B}_1 & \mathbf{D}_1 & \mathbf{A}_1 & 0 & \ldots & \ldots & 0 \\ 0 & \mathbf{B}_2 & \mathbf{D}_2 & \mathbf{A}_2 & 0 & \ldots & 0 \\ 0 & 0 & * & * & * & 0 & 0 \\ 0 & 0 & 0 & * & * & * & 0 \\ 0 & \ldots & \ldots & 0 & \mathbf{B}_{N-1} & \mathbf{D}_{N-1} & \mathbf{A}_{N-1} \\ 0 & \ldots & \ldots & \ldots & 0 & \mathbf{B}_N & \mathbf{D}_N \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{V}_0 \\ \boldsymbol{V}_1 \\ \boldsymbol{V}_2 \\ \vdots \\ \vdots \\ \boldsymbol{V}_{N-1} \\ \boldsymbol{V}_N \end{pmatrix} = \begin{pmatrix} \boldsymbol{C}_0 \\ \boldsymbol{C}_1 \\ \boldsymbol{C}_2 \\ \vdots \\ \vdots \\ \boldsymbol{C}_{N-1} \\ \boldsymbol{C}_N \end{pmatrix}. \tag{8.17}$$

As an example, the coefficient matrix in Figure 8.2 is block-tridiagonal with the block size $M = N_x$. The blocks, which are separated by dashed lines, form N_y rows and columns, each corresponding to a layer y_j of the computational grid. We will leave it as an exercise for the reader to develop the actual form of the block submatrices in the case of five-point approximation and other schemes. Our focus is on the general approach to solving the matrix equation (8.17).

Each step of the forward sweep of the generalized Thomas algorithm consists of inverting a diagonal submatrix and performing matrix operations on two subsequent block-rows to eliminate the subdiagonal blocks. Each such matrix operation is equivalent to a series of elementary row operations and, therefore, does not affect the solution. The first step is (compare with (7.57))

$$\begin{aligned} \mathbf{D}_1' &= \mathbf{D}_1 - \mathbf{B}_1 \cdot \mathbf{D}_0^{-1} \cdot \mathbf{A}_0 \\ \mathbf{B}_1' &= \mathbf{B}_1 - \mathbf{B}_1 \cdot \mathbf{D}_0^{-1} \cdot \mathbf{D}_0 = \mathbf{0} \\ \boldsymbol{C}_1' &= \boldsymbol{C}_1 - \mathbf{B}_1 \cdot \mathbf{D}_0^{-1} \cdot \boldsymbol{C}_0. \end{aligned} \tag{8.18}$$

The following steps are given by the general formula

$$\begin{aligned} \mathbf{D}_i' &= \mathbf{D}_i - \mathbf{B}_i \cdot \left(\mathbf{D}_{i-1}'\right)^{-1} \cdot \mathbf{A}_{i-1} \\ \mathbf{B}_i' &= \mathbf{B}_i - \mathbf{B}_i \cdot \left(\mathbf{D}_{i-1}'\right)^{-1} \cdot \mathbf{D}_{i-1} = \mathbf{0} \\ \boldsymbol{C}_i' &= \boldsymbol{C}_i - \mathbf{B}_i \cdot \left(\mathbf{D}_{i-1}'\right)^{-1} \cdot \boldsymbol{C}_{i-1}. \end{aligned} \tag{8.19}$$

At the end of the forward sweep, the matrix equation transforms into the one with upper block-triangular coefficient matrix:

$$\begin{pmatrix} \mathbf{D}_0 & \mathbf{A}_0 & 0 & \dots & \dots & \dots & 0 \\ 0 & \mathbf{D}_1' & \mathbf{A}_1 & 0 & \dots & \dots & 0 \\ 0 & 0 & \mathbf{D}_2' & \mathbf{A}_2 & 0 & \dots & 0 \\ 0 & 0 & 0 & * & * & 0 & 0 \\ 0 & 0 & 0 & 0 & * & * & 0 \\ 0 & \dots & \dots & 0 & 0 & \mathbf{D}_{N-1}' & \mathbf{A}_{N-1} \\ 0 & \dots & \dots & \dots & 0 & 0 & \mathbf{D}_N' \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{V}_0 \\ \boldsymbol{V}_1 \\ \boldsymbol{V}_2 \\ \vdots \\ \vdots \\ \boldsymbol{V}_{N-1} \\ \boldsymbol{V}_N \end{pmatrix} = \begin{pmatrix} \boldsymbol{C}_0 \\ \boldsymbol{C}_1' \\ \boldsymbol{C}_2' \\ \vdots \\ \vdots \\ \boldsymbol{C}_{N-1}' \\ \boldsymbol{C}_N' \end{pmatrix}. \tag{8.20}$$

The backward sweep is trivial. We invert the diagonal submatrices of the transformed system $\mathbf{D}_i'$ and calculate first

$$\boldsymbol{V}_N = \left(\mathbf{D}_N'\right)^{-1} \cdot \boldsymbol{C}_N' \tag{8.21}$$

and then the rest of the solution according to

$$\begin{aligned} \boldsymbol{V}_i &= \left(\mathbf{D}_i'\right)^{-1} \cdot \left(\boldsymbol{C}_i' - \mathbf{A}_i \cdot \boldsymbol{V}_{i+1}\right), \quad i = N-1, \ldots, 1, \\ \boldsymbol{V}_0 &= \mathbf{D}_0^{-1} \cdot (\boldsymbol{C}_0 - \mathbf{A}_0 \cdot \boldsymbol{V}_1). \end{aligned} \tag{8.22}$$

8.2.2 LU Decomposition

Another variation of the Gauss elimination found its use in the CFD applications, where the matrix $\mathbf{A}$ is not sparse, for example, in spectral methods. It is based on factorization (decomposition) of $\mathbf{A}$ into a product of a lower-triangle matrix $\mathbf{L}$ and upper-triangular matrix $\mathbf{U}$. The procedure of factorization, a description of which can be found in the books on numerical linear algebra, is a version of the Gauss elimination and requires $\sim N^3$ operations. Here, we only discuss the consequences. Rewriting (8.1) as

$$\mathbf{L} \cdot \mathbf{U} \cdot \boldsymbol{v} = \boldsymbol{c} \tag{8.23}$$

and introducing a new vector $\boldsymbol{w} = \mathbf{U} \cdot \boldsymbol{v}$ we can split the equation (8.1) into two:

$$\mathbf{L} \cdot \boldsymbol{w} = \boldsymbol{c} \tag{8.24}$$

and

$$\mathbf{U} \cdot \boldsymbol{v} = \boldsymbol{w}. \tag{8.25}$$

The equations are solved sequentially, first (8.24) and then (8.25). Since the matrices $\mathbf{L}$ and $\mathbf{U}$ are triangular, the solution can be easily obtained by backward substitution (operation count $\sim N^2$). A nice feature of the method is that the only computationally expensive part, the decomposition of $\mathbf{A}$, does not use the right-hand side vector $\boldsymbol{c}$. This becomes useful if (8.1) has to be solved many times with the same matrix $\mathbf{A}$ but different right-hand sides. One can perform the decomposition once and apply (8.24)–(8.25) as many times as required.

8.3 ITERATIVE METHODS

Unlike direct methods, iterative methods are not designed to find an exact (up to the computer round-off error) solution of the matrix equation (8.1). Instead, the methods rely on successive iterations to obtain a sufficiently

accurate approximation of $\boldsymbol{v}$. One important consideration justifies this approach in CFD. The matrix equation is generated by discretization, so its solution inevitably contains some discretization error. We can allow ourselves an additional error of an iteration procedure without significant drop of the overall accuracy provided this error is much smaller than the discretization one. Furthermore, the iterative methods can be arranged so that they utilize the sparse character of matrix $\mathbf{A}$. This greatly reduces the computational cost of the solution and makes iterative methods especially attractive for finite difference and finite volume applications.

The main characteristics of an iterative method are the ability to converge (to achieve an approximation accurate within a given tolerance) and the computational cost of the procedure. The cost is determined by the number of iterations needed for convergence (speed of convergence) and the amount of computations required to complete a single iteration. Both should be small or, at least, not very large for the iteration procedure to be effective.

Numerous iterative methods have been developed over the last decades. We will consider some algorithms that, albeit simple, possess the principal features of the commonly used techniques.

8.3.1 General Methodology

The general iteration procedure used to solve the matrix equation (8.1) is as follows:

1. Guess an initial approximation $\boldsymbol{v}^{(0)}$ of the solution.
2. Use $\mathbf{A}$, $\boldsymbol{c}$, and the already-achieved approximation to find the next, more accurate approximation $\boldsymbol{v}^{(k)}$.
3. Repeat Step 2 iteratively until the convergence criterion is satisfied.

A few comments are in order concerning the definition and realization of the convergence criterion. The definition of the error is

$$\boldsymbol{\epsilon}^{(k)} = \boldsymbol{v} - \boldsymbol{v}^{(k)}, \tag{8.26}$$

where $\boldsymbol{v}$ is the exact solution of (8.1) and $\boldsymbol{v}^{(k)}$ is the approximation obtained after the kth iteration. Since $\boldsymbol{v}$ is typically unavailable during the solution, the error cannot be found by direct application of (8.26). Instead, we can find the difference between successive approximations and evaluate its norm as

$$\delta^{(k)} = \|\boldsymbol{v}^{(k+1)} - \boldsymbol{v}^{(k)}\| = \max_{\ell} |v_{\ell}^{(k+1)} - v_{\ell}^{(k)}| \tag{8.27}$$

or

$$\delta^{(k)} = \|\boldsymbol{v}^{(k+1)} - \boldsymbol{v}^{(k)}\| = \left[\sum_{\ell=1}^{N} \left(v_\ell^{(k+1)} - v_\ell^{(k)}\right)^2\right]^{1/2}. \tag{8.28}$$

If the iterative procedure converges, $\boldsymbol{v}^{(k)}$ and $\boldsymbol{v}^{(k+1)}$ both tend to the exact solution $\boldsymbol{v}$, so the norm of the difference tends to zero. The magnitude of $\delta^{(k)}$ is often used as a convergence criterion. The iterations are stopped when it becomes smaller than the desired tolerance ϵ_0. Note that determining ϵ_0 that guarantees the desired accuracy of the solution is, by itself, a nontrivial problem. We discuss this issue in section 13.2.1.

Another approach is to monitor the norm of the residual vector

$$\boldsymbol{r}^{(k)} = \boldsymbol{c} - \mathbf{A} \cdot \boldsymbol{v}^{(k)}. \tag{8.29}$$

It is easy to see that

$$\mathbf{A} \cdot \boldsymbol{\epsilon}^{(k)} = \mathbf{A} \cdot \left(\boldsymbol{v} - \boldsymbol{v}^{(k)}\right) = \boldsymbol{c} - \mathbf{A} \cdot \boldsymbol{v}^{(k)} = \boldsymbol{r}^{(k)} \tag{8.30}$$

so the residual vanishes with $\boldsymbol{\epsilon}^{(k)}$, as $\boldsymbol{v}^{(k)}$ converged to the exact solution $\boldsymbol{v}$. The criterion of convergence is $\|\boldsymbol{r}^{(k)}\| < \epsilon_0$.

The initial guess $\boldsymbol{v}^{(0)}$ can, in general, be an arbitrary vector. For the sake of faster convergence, however, $\boldsymbol{v}^{(0)}$ should be chosen as close to the exact solution as possible.

8.3.2 Jacobi Iterations

We start with the simplest method, the Jacobi iteration algorithm. The method is rather inefficient and is presented here solely to illustrate the basic approach. First, we rewrite the equations of the system (8.1) so that the first equation is an expression of v_1 through v_2, v_3, ..., the second equation is an expression of v_2 through v_1, v_3, ..., and so on. In general, the ℓth equation becomes

$$v_\ell = \frac{1}{a_{\ell\ell}} \left(c_\ell - \sum_{j=1}^{\ell-1} a_{\ell j} v_j - \sum_{j=\ell+1}^{N} a_{\ell j} v_j\right), \quad \ell = 1, \ldots, N. \tag{8.31}$$

The exact solution $\boldsymbol{v}$ satisfies this equation exactly. In the Jacobi method, we use the results of the previous iteration to compute the right-hand side and assign the left-hand side as the new approximation:

$$v_\ell^{(k+1)} = \frac{1}{a_{\ell\ell}} \left(c_\ell - \sum_{j=1}^{\ell-1} a_{\ell j} v_j^{(k)} - \sum_{j=\ell+1}^{N} a_{\ell j} v_j^{(k)}\right). \tag{8.32}$$

This formula shows why the iteration methods are particularly effective if the matrix **A** is sparse. Only few terms of the sums in the right-hand side of (8.32) are nonzero. For example, let us write the Jacobi iteration formula for the five-point discretization of the Poisson equation (8.7):

$$v_\ell^{(k+1)} = \frac{1}{a_{\ell,\ell}} \left(c_\ell - a_{\ell,\ell-N_x} v_{\ell-N_x}^{(k)} - a_{\ell,\ell-1} v_{\ell-1}^{(k)} - a_{\ell,\ell+1} v_{\ell+1}^{(k)} - a_{\ell,\ell+N_x} v_{\ell+N_x}^{(k)} \right). \tag{8.33}$$

There are only five nonzero terms in the right-hand side. Computing only them and explicitly omitting the others reduces the computational cost of each iteration from $\sim N^2$ to $\sim N$ operations.

8.3.3 Gauss-Seidel Algorithm

The Gauss-Seidel method is an improvement of the Jacobi algorithm, which results in faster convergence. The right-hand side of (8.32) is constantly updated using already found $v_j^{(k+1)}$. If a new approximation is calculated for the number of equation ℓ going from 1 to N, the iterations are

$$v_\ell^{(k+1)} = \frac{1}{a_{\ell\ell}} \left(c_\ell - \sum_{j=1}^{\ell-1} a_{\ell j} v_j^{(k+1)} - \sum_{j=\ell+1}^{N} a_{\ell j} v_j^{(k)} \right). \tag{8.34}$$

For the Laplace and Poisson equation (8.3) and (8.4), solution by the Gauss-Seidel algorithm requires approximately two times fewer iterations than the solution by the Jacobi algorithm.

As an example, let us solve the two-dimensional Poisson equation using the five-point scheme. Each iteration can be arranged so that we begin at the lower-left corner of the computational domain and move to the right, covering one row after another (see Figure 8.3). The Gauss-Seidel version of the Jacobi iteration (8.33) is

$$v_\ell^{(k+1)} = \frac{1}{a_{\ell,\ell}} \left(c_\ell - a_{\ell,\ell-N_x} v_{\ell-N_x}^{(k+1)} - a_{\ell,\ell-1} v_{\ell-1}^{(k+1)} - a_{\ell,\ell+1} v_{\ell+1}^{(k)} - a_{\ell,\ell+N_x} v_{\ell+N_x}^{(k)} \right), \tag{8.35}$$

or, in terms of the grid point values,

$$u_{i,j}^{(k+1)} = \frac{1}{(2/(\Delta x)^2 + 2/(\Delta y)^2)} \left(-f_{i,j} + \right.$$
$$\left. \frac{1}{(\Delta y)^2} u_{i,j-1}^{(k+1)} + \frac{1}{(\Delta x)^2} u_{i-1,j}^{(k+1)} + \frac{1}{(\Delta y)^2} u_{i,j+1}^{(k)} + \frac{1}{(\Delta x)^2} u_{i+1,j}^{(k)} \right). \tag{8.36}$$

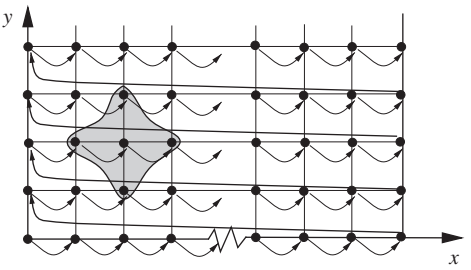


Figure 8.3 A Gauss-Seidel iteration for the five-point scheme (8.5) applied to a two-dimensional elliptic equation. Arrows illustrate the order in which the grid point values are updated. A difference molecule is also shown.

8.3.4 Successive Over- and Underrelaxation

Successive *over- and underrelaxations* are the techniques that can be applied to accelerate convergence of the iterative methods such as the Gauss-Seidel algorithm. The principal idea is to determine the direction in which the approximate solution $\boldsymbol{v}^{(k)}$ evolves with k and correct the solution between the iterations so as to accelerate this evolution. The correction is achieved by performing the simple operation

$$\boldsymbol{v}^* = \boldsymbol{v}^{(k)} + \omega \left(\boldsymbol{v}^{(k+1)} - \boldsymbol{v}^{(k)} \right) \tag{8.37}$$

and using $\boldsymbol{v}^*$ as a more accurate new approximation instead of $\boldsymbol{v}^{(k+1)}$. The technique is schematically shown in Figure 8.4.

The difference between the over- and underrelaxation is in the value of the coefficient ω. For the problems, where, as illustrated in Figure 8.4a, the

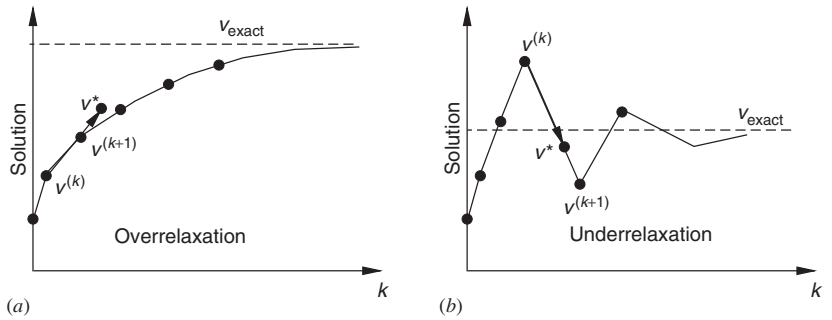


Figure 8.4 Over- and underrelaxation improvements of an iteration procedure.

approximations converge to the exact solution *monotonically*, the convergence can be accelerated by using $1 < \omega < 2$, which enhances the step in the right direction. This method is called *overrelaxation*. When, as shown in Figure 8.4b, the iterations lead to nonmonotonic behavior, the overrelaxation would compromise the convergence. Acceleration can, however, be achieved if a value $0 < \omega < 1$ is used thus reducing the overshoot. This method is called *underrelaxation*. The underrelaxation is sometimes used to achieve convergence of otherwise diverging iteration procedures.

Optimal values of ω that provide the fastest convergence have been found theoretically for simple model problems, such as the Laplace or Poisson equations in rectangular domains. For complex CFD problems, there are no rigorous theoretical results, but the estimates based on experience and extrapolation of theoretical data are usually available.

8.3.5 Convergence of Iterative Procedures

The question left out so far is that of convergence. Can we be sure, when we start the computations, that the iterations will converge to a final equilibrium point? What is the speed of convergence?

The answer to the first question for the Gauss-Seidel procedure is given by the following sufficient condition. The Gauss-Seidel procedure applied to a system of linear equations converges if the system is irreducible (cannot be separated into smaller decoupled systems, in other words, rearranged in such a way that some unknowns can be found without solving the entire system) and if the following is true for the coefficient matrix:

$$|a_{\ell\ell}| \geq \sum_{j=1, j\neq\ell}^{N} |a_{\ell j}| \text{ for all } \ell \text{ and } |a_{\ell\ell}| > \sum_{j=1, j\neq\ell}^{N} |a_{\ell j}| \text{ for some } \ell. \quad (8.38)$$

The condition (8.38) is referred to as the *diagonal dominance* of matrix **A**.

If the system of finite difference equations does not satisfy the convergence condition (8.38), it can often be rearranged by changing the order of unknowns so that the largest coefficients lie on the diagonal of **A**. The new system may satisfy the condition. The diagonal dominance is not a necessary condition. Quite often, the Gauss-Seidel iterations converge even though (8.38) is not satisfied.

As an example, let us again consider the five-point scheme for the Poisson equation, rewritten as (8.36). We see that no rearranging is required. The diagonal coefficient $a_{\ell,\ell} = \left[(2/\Delta x)^2 + 2/(\Delta y)^2\right]$ is the largest element in each row. It is exactly equal to the sum of all the other coefficients. The first condition in (8.38) is, thus, satisfied. The second condition that,

at least in one row, $|a_{\ell\ell}|$ is greater than the sum of the other coefficients, is normally satisfied by the rows of $\mathbf{A}$ corresponding to the boundary points of the computational domain. For example, if the Dirichlet boundary condition $u = g$ is imposed at the point (x_0, y_j) of the grid, the equation is

$$u_{0,j} = g_j$$

The corresponding row matrix $\mathbf{A}$ has only one non-zero element, and this element is on the main diagonal.

The rate of convergence varies with the type of the problem, method, and the grid size. There is no general theory that would be valid for CFD computations. The conclusions are often made based on numerical experiments. There are several well-established trends, though.

As an illustration, we solve the two-dimensional Poisson equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \sin(\pi x)\sin(\pi y)$$

in the square domain $[0, 1] \times [0, 1]$ with homogeneous boundary conditions $u = 0$ at all boundaries. The equation is discretized on a uniform grid using the five-point scheme (8.5). The matrix equation is solved by three simple iterative methods: Jacobi (8.33), Gauss-Seidel (8.35), and Gauss-Seidel with successive overrelaxation with $\omega = 1.5$. To illustrate the effect of numerical resolution on convergence, the solution is computed using three grids: with $N_x = N_y = 10$, $N_x = N_y = 30$, and $N_x = N_y = 100$. The initial guess $u_{i,j}^{(0)} = 0$ is used for all cases. The convergence is monitored by the norm $||\boldsymbol{r}^k|| = \left[\sum_{i,j}\left(r_{i,j}^{(k)}\right)^2\right]^{1/2}$ of the residual vector (8.29). The calculations continue until the norm reduces below $10^{-6}||\boldsymbol{r}^0||$.

The results are shown in Table 8.1 and Figure 8.5. They illustrate the commonly observed (although not absolute) trends:

1. Relaxation methods (with properly chosen ω) are faster than Gauss-Seidel, which, in turn, is faster than Jacobi.
2. Finer grid (larger N_x and N_y) requires a larger (sometimes much larger) number of iterations to achieve the same reduction of residual.

Considering the second trend, we should also remember that the computational cost of each iteration increases proportionally to the total number of grid points. We see that using a finer grid requires larger, perhaps much larger, amount of computational operations. In our example, the computational cost of the solution with $N_x = N_y = 100$ is about four orders

Table 8.1 Example of Performance of Iteration Methods. Poisson Equation is Solved in a Rectangular Domain Using the Finite Difference Scheme of the Second Order on Three Different Computational Grids. The Table Shows the Number of Iterations Needed to Reduce the Norm of Residual $||\boldsymbol{r}^{(k)}||$ Below $10^{-6}||\boldsymbol{r}^{(0)}||$

Method	$N_x = N_y = 10$	$N_x = N_y = 30$	$N_x = N_y = 100$
Jacobi	276	2516	27992
Gauss-Seidel	139	1259	13997
SOR	90	837	9329

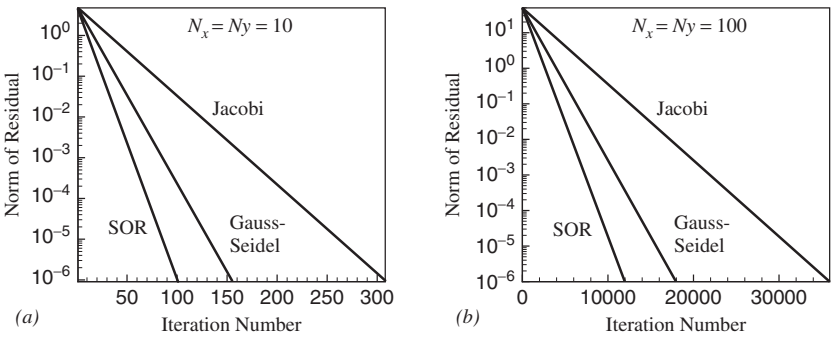


Figure 8.5 Example of performance of Jacobi, Gauss-Seidel, and SOR iteration methods (see Table 8.1 and text). Norm of residual $||\boldsymbol{r}^{(k)}||$ (see (8.29) is shown as a function of the iteration number k. Note that the abscissa scales in the two graphs are different by about two orders of magnitude.

of magnitude higher than the cost of the solution with $N_x = N_y = 10$. Solutions on fine grids are nevertheless often unavoidable, since solutions computed on crude grids have unacceptably large discretization errors.

In general there is, typically, a need for a compromise between the small discretization error achievable on fine grids and faster convergence provided by crude grids. Even better would be a strategy that combines the advantages of fine and crude grids. One simple approach is to start computations on a crude grid, quickly reach convergence, and then interpolate the computed variables to a fine grid and use it as a good initial approximation $\boldsymbol{v}^{(0)}$ for a fine grid solution. A more advanced, efficient, and widely used strategy based on the multigrid method is discussed in the next section.

8.3.6 Multigrid Methods

An important observation can be made regarding the error $\boldsymbol{\epsilon}^{(k)} = \boldsymbol{v}^{(k)} - \boldsymbol{v}$ of iteration solutions of matrix equations. The components of the error having typical length scales comparable to the grid steps vanish quickly, after just few iterations. These components will be referred to as *small-scale* in the following discussion. On the contrary, the large-scale components (with the length scales larger or much larger than the grid steps) decrease slowly, require a larger number of iterations, and are, in general, responsible for slow convergence on finer grids.

We can now conjecture the main idea of the multigrid method. The large-scale error can be accurately represented on a coarser grid. Iterations on such a grid can be performed with lower computational cost. Furthermore, the components of the error, which are large-scale on a fine grid, are small-scale on a coarser grid. Their removal occurs faster if iterations are performed on a coarser grid. Why not use several grids of different degrees of coarseness interpolating solutions between them and allowing each grid to take care of the respective length scales?

We will give a sketch of a simple version of the strategy. As an example, we will solve a two-dimensional elliptic problem expressed in the general matrix form (8.1). Two uniform structured grids shown in Figure 8.6 will be used. One is the fine grid with steps Δx and Δy, on which the actual solution is to be found. We also introduce a coarse grid that includes only every second grid point in each direction. Its steps are $2\Delta x$ and $2\Delta y$.

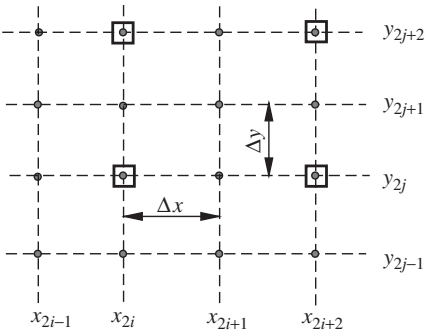


Figure 8.6 Example of a multigrid approach. Fine (circles) and coarse (squares) grids are used to solve a two-dimensional elliptic problem.

The multigrid iteration procedure is as follows:

Step 1: Conduct k iterations on the fine grid. The result is the k-th approximation $\boldsymbol{v}^{(k)}$. Calculate the residual $\boldsymbol{r}^{(k)} = \boldsymbol{c} - \mathbf{A} \cdot \boldsymbol{v}^{(k)}$.

Step 2: The error of approximation $\boldsymbol{\epsilon}^{(k)} = \boldsymbol{v} - \boldsymbol{v}^{(k)}$ cannot be calculated directly, but we know that it satisfies the correction equation (see (8.30))

$$\mathbf{A} \cdot \boldsymbol{\epsilon}^{(k)} = \boldsymbol{r}^{(k)}. \tag{8.39}$$

We can also assume that, after the k iterations of Step 1 performed on the fine grid, the error $\boldsymbol{\epsilon}^{(k)}$ mostly contains fluctuations with length scales larger than Δx and Δy. Let us estimate the error applying efficient coarse grid iterations to the correction equation (8.39). For this purpose, we restrict (8.39) (both the coefficients of matrix $\mathbf{A}$ and the right-hand-side $\boldsymbol{r}^{(k)}$) to the points of the coarse grid. After several iterations, we obtain a coarse grid approximation of the error, which we denote as $\tilde{\boldsymbol{\epsilon}}^{(k)}$.

Step 3: Interpolate $\tilde{\boldsymbol{\epsilon}}^{(k)}$ onto the fine grid and update the solution as $\tilde{\boldsymbol{v}}^{(k)} = \boldsymbol{v}^{(k)} + \tilde{\boldsymbol{\epsilon}}^{(k)}$. If $\tilde{\boldsymbol{\epsilon}}^{(k)}$ is an accurate approximation of $\boldsymbol{\epsilon}^{(k)}$, the updated field $\tilde{\boldsymbol{v}}^{(k)}$ is much closer to the exact solution than the original $\boldsymbol{v}^{(k)}$.

Step 4: Check convergence. If not achieved, repeat starting with Step 1 and using $\tilde{\boldsymbol{v}}^{(k)}$ as a new initial guess.

It is important to stress that the procedure outlined here is only a simple example. Actual multigrid solvers are quite complex and diverse. They are routinely applied to unstructured and nonuniform grids. Moreover, the advanced multigrid solvers typically use several embedded grids of various degrees of coarseness.

At least a brief discussion is needed of the methods used to transfer the information between the fine and coarse grids. We will limit the discussion to the example illustrated in Figure 8.6. The variables computed on the fine grid will be denoted as $u^{\Delta}_{i,j}$, and their counterparts on the coarse grid as $u^{2\Delta}_{i,j}$. Let the points used for the coarse grid be those with even indices (x_{2i}, y_{2j}) (see Figure 8.6). The restriction of a fine grid solution onto the coarse grid can be achieved if we directly transfer the values at the common grid points and discard the others:

$$u^{2\Delta}_{2i,2j} = u^{\Delta}_{2i,2j}, \quad \text{discard } u^{\Delta}_{i,j} \text{ if } i \text{ or } j \text{ is odd.} \tag{8.40}$$

A better method is the *full weighting*, in which we evaluate a coarse grid value as an average of the fine grid solution over the surrounding area. Numerical approximations of integrals are used to compute the averages. The full weighting operation has several advantages over the direct transfer. It can be applied to unstructured grids, in which the points of fine and coarse grids do not necessarily coincide. It clearly makes better use of the information available from the fine grid solution. At last, it commutes with the linear interpolation procedure discussed next. In our example of two-dimensional structured uniform grids shown in Figure 8.6, the full weighting formula is (see Figure 8.7)

$$u^{2\Delta}_{2i,2j} = \frac{1}{16}\Big[u^{\Delta}_{2i-1,2j-1} + u^{\Delta}_{2i-1,2j+1} + u^{\Delta}_{2i+1,2j-1} + u^{\Delta}_{2i+1,2j+1} + 2\left(u^{\Delta}_{2i,2j-1} + u^{\Delta}_{2i,2j+1} + u^{\Delta}_{2i-1,2j} + u^{\Delta}_{2i+1,2j}\right) + 4u^{\Delta}_{2i,2j}\Big]. \quad (8.41)$$

The inverse operation of transformation from a coarse grid to a fine grid is done by one of the interpolation methods. The most common is the linear interpolation, which has the accuracy of the second order. In the case of our two-grid example, the linear interpolation is

$$\begin{aligned} u^{\Delta}_{2i,2j} &= u^{2\Delta}_{2i,2j}, \\ u^{\Delta}_{2i+1,2j} &= \left(u^{2\Delta}_{2i,2j} + u^{2\Delta}_{2i+2,2j}\right)/2, \\ u^{\Delta}_{2i,2j+1} &= \left(u^{2\Delta}_{2i,2j} + u^{2\Delta}_{2i,2j+2}\right)/2, \\ u^{\Delta}_{2i+1,2j+1} &= \left(u^{2\Delta}_{2i,2j} + u^{2\Delta}_{2i,2j+2} + u^{2\Delta}_{2i+2,2j} + u^{2\Delta}_{2i+2,2j+2}\right)/4. \end{aligned} \quad (8.42)$$

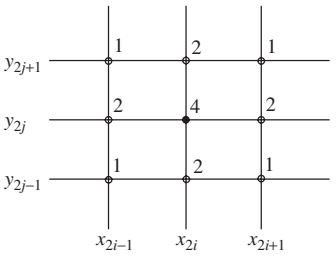


Figure 8.7 Full-weighting restriction on a two-dimensional uniform grid. The coarse-grid value $u^{2\Delta}_{2i,2j}$ is evaluated as a combination (8.41) of fine-grid values of a nine-point stencil with weight coefficients as indicated in the figure.

8.3.7 Pseudo-transient Approach

Another approach to solution of a steady-state problem is to introduce fictitious "time" by adding "time derivative" to the equation. For example, the elliptic equations (8.4) and (8.3) are replaced by the parabolic equations:

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} - f \tag{8.43}$$

and

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}. \tag{8.44}$$

Solutions of parabolic systems converge to steady states, unless they are forced to do otherwise by time-dependent boundary conditions or source terms. Evidently, convergence should be expected in the case of the pseudo-transient approach. The resulting steady states correspond to the solutions of the original elliptic equations. For example, the steady-state solutions of (8.43) and (8.44) are the solutions of (8.4) and (8.3).

The parameter τ does not have a physical meaning. Its only purpose is to facilitate the iteration procedure of finding a solution of a steady-state problem. The iterations can now be considered as "time steps" of a marching scheme. This allows us to use all the methods developed for nonsteady problems. For example, (8.43) and (8.44) can be solved by the simple explicit method with central differences for $\partial^2 u/\partial x^2$ and $\partial^2 u/\partial y^2$ or by the ADI method described in section 9.3.3.

8.4 SYSTEMS OF NONLINEAR EQUATIONS

We conclude the chapter by a discussion of methods used to solve nonlinear equations. At first glance, it might seem that this discussion should have started the chapter. The equations describing convection heat transfer and fluid flows are nonlinear, after all. Nonlinearity is always present in the convective flux terms of the momentum, mass, and energy conservation equations. Even conduction heat transfer may require nonlinear equations—for example, if variation of physical properties with temperature is nonnegligible. CFD solution of any such problem involves solution of a system of nonlinear algebraic equations resulting from finite difference, finite volume, or other discretization.

We started this chapter by considering linear systems because direct solution of nonlinear equations is, in fact, avoided by virtually all practical CFD methods. The nonlinear systems are difficult and computationally

expensive to solve. For this reason, the common approach is to rely on linearization and/or multistep algorithms to reduce the task to the solution of linear matrix equations.

This section provides a brief review of the three methods for nonlinear systems: the Newton method included for the sake of completness and the iteration procedures based on linearization and sequential iterations.

8.4.1 Newton's Algorithm

We solve a general system of nonlinear equations

$$f_j(v_1, \ldots, v_n) = 0,\ j = 1, \ldots, n, \tag{8.45}$$

which is expressed in vector form as

$$\boldsymbol{F}(\boldsymbol{v}) = 0. \tag{8.46}$$

The functions f_j are assumed to have the needed differentiability and smoothness properties.

Many methods used to solve such systems are based on the Newton's algorithm, also called the Newton-Raphson algorithm. The basis formula is obtained by taking the multidimensional Taylor series expansion of (8.46) and truncating all but the first-order terms. Let us assume that we already have an approximation $\boldsymbol{v}^{(k)}$ of the solution. Truncated expansion around it gives, for the jth equation in (8.45),

$$f_j(v_1, \ldots, v_n) = f_j(v_1^{(k)}, \ldots, v_n^{(k)}) + \sum_{i=1}^{n} \frac{\partial f_j}{\partial v_i}(v_1^{(k)}, \ldots, v_n^{(k)}) \left(v_i - v_i^{(k)}\right). \tag{8.47}$$

If $\boldsymbol{v}$ is a solution of (8.46), the left-hand side of (8.47) is zero and we obtain a linear equation for $\boldsymbol{v}$:

$$\sum_{i=1}^{n} \frac{\partial f_j}{\partial v_i}(v_1^{(k)}, \ldots, v_n^{(k)}) \left(v_i - v_i^{(k)}\right) = -f_j(v_1^{(k)}, \ldots, v_n^{(k)}) \tag{8.48}$$

or, in the matrix form,

$$\mathbf{J}^{(k)} \cdot \left(\boldsymbol{v} - \boldsymbol{v}^{(k)}\right) = -\boldsymbol{F}(\boldsymbol{v}^{(k)}), \tag{8.49}$$

where $\mathbf{J} = \{\partial f_j / \partial v_i\}$ is the Jacobian matrix of $\boldsymbol{F}(\boldsymbol{v})$. Now we recall that (8.47) is just a first-order approximation of (8.45), so $\boldsymbol{v}$ in (8.49) is not

a solution but rather the next, better approximation. This leads to the Newton's iteration formula

$$\mathbf{J}^{(k)} \cdot \left(\boldsymbol{v}^{(k+1)} - \boldsymbol{v}^{(k)} \right) = -\boldsymbol{F}(\boldsymbol{v}^{(k)}) \tag{8.50}$$

and

$$\mathbf{J}^{(k)} \cdot \boldsymbol{v}^{(k+1)} = \mathbf{J}^{(k)} \cdot \boldsymbol{v}^{(k)} - \boldsymbol{F}(\boldsymbol{v}^{(k)}) \equiv \boldsymbol{c}^{(k)}, \tag{8.51}$$

which is the familiar matrix equation.

The formulas (8.50) and (8.51) may look attractive on paper but are, in fact, practically never applicable to CFD problems because of the significant additional computational cost involved into the calculation of the Jacobian $\mathbf{J}$, which has to be done anew at every iteration. Furthermore, the nature of functions f_i may be such that their differentiation needed to evaluate $\mathbf{J}$ is either impossible or very difficult. A variety of more efficient methods based on modification of the Newton's algorithm have been developed. They, too, found no viable applications in CFD.

8.4.2 Iteration Methods Using Linearization

The iteration methods discussed in this chapter for the linear systems can be utilized to solve systems of nonlinear equations. The commonly used approach is to linearize the equations replacing the unknown coefficients by their estimates taken from the previous iteration. For example, the quadratic convection term vw (here, v and w are the velocity components) in the momentum equation can be linearized at the $(k+1)$st iteration as

$$v^{k+1} w^{k+1} \approx v^k w^{k+1}. \tag{8.52}$$

We will illustrate the procedure on the example of the steady-state conduction heat transfer in the situation, when the variation of temperature is large, so the temperature dependence of physical properties cannot be neglected. The heat conduction equation is

$$\nabla (\kappa \nabla u) = \dot{Q}, \tag{8.53}$$

where u is temperature, $\dot{Q}$ is the intensity of internal heat sources, and the conductivity is now not a constant, but $\kappa = \kappa(u(\boldsymbol{x}))$.

Let us apply the central difference discretization similar to the discretization (4.39) derived for the one-dimensional heat equation. In two

dimensions, we obtain for (8.53):

$$\frac{1}{\Delta x}\left(\frac{\kappa_{i+1,j}+\kappa_{i,j}}{2}\frac{u_{i+1,j}-u_{i,j}}{\Delta x}-\frac{\kappa_{i,j}+\kappa_{i-1,j}}{2}\frac{u_{i,j}-u_{i-1,j}}{\Delta x}\right)+$$
$$\frac{1}{\Delta y}\left(\frac{\kappa_{i,j+1}+\kappa_{i,j}}{2}\frac{u_{i,j+1}-u_{i,j}}{\Delta y}-\frac{\kappa_{i,j}+\kappa_{i,j-1}}{2}\frac{u_{i,j}-u_{i,j-1}}{\Delta y}\right)=\dot{Q}_{i,j}. \tag{8.54}$$

The entire system of equations can be symbolically written as

$$\boldsymbol{N}(\boldsymbol{\kappa},\boldsymbol{u})=\boldsymbol{Q}. \tag{8.55}$$

The system is linearized by approximating $\boldsymbol{\kappa}$ by it estimate known from the previous iteration. The system of linear equations

$$\boldsymbol{N}(\boldsymbol{\kappa}^{(k)},\boldsymbol{u}^{(k+1)})=\boldsymbol{Q} \tag{8.56}$$

is solved on the $(k+1)$st iteration. Returning to the finite difference formula, we can write the linearized equation as

$$\frac{1}{\Delta x}\left(\frac{\kappa^{(k)}_{i+1,j}+\kappa^{(k)}_{i,j}}{2}\frac{u^{(k+1)}_{i+1,j}-u^{(k+1)}_{i,j}}{\Delta x}-\frac{\kappa^{(k)}_{i,j}+\kappa^{(k)}_{i-1,j}}{2}\frac{u^{(k+1)}_{i,j}-u^{(k+1)}_{i-1,j}}{\Delta x}\right)+$$
$$\frac{1}{\Delta y}\left(\frac{\kappa^{(k)}_{i,j+1}+\kappa^{(k)}_{i,j}}{2}\frac{u^{(k+1)}_{i,j+1}-u^{(k+1)}_{i,j}}{\Delta y}-\frac{\kappa^{(k)}_{i,j}+\kappa^{(k)}_{i,j-1}}{2}\frac{u^{(k+1)}_{i,j}-u^{(k+1)}_{i,j-1}}{\Delta y}\right)=\dot{Q}^{(k+1)}_{i,j}. \tag{8.57}$$

The system of linearized equations can be solved by any of the direct or iterative methods discussed earlier in this chapter. The solution is used to evaluate $\boldsymbol{\kappa}^{(k+1)}=\boldsymbol{\kappa}(\boldsymbol{u}^{(k+1)})$, which is employed at the next iteration. The procedure continues until convergence to a sufficiently accurate approximation of the solution of the nonlinear problem (8.55) is achieved.

The nonlinearity complicates the issue of convergence, which typically cannot be guaranteed a-priori and should be determined experimentally. Another consequence of the nonlinearity is that the convergence of the iteration procedures may require strong underrelaxation. The method typically needs a larger number of iterations than the Newton's algorithm, but each iteration is computationally less expensive to perform. The efficiency can be further improved by applying optimized underrelaxation and multigrid techniques.

8.4.3 Sequential Solution

The linearized iterations procedure can be applied to solve steady-state problems in fluid dynamics and convection heat transfer, but becomes cumbersome and computationally expensive for multidimensional flows. More efficient methods have been developed that utilize specific properties of the PDE systems describing the flows.

One such property is that a dominant variable can be assigned to every evolutionary equation in the sense that the equation describes transport and conservation of this particular variable. For example, the momentum component ρu_i is the dominant variable for the ith component of the momentum equation, and internal energy e is the dominant variable of the energy conservation equation (see Chapter 2).

The iteration approach is modified so that each PDE is solved separately for its dominant variable as an unknown. The other variables in the equation are replaced by the best currently available estimates and treated as known. The procedure is conducted for the entire system but sequentially, one PDE at a time. Of course, the result is not a correct solution, since the coupling between the equations is disregarded. For this reason, iterations are necessary. On every cycle, we solve the equations with the results found at the previous cycle serving as the best available estimates of non-dominant variables.

The algorithm, which has the name of *sequential iteration method*, consists of the two sets of iterations, one embedded into another. There are two principal steps:

Step 1: Solve each PDE for the dominant variable with other variables replaced by the best available estimates. Since the equations themselves are inaccurate, it is unnecessary to invest computational resources to achieve very high accuracy at this step. Typically, an iterative procedure with relatively few cycles is applied. These cycles are called inner iterations.

Step 2: Verify convergence of the new approximation $\boldsymbol{v}^{(k)}$, substituting it into the nonlinear equations and computing the norm of residuals. If the convergence is not achieved, repeat step 1 using $\boldsymbol{v}^{(k)}$ as the new estimate of the nondominant variables. The cycles form the outer iterations.

REFERENCES AND SUGGESTED READING

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes*. Cambridge, UK: Cambridge University Press.

L. N. Trefethen, and D. Bau III. *Numerical Linear Algebra*. Cambridge, UK: Cambridge University Press, 1997.

PROBLEMS

1. If your course involves exercises with a CFD code, study the manual to determine which methods are available for solution of linear and nonlinear systems of algebraic equations. Does the code use any of the algorithms discussed in this chapter—for example, iterative methods with over- and underrelaxation, linearization, sequential solution of coupled equations, or multigrid acceleration?

2. Develop a finite difference scheme of the second order for the equation

$$\frac{\partial}{\partial x}\left[a(x,y)\frac{\partial u}{\partial x}\right]+\frac{\partial}{\partial y}\left[b(x,y)\frac{\partial u}{\partial y}\right]=g(x,y).$$

Assume that a, b, and g are known functions and use the five-point difference molecule on a uniform Cartesian grid (see Figure 8.1). *Hint:* Use the discretization approach applied to a one-dimensional heat equation with variable conductivity in section 4.3.

3. Rewrite the scheme developed in Problem 2 in the matrix form. Develop the row equation and expressions for coefficients as in (8.7), (8.8).

4. Derive the expressions for submatrices $\mathbf{B}_j$, $\mathbf{D}_j$, $\mathbf{A}_j$ in the block-tridiagonal form (8.17) of the matrix equation for the five-point discretization (8.5) of the two-dimensional Poisson equation. Consider only the blocks corresponding to internal points of the computational domain.

5. Consider the two-dimensional Poisson equation (8.4) in a rectangular domain $0 \le x \le L_x$, $0 \le y \le L_y$. The boundary conditions are

$$\frac{\partial u}{\partial x}(0,y)=g_1(y),\ u(L_x,y)=g_2(y),\ \frac{\partial u}{\partial y}(x,0)=g_3(x),$$
$$u(x,L_y)=g_4(x).$$

The computational grid is uniform with steps $\Delta x = L_x/N_x$ and $\Delta y = L_y/N_y$. Develop the system of finite difference equations approximating the entire problem (PDE and boundary conditions) with the second order of accuracy. Write the scheme in the matrix form similar

to (8.7), (8.8). Take into account that the equations and expressions for coefficients are different for internal and boundary grid points.

6. Develop a Gauss-Seidel algorithm (similar to (8.35)) for the central difference scheme applied to the three-dimensional Poisson equation (8.12).

7. Consider the steady-state version of the governing equations for an incompressible flow and convection heat transfer (2.20), (2.27). Disregard the incompressibility condition (2.6) and pressure field p for the moment. Methods dealing with them will be discussed in Chapter 10. Can the sequential solution procedure be applied to the system? For each equation, determine the dominant variable and write the symbolic linearized systems similar to (8.56).

Programming Exercises

1. Implement the Jacobi, Gauss-Seidel, and Gauss-Seidel with over- and underrelaxation iterative schemes for the five-point discretization scheme of the two-dimensional Poisson equation. Conduct computations to verify the data presented in Table 8.1. Perform additional numerical experiments to determine the value of relaxation parameter ω optimal for this particular problem.
2. Implement the generalized Thomas algorithm for a block-tridiagonal matrix. Use one of the freely downloadable matrix inversion subroutines (e.g., a subroutine from the *Lapack* linear algebra package at the www.netlib.org repository). Apply the algorithm to solve the two-dimensional Poisson problem illustrated in Table 8.1 and Figure 8.5.

9

UNSTEADY PROBLEMS OF FLUID FLOWS AND HEAT TRANSFER

9.1 INTRODUCTION

In the next two chapters, we extend the methods of finite difference and finite volume solution to unsteady multidimensional flows and heat transfer. It would be impossible to cover the entire variety of existing schemes in a single book, especially if the book, like ours, is an introduction into the subject. A large number of schemes, many of them rather complex, would have to be described. Furthermore, there is a danger in embarking on a painstakingly detailed discussion of particular algorithms. CFD is a rapidly evolving field. An algorithm may become obsolete soon after the book is published. For all these reasons, the following two chapters will focus on common principles of widely used methods and on introduction of several time-honored techniques that have formed the basis of modern CFD.

The main attention will be given to calculation of incompressible flows. Chapter 10 is entirely devoted to this subject. The rest of this chapter contains a brief review of the issues arising in computations of unsteady multidimensional compressible flows and heat transfer. Description of several schemes is included for the sake of completeness and can be skipped with no loss of continuity. The reader interested in these areas is encouraged to study the chapter and proceed to the books listed at the end for a deeper and more detailed discussion.

9.2 COMPRESSIBLE FLOWS

9.2.1 Overview and General Comments

We have finally arrived at the point where we can discuss the methods used to solve actual *fluid flow equations*. The discussion starts with compressible flows in this section and continues with incompressible flows in the next chapter. The reasons for such a separation will be explained a little later.

The system of equations to be solved always includes the continuity (mass conservation) equation

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u_k)}{\partial x_k} = 0 \tag{9.1}$$

and the Navier-Stokes (momentum conservation) equations

$$\rho \frac{Du_i}{Dt} = \rho f_i - \frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu\left(\frac{\partial u_k}{\partial x_k}\right)\delta_{ij}\right], \tag{9.2}$$

where, as usual, we assume summation over repeating indices.

If the flow is compressible, the system should also contain the thermodynamic equation of state connecting pressure and density (e.g., the ideal gas equation (2.29)). If significant heat transfer occurs, the energy conservation equation in some form (see section 2.5) should be included.

We would like to stress that it is not always necessary to solve the full system of conservation equations. In many applications, the system can be replaced, with no significant loss of accuracy, by one of the asymptotic approximations (e.g., an incompressible flow, inviscid flow, or boundary layer). Moreover, use of an approximation is often a requirement. Only in this way the specific properties of the solution can be utilized for developing more accurate and efficient numerical algorithms. Significance of this statement will be illustrated in the next chapter, when we discuss the implications of incompressibility.

Mathematical Nature of the Navier-Stokes Equations: We know that the partial differential equations of the second order can be classified into three basic types: parabolic, hyperbolic, and elliptic equations. As we have already demonstrated on the example of model one-dimensional equations, the underlying physical phenomena and the methods of numerical solution are different for different types. The classification with all its

consequences remains valid for the full Navier-Stokes equations. The analysis is, however, complicated by the fact that the Navier-Stokes equations are three-dimensional, nonlinear, and of mixed type. We shall discuss this issue in simple terms rather than going into mathematical details.

First of all, the viscous dissipation terms render the Navier-Stokes equations parabolic. The terms have the differential form (for simplicity, we only consider the first part)

$$\frac{\partial \sigma_{i,j}}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right]$$

and the integral form

$$\int_S \left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right] n_j \, dS\,.$$

Leaving only these terms in the right-hand side of the momentum equations and keeping only the time derivative terms in the left-hand side, we obtain the truncated equations, which are clearly parabolic. This can be illustrated by taking a one-dimensional version with u_i depending on only one coordinate and time

$$\rho \frac{\partial u_i}{\partial t} = \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_i}{\partial x_j}\right)\right]$$

and applying the classification criterion introduced in section 3.2.1. Identification of the equations as parabolic is also in agreement with the diffusion nature of the process of viscous dissipation.

If we only keep the time derivative and convection terms such as

$$\rho \boldsymbol{V} \cdot \nabla u \quad \text{or} \quad \int_S \rho u \boldsymbol{V} \cdot \boldsymbol{n} \, dS\,,$$

the truncated system becomes hyperbolic. This can be shown rigorously using the theory of classification of systems of PDE, which is not considered in this book, but can be found in more specialized literature. We only mention that each of the truncated equations looks exactly as a nonlinear multidimensional version of the linear convection equation

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0. \tag{9.3}$$

We should expect wavelike solutions. The analogy extends to the physical nature of the phenomenon described by the convection terms, which is the transport or propagation of fields by the flow velocity.

We see that the general Navier-Stokes equations for compressible flows (9.1), (9.2) can be characterized as *hyperbolic-parabolic*. The classification, however, changes when we apply the asymptotic approximations. Let us consider the most prominent cases. If the effect of viscosity is negligible, and the momentum equations are the Euler equations (2.21), the unsteady system becomes purely *hyperbolic*. In the steady-state case, the nature of the Euler system depends on the Mach number (the ratio $M = v/a$ between the typical flow velocity and the speed of sound). The equations are *elliptic* for subsonic flow ($M < 1$) and *hyperbolic* for supersonic flows ($M > 1$). At last, if we retain the viscosity and assume that the fluid is incompressible, the unsteady Navier-Stokes system retains its hyperbolic-parabolic character, but also acquires *elliptic* features because of the new role of pressure, which is discussed in details in Chapter 10.

Features of Compressible Flows: A flow is considered compressible if the Mach number is sufficiently large, approximately larger than 0.3. The effect of compressibility may need to be taken into account in many areas, for example in high-speed aerodynamics, turbomachinery, and combustion. Led by industrial and military applications, the CFD analysis of compressible flows has long become an established discipline with its own specialized methods.

Several features of compressible flows have to be kept in mind, since they directly affect the choice of numerical approach. The first is, of course, the general hyperbolic character of the equations. The numerical schemes should be stable and accurate (without excessive numerical dissipation and dispersion) for solutions dominated by waves. Our experience of finding such schemes for the linear convection equation (see section 7.1) is evidently useful here.

The Reynolds number in compressible flows is usually very high. This often means that the solution domain can be divided into thin boundary layers, where viscosity, heat transfer, and turbulence are important, and the outer domain, where the flow can be approximately considered ideal (nonviscous and nonconductive) or even potential (with zero vorticity $\boldsymbol{\omega} = \nabla \times \boldsymbol{v}$). The logical, historically established, and still used approach is to combine the ideal flow equations in the outer domain with physically justified models of the boundary layer behavior.

In the flows or within the flow domains, where turbulence is present, we have to apply turbulence models, some of which are discussed in Chapter 11. The effects of heat transfer and temperature variability of fluid properties are, often, important in turbulent compressible flows. The energy conservation equation has to be solved.

Some compressible flows are supersonic ($M > 1$), which means existence of shock waves. This raises complex questions for CFD analysis. The shocks are thin and characterized by very sharp gradients of flow variables, strong dissipation, and generation of heat. Direct numerical resolution of these processes is difficult or impossible in practical applications. The CFD analysis is often only feasible if it relies on the idealized physical models, in which shocks are treated as infinitely thin surfaces of solution discontinuity. Presence of such discontinuities completely changes the landscape of available computational techniques. Many of them, in particular those based on the second-order central differences popular in other areas of CFD, cannot be used since they lead to strong numerical dispersion and spurious oscillations around the shock. To address this issue, specialized methods for flows with discontinuities have been developed.

All these and other issues are discussed in depth in the books dealing with the specific subject of CFD analysis of compressible flows. Several references are listed at the end of this chapter. Our discussion is limited to a few methods that illustrate common techniques and often encountered difficulties.

It is customary and convenient to present the compressible Navier-Stokes system in standardized vector form (see section 2.9)

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{A}}{\partial x} + \frac{\partial \boldsymbol{B}}{\partial y} + \frac{\partial \boldsymbol{C}}{\partial z} = 0. \tag{9.4}$$

For simplicity and without loss of generality, we omit body forces and internal heat sources. The formal representation (9.4) is convenient in the sense that it is valid for all compressible flows. Details of a specific flow are hidden in the expressions (2.42)–(2.43) for $\boldsymbol{U}$, $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$. A numerical scheme designed and analyzed for (9.4) can be applied to any flow, provided proper discretization of (2.42)–(2.43) have been developed. The form (9.4) expresses the hyperbolic character of the equations (compare with the model hyperbolic equation $u_t + cu_x = 0$, which can be rewritten as $u_t + v_x = 0$ with $v = cu$). We would like to stress that the seeming simplicity of (9.4) does not change the complex three-dimensional nonlinear character of the equations.

9.2.2 Explicit MacCormack Method

The finite difference methods designed for the model one-dimensional hyperbolic equations (see section 7.1) can be generalized to the case of compressible flow equations (9.4). Some of them are, in fact, quite efficient. An example of such a method is the explicit MacCormack scheme introduced in section 7.1.2. Every time step is split into two substeps:

$$\text{Predictor: } \boldsymbol{U}^*_{i,j,k} = \boldsymbol{U}^n_{i,j,k} - \frac{\Delta t}{\Delta x}\left(\boldsymbol{A}^n_{i+1,j,k} - \boldsymbol{A}^n_{i,j,k}\right) - \frac{\Delta t}{\Delta y}\left(\boldsymbol{B}^n_{i,j+1,k} - \boldsymbol{B}^n_{i,j,k}\right) - \frac{\Delta t}{\Delta z}\left(\boldsymbol{C}^n_{i,j,k+1} - \boldsymbol{C}^n_{i,j,k}\right), \tag{9.5}$$

$$\text{Corrector: } \boldsymbol{U}^{n+1}_{i,j,k} = \frac{1}{2}\left[\boldsymbol{U}^n_{i,j,k} + \boldsymbol{U}^*_{i,j,k} - \frac{\Delta t}{\Delta x}\left(\boldsymbol{A}^*_{i,j,k} - \boldsymbol{A}^*_{i-1,j,k}\right) - \frac{\Delta t}{\Delta y}\left(\boldsymbol{B}^*_{i,j,k} - \boldsymbol{B}^*_{i,j-1,k}\right) - \frac{\Delta t}{\Delta z}\left(\boldsymbol{C}^*_{i,j,k} - \boldsymbol{C}^*_{i,j,k-1}\right)\right]. \tag{9.6}$$

We assume that the computational grid is rectangular and uniform with grid steps Δx, Δy, and Δz. The scheme has the truncation error T.E. $= O\left[(\Delta x)^2, (\Delta y)^2, (\Delta z)^2, (\Delta t)^2\right]$.

To keep the second order of approximation, the partial derivatives that appear in the viscous and diffusion terms of $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ (see (2.42)–(2.43)) must be approximated with, at least, the same order. In the case of a simple one-step scheme, the task would be trivial. We would simply have to use finite difference formulas of appropriate order. For multistep schemes such as (9.5)–(9.6), the problem is more complex. The order of approximation of the entire scheme can be higher than the orders of approximation used at separate substeps. An example of such behavior is given next.

For the MacCormack method, the second order of approximation is conserved if the following rules are followed. For the x-derivatives in $\boldsymbol{A}$, we use one-sided differences in the direction opposite to the direction of the difference used at this particular substep for $\partial \boldsymbol{A}/\partial x$. Central differences are applied for the y- and z-derivatives. Similarly, one-sided differences are used for the y-derivatives in $\boldsymbol{B}$ and z-derivatives in $\boldsymbol{C}$. The directions of these differences are opposite to the directions used for $\partial \boldsymbol{B}/\partial y$ and $\partial \boldsymbol{C}/\partial z$, respectively. For the other derivatives in $\boldsymbol{B}$ and $\boldsymbol{C}$, central differences are applied. As an illustration, we discretize the third

component of $\boldsymbol{C}$ (see (2.43)):

$$C_3 = \rho vw - \sigma_{yz} = \rho vw - \mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right).$$

In the predictor step, this term is approximated as

$$C^n_{i,j,k} = (\rho vw)^n_{i,j,k} - \mu\left(\frac{v^n_{i,j,k} - v^n_{i,j,k-1}}{\Delta z} + \frac{w^n_{i,j+1,k} - w^n_{i,j-1,k}}{2\Delta y}\right),$$

while in the corrector step, the approximation is

$$C^{\overline{n+1}}_{i,j,k} = (\rho vw)^{\overline{n+1}}_{i,j,k} - \mu\left(\frac{v^{\overline{n+1}}_{i,j,k+1} - v^{\overline{n+1}}_{i,j,k}}{\Delta z} + \frac{w^{\overline{n+1}}_{i,j+1,k} - w^{\overline{n+1}}_{i,j-1,k}}{2\Delta y}\right).$$

As it is typical for schemes applied to full Navier-Stokes equations, a rigorous stability analysis of the MacCormack method is impossible. One can, however, use the empirical formula (see Tannehill et al. 1997)

$$\Delta t \le \frac{\sigma(\Delta t)_{CFL}}{1 + 2/Re_\Delta}. \tag{9.7}$$

Several important and interesting observations can be made in regard of (9.7). We will use them to illustrate the common problems arising in the time integration of the Navier-Stokes equations. First, the stability condition is usually inexact. As a result, the scheme can sometimes become unstable at Δt close to the stability limit even if the condition is formally satisfied. To deal with this problem and avoid the instability, the *safety factor* σ is introduced. Although (9.7) is supposed to guarantee a stable time step at $\sigma = 1$, it is common to use $\sigma \approx 0.9$ or smaller.

Second, the stability condition is a combination of the inviscid (Courant-Friedrichs-Lewy or CFL) limit and the viscous limit. This reflects the fact that the equations combine hyperbolic (convective) and parabolic (viscous) features. The CFL limit for the MacCormack scheme is

$$(\Delta t)_{CFL} = \left[\frac{|u|}{\Delta x} + \frac{|v|}{\Delta y} + \frac{|w|}{\Delta z} + a\left(\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} + \frac{1}{(\Delta z)^2}\right)^{1/2}\right]^{-1}, \tag{9.8}$$

where a is the local speed of sound. To account for the effect of viscosity on stability, we multiply the time step limit by the coefficient that contains the minimum mesh Reynolds number

$$Re_\Delta = \min(Re_{\Delta x}, Re_{\Delta y}, Re_{\Delta z}), \tag{9.9}$$

with

$$Re_{\Delta x} = \frac{\rho|u|\Delta x}{\mu},\ Re_{\Delta y} = \frac{\rho|v|\Delta y}{\mu},\ Re_{\Delta z} = \frac{\rho|w|\Delta z}{\mu}. \tag{9.10}$$

Third, as can be seen in (9.8) and (9.10), the stability limits are derived on the basis of the "freezing" assumption. The variable velocity components u, v, and w are used as if they were constants. This means that (9.7) provides a *local* stability criterion valid at a given space point and a given moment of time. In calculations, the time step that provides stability for the entire solution is evaluated using some estimates of u, v, and w. Alternatively, if the variable time-step approach is followed, the lower bound of (9.7) is calculated after each time step on the basis of the current velocity field and used to determine the stable $\Delta t^n = t^{n+1} - t^n$.

The last comment concerns the limited applicability of the MacCormack and many other explicit schemes to the aerodynamic flows at high Reynolds numbers. Such flows are characterized by thin boundary layers at solid walls, within which the flow has sharp gradients and, thus, the computational grid must be refined (see Chapter 12 for a discussion of refinement). The small grid steps would require small time step Δt. This can be easily seen. Let, for example, the requirements of the resolution of boundary layers be such that $\Delta x \ll \Delta y, \Delta z$. From (9.8) and (9.10) we have $(\Delta t)_{CFL} = O(\Delta x)$ and $Re_\Delta = O(\Delta x)$. This means (see (9.7)) that for stability we have to use the time step $\Delta t < \text{const}(\Delta x_{min})^2$. The result can be an unacceptably large number of required time steps. The problem is not unique and concerns other explicit methods. Having said that we should also mention that in CFD applications the turbulent boundary layers are often subject to physical modeling (see Chapter 11), and excessively fine grids are not required.

9.2.3 Beam-Warming Method

The stability limit on the time step can be avoided if we apply an implicit method. Such methods are usually unconditionally stable, so larger time steps can be used. The downside is, of course, the higher complexity of the schemes and larger amount of computations needed to complete

each step. This is especially true since the governing equations (9.4) are multidimensional and nonlinear (remember that the vector fields $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ in (2.42)–(2.43) are quadratic in terms of the flow variables). The high computational cost of a time step can make an implicit solution completely unfeasible, unless we find a way to linearize the equations and solve the resulting discretized system efficiently.

The historically first efficient implicit method was designed by Beam and Warming (1976). A family of schemes following the same principles was later developed. The common features of these schemes are that they are *noniterative* (do not require iterations to complete one time step), use a truncated Taylor expansion for *linearization*, and avoid dealing with nontridiagonal matrices that appear in multidimensional problems using *approximate factorization*.

We will give a brief description of the classical Beam-Warming scheme for the case of two-dimensional compressible ideal (non viscous) flow. The equations are written in the vector form:

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{A}}{\partial x} + \frac{\partial \boldsymbol{B}}{\partial y} = 0. \tag{9.11}$$

An extension to the three-dimensional case is straightforward. A more detailed discussion of the method can be found, for example, in Tannehill et al. (1997).

The first step is to apply the implicit Crank-Nicolson time integration, which results in the second order approximation:

$$\boldsymbol{U}^{n+1} = \boldsymbol{U}^n - \frac{\Delta t}{2}\left[\left(\frac{\partial \boldsymbol{A}}{\partial x} + \frac{\partial \boldsymbol{B}}{\partial y}\right)^n + \left(\frac{\partial \boldsymbol{A}}{\partial x} + \frac{\partial \boldsymbol{B}}{\partial y}\right)^{n+1}\right]. \tag{9.12}$$

The vectors $\boldsymbol{U}^n$, $\boldsymbol{U}^{n+1}$, $\boldsymbol{A}^n$, $\boldsymbol{B}^n$, $\boldsymbol{A}^{n+1}$, and $\boldsymbol{B}^{n+1}$ are the vector functions of x and y, which are obtained after the time discretization, but before we conduct the spatial one.

The terms in (9.12) that include $\boldsymbol{A}^{n+1}$ and $\boldsymbol{B}^{n+1}$ are nonlinear with respect to the unknown elements of $\boldsymbol{U}^{n+1}$. Following Beam and Warming, we approximate them by local linearizations in the form of Taylor approximations around the time level t^n:

$$\boldsymbol{A}^{n+1} \approx \boldsymbol{A}^n + \mathbf{F}^n \cdot \left(\boldsymbol{U}^{n+1} - \boldsymbol{U}^n\right) \tag{9.13}$$

$$\boldsymbol{B}^{n+1} \approx \boldsymbol{B}^n + \mathbf{G}^n \cdot \left(\boldsymbol{U}^{n+1} - \boldsymbol{U}^n\right), \tag{9.14}$$

where $\mathbf{F}^n$ and $\mathbf{G}^n$ are the Jacobian matrices

$$\mathbf{F}^n = \left(\frac{\partial \boldsymbol{A}}{\partial \boldsymbol{U}}\right)^n, \quad \mathbf{G}^n = \left(\frac{\partial \boldsymbol{B}}{\partial \boldsymbol{U}}\right)^n. \tag{9.15}$$

The error introduced by the linearization can be easily estimated if we consider that the reminders of the Taylor series omitted in (9.13) and (9.14) are

$$\sim \left(\boldsymbol{U}^{n+1} - \boldsymbol{U}^n\right)^2 \sim \left(\frac{\partial \boldsymbol{U}}{\partial t}\Delta t\right)^2 \sim (\Delta t)^2.$$

The additional error is, thus, of the second order in time.

Substituting the expansions (9.13)–(9.14) into (9.12) and rearranging the equation so that the terms with the unknown $\boldsymbol{U}^{n+1}$ are in the left-hand side and all the other terms are in the right-hand side, we obtain

$$\left[\mathbf{I} + \frac{\Delta t}{2}\left(\frac{\partial}{\partial x}\mathbf{F}^n + \frac{\partial}{\partial y}\mathbf{G}^n\right)\right] \cdot \boldsymbol{U}^{n+1} =$$
$$\left[\mathbf{I} + \frac{\Delta t}{2}\left(\frac{\partial}{\partial x}\mathbf{F}^n + \frac{\partial}{\partial y}\mathbf{G}^n\right)\right] \cdot \boldsymbol{U}^n - \Delta t\left(\frac{\partial \boldsymbol{A}}{\partial x} + \frac{\partial \boldsymbol{B}}{\partial y}\right)^n, \tag{9.16}$$

where $\mathbf{I}$ is the identity matrix (the matrix whose elements are zeros except for the elements on the main diagonal, which are all equal to one). The equation (9.16) is written in the operator form. The operator

$$\mathbf{I} + \frac{\Delta t}{2}\left(\frac{\partial}{\partial x}\mathbf{F}^n + \frac{\partial}{\partial y}\mathbf{G}^n\right)$$

stands for the combination of matrix multiplications and additions, and differentiations in x and y, such that the result of its action on a vector field, for example on $\boldsymbol{U}^{n+1}$, is

$$\left[\mathbf{I} + \frac{\Delta t}{2}\left(\frac{\partial}{\partial x}\mathbf{F}^n + \frac{\partial}{\partial y}\mathbf{G}^n\right)\right] \cdot \boldsymbol{U}^{n+1} =$$
$$\boldsymbol{U}^{n+1} + \frac{\Delta t}{2}\frac{\partial}{\partial x}\left(\mathbf{F}^n \cdot \boldsymbol{U}^{n+1}\right) + \frac{\Delta t}{2}\frac{\partial}{\partial y}\left(\mathbf{G}^n \cdot \boldsymbol{U}^{n+1}\right). \tag{9.17}$$

The equation (9.16) is linear for the unknown variables in $\boldsymbol{U}^{n+1}$. The spatial discretization applied at this stage would result in a system of linear algebraic equations (a matrix equation). Because of multidimensionality of the problem, the coefficient matrix of the system would be nontridiagonal.

This can be easily seen if we consider expressions for components of matrices $\mathbf{F}^n$ and $\mathbf{G}^n$ and approximate (9.17) by a second-order scheme on five-point difference molecule. The coefficients of the resulting system of discretization equations would form a matrix with a nontridiagonal structure similar to the structure shown in Figure 8.2.

The Thomas algorithm cannot be applied to a matrix equation with nontridiagonal matrix. One of the direct or, more efficiently, iterative methods discussed in Chapter 8 should be applied. Another efficient approach is to apply the approximation of the multidimensional operator by a sequence of one-dimensional operators. There exist several versions, including the *approximate factorization* method explained here.

The operator in the left-hand side of (9.17) is replaced (*factorized*) as

$$\left[\mathbf{I}+\frac{\Delta t}{2}\left(\frac{\partial}{\partial x}\mathbf{F}^n+\frac{\partial}{\partial y}\mathbf{G}^n\right)\right]\cdot \boldsymbol{U}^{n+1}\approx\left(\mathbf{I}+\frac{\Delta t}{2}\frac{\partial}{\partial x}\mathbf{F}^n\right)\cdot\left(\mathbf{I}+\frac{\Delta t}{2}\frac{\partial}{\partial y}\mathbf{G}^n\right)\cdot \boldsymbol{U}^{n+1}. \quad (9.18)$$

Direct matrix multiplication shows that the right-hand side and left-hand side differ by the term

$$\left(\frac{\Delta t}{2}\right)^2\left(\frac{\partial}{\partial x}\mathbf{F}^n\cdot\frac{\partial}{\partial y}\mathbf{G}^n\right)\cdot \boldsymbol{U}^{n+1},$$

which constitutes the error of the approximation. This error is $\sim(\Delta t)^2$, so it does not affect the second order of approximation set by the Crank-Nicolson scheme. After a similar factorization of the operator in the right-hand side, the equation (9.16) becomes

$$\left(\mathbf{I}+\frac{\Delta t}{2}\frac{\partial}{\partial x}\mathbf{F}^n\right)\left(\mathbf{I}+\frac{\Delta t}{2}\frac{\partial}{\partial y}\mathbf{G}^n\right)\cdot \boldsymbol{U}^{n+1}=\left(\mathbf{I}+\frac{\Delta t}{2}\frac{\partial}{\partial x}\mathbf{F}^n\right)\cdot\left(\mathbf{I}+\frac{\Delta t}{2}\frac{\partial}{\partial y}\mathbf{G}^n\right)\cdot \boldsymbol{U}^n-\Delta t\left(\frac{\partial \boldsymbol{A}}{\partial x}+\frac{\partial \boldsymbol{B}}{\partial y}\right)^n \quad (9.19)$$

A simpler and computationally more efficient version is obtained if we introduce the new vector field $\Delta \boldsymbol{U}=\boldsymbol{U}^{n+1}-\boldsymbol{U}^n$ and rewrite the equation as

$$\left(\mathbf{I}+\frac{\Delta t}{2}\frac{\partial}{\partial x}\mathbf{F}^n\right)\cdot\left(\mathbf{I}+\frac{\Delta t}{2}\frac{\partial}{\partial y}\mathbf{G}^n\right)\cdot\Delta \boldsymbol{U}=-\Delta t\left(\frac{\partial \boldsymbol{A}}{\partial x}+\frac{\partial \boldsymbol{B}}{\partial y}\right)^n \quad (9.20)$$

The advantage of the factorized forms (9.19) and (9.20) in comparison to the original form (9.16) is that now the left-hand side is a product of

two operators, which are one-dimensional in the sense that each involves a derivative with respect to only one coordinate. The time step can be arranged as a sequence of substeps, which only require solution of matrix equations with tridiagonal matrices if central differences of second order are applied for discretization. For example, the sequence for (9.20) is:

Step 1: Solve

$$\left(\mathbf{I} + \frac{\Delta t}{2}\frac{\partial}{\partial x}\mathbf{F}^n\right) \cdot \widetilde{\Delta \boldsymbol{U}} = -\Delta t \left(\frac{\partial \boldsymbol{A}}{\partial x} + \frac{\partial \boldsymbol{B}}{\partial y}\right)^n$$

Step 2: Solve

$$\left(\mathbf{I} + \frac{\Delta t}{2}\frac{\partial}{\partial y}\mathbf{G}^n\right) \cdot \Delta \boldsymbol{U} = \widetilde{\Delta \boldsymbol{U}}$$

Step 3: Update the solution $\boldsymbol{U}^{n+1} = \boldsymbol{U}^n + \Delta \boldsymbol{U}$.

The Beam-Warming algorithm presented here is unconditionally stable and of the second order in time. The order in space is determined by the scheme used for spatial discretization. Other versions of the method, some with higher order of time integration, have been developed. They are discussed, for example, in Tannehill et al. (1997).

9.2.4 Upwinding

We already saw an example of upwinding in section 7.1.1 when we solved the linear convection equation

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0, \quad c > 0. \tag{9.21}$$

It was found, perhaps surprisingly, that a simple explicit scheme is only usable if it is based on the backward difference for the space derivative:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_i^n - u_{i-1}^n}{\Delta x} = 0. \tag{9.22}$$

The other two schemes we analyzed, those based on forward and central differences, were found unconditionally unstable.

The results can be generalized to the rule valid for arbitrary sign of c. Considering the amplification factors (7.12)–(7.14), it is easy to see

that the only meaningful simple explicit scheme is the *upwind scheme*, in which the space derivative is approximated as

$$\left.\frac{\partial u}{\partial x}\right|_i^n \approx \begin{cases} \left(u_i^n - u_{i-1}^n\right)/\Delta x & \text{if} \quad c > 0 \\ \left(u_{i+1}^n - u_i^n\right)/\Delta x & \text{if} \quad c \le 0. \end{cases} \tag{9.23}$$

The same scheme is obtained if we follow the finite volume approach and apply the *upwind interpolation* of the first order (5.17) (see section 5.3.1). The upwind scheme is stable if the CFL condition $|\nu| = |c|\Delta t/\Delta x \le 1$ is satisfied. On the contrary, central and downwind (one-sided and opposite to upwind) approximations of $\partial u/\partial x$ create unconditionally unstable schemes.

The clear advantage of the upwind approximation of spatial derivatives (or, simply, *upwinding*) is associated with the physical properties of the solutions of (9.21). The equation is the simplest representative of hyperbolic equations, solutions of which are dominated by waves propagating along characteristics. In the case of (9.21), there is only one wave that propagates to the right if $c > 0$ and to the left if $c < 0$ (see Figure 9.1).

The direction of the wave motion is also the direction in which information propagates in the solution. At $c > 0$, the value of u at x_i is affected by the solution at x_{i-1}, but absolutely insensitive to the solution at x_{i+1}. At $c < 0$, the direction of wave motion and, thus, the roles of x_{i-1} and x_{i+1}, are reversed. The upwind scheme reflects this behavior, while the central and downwind schemes ignore it and introduce the artificial, physically impossible influence of the grid point, which is *downstream* (x_{i+1} at $c > 0$ or x_{i-1} at $c < 0$). As a result, the upwind scheme produces an acceptable approximation, while central and downwind schemes do not.

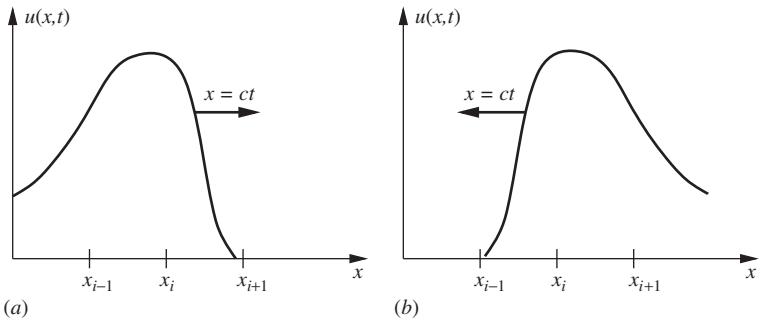


Figure 9.1 Wavelike solutions of the linear convection equation (9.21) at *(a)* $c > 0$ and *(b)* $c < 0$.

The idea of upwinding extends to many areas of CFD as a broad, nearly philosophical concept: If a solution has strong wavelike features propagating in certain directions, better stability and accuracy can be achieved through the use of upwind differentiation or, in the case of finite volume methods, upwind interpolation. The upwinding is understood here not as a particular approximation, such as (9.23), but as the general approach, according to which we design a difference or interpolation formula so that it either uses only the grid points on the upwind side or gives to these points larger weights than to the downwind points (the second strategy is also called *upwind-bias*). It is important to mention that first-order schemes, such as (9.23) have very strong numerical dissipation and low accuracy (see section 4.3.2). For this reason, the modern upwind or upwind-bias methods mostly use multipoint approximations of the second order or higher.

One important situation, in which the upwinding can be helpful, is the approximation of convection terms of the Navier-Stokes equations. The analogy between these terms and the linear convection equation (9.21) has already been discussed. For example, considering the x-momentum equation in conservation form, which starts as

$$\frac{\partial}{\partial t}(\rho u) + \frac{\partial}{\partial x}(u\rho u) + \frac{\partial}{\partial y}(v\rho u) + \frac{\partial}{\partial z}(w\rho u) = \ldots,$$

we see that the solution should contain certain wave-like structures propagated by the flow velocity with components u, v, and w.

Of course, the Navier-Stokes equations are not purely hyperbolic. The waves are affected by viscosity and pressure and do not have clearly defined domains of influence and dependence. In many cases, schemes with central difference approximation of convection terms produce good results. Still, there are cases in which the hyperbolic effects are quite strong. This happens, in particular, in the *convection-dominated* flows, in which the amplitude of the convection terms is much larger than the amplitude of viscous terms. The central difference formulas are known not to work in an optimal way in such flows. The situation becomes even worse when symmetric formulas of high order (fourth of higher) are applied. The main troubles are the low stability limit and spurious small-scale oscillations in the numerical solution. Adding a certain amount of upwinding usually eliminates or reduces these effects and makes the computational schemes more robust. There are various techniques of doing that.

Typically, an upwind-bias discretization or interpolation formula can be considered as a combination of weighted symmetric and purely upwind formulas.

9.2.5 Methods for Purely Hyperbolic Systems

The need for special treatment is more severe in the case of purely hyperbolic systems. In CFD, such systems are often associated with inviscid gas dynamics and with supersonic flows, although hyperbolic equations may describe processes in many other areas: acoustics, optics, electrodynamics, population balance dynamics, and so on. The difficulty of numerical analysis of purely hyperbolic systems is often increased by the presence of discontinuities in the solution. The shock waves in supersonic flows provide the best known, but not unique, example. A numerical scheme should be able to reproduce the shape and the motion of a shock correctly and to maintain its structure as a discontinuity.

Using inappropriate discretization schemes, for example, the schemes that involve downwind grid points can lead to serious troubles. In particular, spurious oscillations appear in the areas of strong gradients of the solution and around the discontinuities. As an example, consider a shock wave solution of the linear convection equation (9.21). The exact solution shown in Figure 9.2a is a step-wave moving without changing its shape to the right (we assume that $c > 0$). Approximation by a stable, but not purely upwind scheme—such as, for example, the Lax-Wendroff method (7.24)—would lead to the unphysical behavior illustrated in Figure 9.2b.

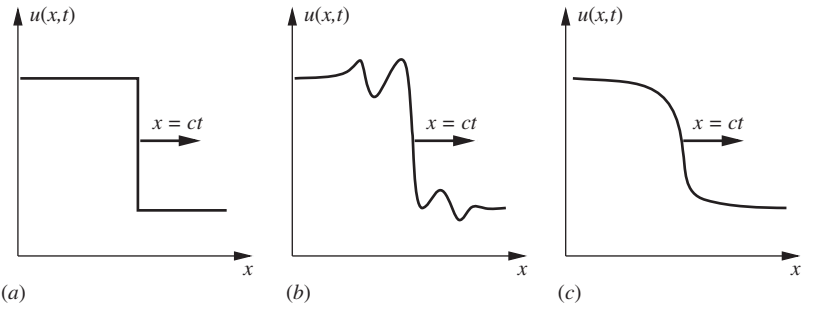


Figure 9.2 Discontinuous solution of the linear convection equation (9.21) at $c > 0$. *(a)* exact solution; *(b)* non-TVD numerical solution with spurious oscillations; *(c)* numerical solution with smearing due to the first order of approximation.

A useful property of a solution of a hyperbolic system is its *total variation*. In the one-dimensional case, the total variation is defined as

$$TV(u) \equiv \int_{\Omega} \left| \frac{\partial u}{\partial x} \right| dx \tag{9.24}$$

or, in terms of numerical solution,

$$TV(u) \equiv \sum_{i=1}^{N} |u_{i+1} - u_i|. \tag{9.25}$$

The integration and summation are over the entire solution domain and all grid points, respectively. In a solution consisting of propagating waves, the shape of u does not change, so the total variation $TV(u)$ remains unchanged in time. A good numerical scheme should, ideally, reproduce this property. In practice, we require that the scheme does not allow the total variation to grow. The schemes satisfying the condition

$$TV\left(u^{n+1}\right) \leq TV\left(u^{n}\right) \tag{9.26}$$

are called *total-variation-diminishing*, or TVD. The importance of the TVD requirement becomes clear if we consider the illustration in Figure 9.2b. The schemes that generate spurious oscillations around a shock increase the total variation. They are not TVD and would be banned if the TVD requirement were imposed.

Let us now focus on purely upwind TVD schemes. One example is the first-order scheme (9.23) for the linear convection equation. Other first order methods have been developed over the years to calculate solutions with discontinuities in simple and in multidimensional complex systems. They all, however, have a serious flaw. The first-order approximation means very strong numerical dissipation. The sharp-gradient features of the solution, including the shocks, are smeared out. This is illustrated in Figure 9.2c.

The undesirable smearing can be avoided if we use the schemes of second order of accuracy, which have much lower numerical dissipation. An obvious way to do so, while maintaining the upwind character, is to apply the second-order one-sided differences or interpolation schemes. For example, instead of (9.23) we can use

$$\left.\frac{\partial u}{\partial x}\right|_i^n \approx \begin{cases} \left(3u_i^n - 4u_{i-1}^n + u_{i-2}^n\right)/2\Delta x & \text{if} \quad c > 0 \\ \left(-3u_i^n + 4u_{i+1}^n - u_{i+2}^n\right)/2\Delta x & \text{if} \quad c \leq 0. \end{cases} \tag{9.27}$$

The problem of this approach is that it brings back the spurious oscillations around the discontinuities. One popular way to improve the situation is to modify the schemes so that they become TVD, thus preventing the oscillations. The modifiers (also called limiters) are specifically design so as to limit the possible variations of the solution. The second order of approximation can be lost near a discontinuity, but maintained in the areas where the solution is smooth. The approach is very effective. It is, in fact, so effective that it allows us to avoid oscillations in nonupwind schemes—such as, for example, the Beam-Warming or the Lax-Wendroff method. A detailed discussion of these *high-resolution schemes* for hyperbolic systems goes far beyond the modest scope of our text, but can be found elsewhere—for example, in Leveque 2002.

9.3 UNSTEADY CONDUCTION HEAT TRANSFER

We solve the equation

$$\frac{\partial u}{\partial t} = a^2 \nabla^2 u, \tag{9.28}$$

which describes the process of diffusion of field $u(\boldsymbol{x}, t)$ with constant diffusivity a^2. The physical nature of the diffusion can vary, although the most common applications are the conduction heat transfer and molecular diffusion of an admixture in a motionless medium.

There are various techniques, numerical and analytical, to solve the equation. The finite element method is, in particular, efficient, versatile, and popular. Many widely used engineering finite element tools have the heat transfer capability, which provides a relatively simple and straightforward solution. The more academic Green's function approach can also be mentioned. We focus exclusively on the finite difference technique. A reader interested in other methods is urged to consult more comprehensive or more specialized books. Some references are provided at the end of the chapter.

We have already considered finite difference schemes for (9.28), but only in the special cases of a steady-state problem (in Chapter 8) or one-dimensional unsteady problem (in Chapter 7). Here, we address the general unsteady-state multidimensional form of the equation. For simplicity, the schemes are presented on the example of the two-dimensional equation discretized on a uniform Cartesian grid with steps Δx and Δy.

9.3.1 Simple Methods for Multidimensional Heat Conduction

The schemes derived earlier for the one-dimensional heat equation can be easily generalized to the two-dimensional case. The five-point central difference approximation can be used for spatial derivatives. The simple explicit, implicit, and Crank-Nicolson methods become

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} = a^2 \left[\frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{(\Delta x)^2} + \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{(\Delta y)^2} \right], \tag{9.29}$$

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} = a^2 \left[\frac{u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}}{(\Delta x)^2} + \frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{(\Delta y)^2} \right], \tag{9.30}$$

and

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} = \frac{a^2}{2} \left[\frac{u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}}{(\Delta x)^2} + \frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{(\Delta x)^2} + \frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{(\Delta y)^2} + \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{(\Delta y)^2} \right]. \tag{9.31}$$

The truncation error of these schemes is of the same order as the error of the one-dimensional version: $O(\Delta t, (\Delta x)^2, (\Delta y)^2)$ for the simple explicit and implicit methods and $O((\Delta t)^2, (\Delta x)^2, (\Delta y)^2)$ for the Crank-Nicolson method.

Generalization of stability results obtained for one-dimensional equations to the multidimensional case must be done with caution. Implicit schemes (9.30) and (9.31) remain unconditionally stable. For the explicit scheme (9.29), the Fourier analysis gives the stability condition

$$a^2 \Delta t \left(\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} \right) \leq \frac{1}{2}, \tag{9.32}$$

which is more restrictive than the condition $r \leq 1/2$ for the one-dimensional equation derived in section 7.2.1. For example, if we take $\Delta x = \Delta y$, it is easy to mistakenly assume that

$$r = \frac{a^2 \Delta t}{(\Delta x)^2} = \frac{a^2 \Delta t}{(\Delta y)^2} \leq \frac{1}{2}$$

is sufficient in the two-dimensional case. In fact, according to (9.32), we must use $r \leq 1/4$. Similarly, if the simple explicit method is applied to the three-dimensional heat equation, the stability condition changes to

$$a^2 \Delta t \left(\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} + \frac{1}{(\Delta z)^2} \right) \leq \frac{1}{2}. \tag{9.33}$$

9.3.2 Approximate Factorization

The stability requirements of explicit methods, such as (9.32) and (9.33), often impose severe limitations on the size of the time step Δt. This typically happens because at least one of the grid steps Δx, Δy, Δz should be small, at least in some areas, to resolve strong variations of the solution. The implicit schemes, such as (9.30) and (9.31), do not have this disadvantage, since they are unconditionally stable and allow us to take arbitrarily large time steps. In the multidimensional case, however, the implicit schemes present the difficulty of having to solve a matrix equation with nontridiagonal matrix. The simple and effective Thomas algorithm cannot be used.

We have already encountered this situation when we discussed the Beam-Warming scheme applied to compressible flows. We learned that, in addition to iteration methods, there is an efficient approach based on approximate factorization. Every implicit step is split into two or more substeps, each requiring solution of a tridiagonal system.

We will show how the approximate factorization method is applied to the heat equation. The basic technique is the same as in the Beam-Warming method, but the realization is simpler and can be presented in terms of discretization equations. We will work with the increments of solution variables at grid points

$$\Delta u_{i,j} = u_{i,j}^{n+1} - u_{i,j}^{n}. \tag{9.34}$$

Similar increments were used by the Beam-Warming scheme, although we introduced them close to the end of the derivation.

The finite difference operators approximating the second derivatives of u are:

$$L_x u_{i,j}^n = \frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{(\Delta x)^2}, \quad L_y u_{i,j}^n = \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{(\Delta y)^2}. \tag{9.35}$$

The scheme can be developed on the basis of an arbitrarily weighted implicit-explicit scheme. As in our earlier discussion of the Beam-Warming method, we will use the Crank-Nicolson scheme. The finite difference formula (9.31) can be rewritten as

$$\frac{1}{\Delta t}\Delta u_{i,j} = \frac{a^2}{2}\left(L_x u_{i,j}^{n+1} + L_x u_{i,j}^{n} + L_y u_{i,j}^{n+1} + L_y u_{i,j}^{n}\right). \tag{9.36}$$

Replacing $u_{i,j}^{n+1}$ by $u_{i,j}^{n} + \Delta u_{i,j}$ we obtain

$$\frac{1}{\Delta t}\Delta u_{i,j} = \frac{a^2}{2}\left(L_x \Delta u_{i,j} + L_y \Delta u_{i,j} + 2L_x u_{i,j}^{n} + 2L_y u_{i,j}^{n}\right). \tag{9.37}$$

Moving the terms with the unknowns $\Delta u_{i,j}$ into the left-hand side and all the other terms into the right-hand side produces

$$\left(1 - \frac{a^2\Delta t}{2}L_x - \frac{a^2\Delta t}{2}L_y\right)\Delta u_{i,j} = a^2\Delta t\left(L_x + L_y\right)u_{i,j}^{n}. \tag{9.38}$$

The system of such equations can be written as a matrix equation with nontridiagonal matrix and is as costly to solve as the system associated with the traditional Crank-Nicolson scheme. To be able to use the Thomas algorithm, the procedure of approximate factorization is applied. Similarly to (9.18), we approximate the left-hand side as

$$\left(1 - \frac{a^2\Delta t}{2}L_x - \frac{a^2\Delta t}{2}L_y\right)\Delta u_{i,j} \approx \left(1 - \frac{a^2\Delta t}{2}L_x\right)\left(1 - \frac{a^2\Delta t}{2}L_y\right)\Delta u_{i,j}. \tag{9.39}$$

The approximation introduces the error $(1/4)a^4(\Delta t)^2 L_x L_y \Delta u_{i,j}$, which is of the second order in Δt and can, therefore, be tolerated.

Algorithmically, the factorized operator is a consecutive application of the central difference operators (9.35) in the x- and y-directions. We introduce the intermediate increment

$$\widetilde{\Delta u}_{i,j} = \left(1 - \frac{a^2\Delta t}{2}L_y\right)\Delta u_{i,j}, \tag{9.40}$$

which can be found as a solution of

$$\left(1 - \frac{a^2\Delta t}{2}L_x\right)\widetilde{\Delta u}_{i,j} = a^2\Delta t\left(L_x + L_y\right)u_{i,j}^{n}. \tag{9.41}$$

The procedure can be summarized as follows: Solve, using Thomas algorithm, (9.41) for $\widetilde{\Delta u}_{i,j}$ and then (9.40) for Δu_{ij}, use the calculated $\Delta u_{i,j}$ to find $u_{i,j}^{n+1}$ according to (9.34). The scheme is of the second order in time and space and unconditionally stable. These properties are retained by the three-dimensional version, which requires factorization into three substeps.

9.3.3 ADI Method

The approximate factorization concept was, in fact, originally developed for the transient multidimensional heat equation. The first scheme was published more than 50 years ago by Peaceman and Rachford (1955) under the name of ADI (alternating directions implicit) scheme. We will briefly describe its simplest version.

In the ADI method, the solution for the implicit values $u_{i,j}^{n+1}$ is split into two successive substeps. On the first, an intermediate solution is found using the scheme, which is implicit for the x-derivative term and explicit for the y-derivative term:

$$\frac{\tilde{u}_{i,j} - u_{i,j}^{n}}{\Delta t/2} = a^2 \left[\frac{\tilde{u}_{i+1,j} - \tilde{u}_{i,j} + \tilde{u}_{i-1,j}}{(\Delta x)^2} + \frac{u_{i,j+1}^{n} - u_{i,j}^{n} + u_{i,j-1}^{n}}{(\Delta y)^2} \right]. \quad (9.42)$$

On the second substep, the solution is updated using the scheme, which is implicit for the y-term and explicit for the x-term:

$$\frac{u_{i,j}^{n+1} - \tilde{u}_{i,j}}{\Delta t/2} = a^2 \left[\frac{\tilde{u}_{i+1,j} - \tilde{u}_{i,j} + \tilde{u}_{i-1,j}}{(\Delta x)^2} + \frac{u_{i,j+1}^{n+1} - u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{(\Delta y)^2} \right]. \quad (9.43)$$

Rearranging the equations so that the quantities unknown at each particular substep are in the left-hand side and the known quantities are in the right-hand side, we obtain two systems, which are expressed in terms of one-dimensional differential operators as

$$\left(1 - \frac{a^2 \Delta t}{2} L_x\right) \tilde{u}_{i,j} = \left(1 + \frac{a^2 \Delta t}{2} L_y\right) u_{i,j}^{n}, \quad (9.44)$$

$$\left(1 - \frac{a^2 \Delta t}{2} L_y\right) u_{i,j}^{n+1} = \left(1 + \frac{a^2 \Delta t}{2} L_x\right) \tilde{u}_{i,j}. \quad (9.45)$$

Clearly, the system (9.44) can be separated into N_y independent subsystems, one for every j. Each such subsystem has a tridiagonal coefficient matrix. Similarly, the system (9.45) consists of N_x subsystems with tridiagonal matrices, one for every i. The subsystems can be solved efficiently using the Thomas algorithm.

The ADI method has the truncation error $O\left[(\Delta t)^2, (\Delta x)^2, (\Delta y)^2\right]$. The scheme is unconditionally stable when applied to a two-dimensional problem. Its extension to three dimensions, which would include three substeps, each implicit in one direction, retains the second-order accuracy and efficiency but becomes only conditionally stable. It requires that r_x, r_y, and $r_z = a^2 \Delta t/(\Delta z)^2$ are all smaller than 3/2.

An important aspect of the approximate factorization methods is that the intermediate solution, such as $\tilde{u}_{i,j}$, should be viewed as a preliminary approximation of $u_{i,j}^{n+1}$. Generally, it *does not* represent the solution at some intermediate time level, say at $t^{n+1/2} = t_n + \Delta t/2$. This has a significant implication that the intermediate solution does not have to satisfy the physical boundary conditions imposed on the solution $u(x, y, t)$ of the PDE problem. Moreover, imposing such conditions may introduce additional truncation error and compromise the accuracy of the entire scheme. For example, imposing physical boundary conditions on $\tilde{u}_{i,j}$ in the ADI method may result in a truncation error $\sim O(\Delta t)$.

The loss of accuracy can be avoided if special numerical boundary conditions are designed from the factorized equations themselves. As an example, to obtain the necessary conditions on $\tilde{u}$ in the ADI method, we should subtract (9.43) from (9.42) and evaluate the resulting equation at boundary point, while substituting the physical boundary conditions for u^n and u^{n+1}.

As a last comment, the approximate factorization methods can also be used to solve the steady-state elliptic problems. The application is based on the pseudo-transient approach. The elliptic problem is converted into a parabolic problem through a fictitious time-derivative term (see section 8.3.7).

REFERENCES AND SUGGESTED READING

Beck, J. V., K. D. Cole, A. Haji-Sheikh, and B. Litkouhi. *Heat Conduction Using Green's Functions*. Hemisphere, 1992.

Fletcher, C. A. J. *Computational Techniques for Fluid Dynamics, Vol. 2: Specific Techniques for Different Flow Categories*. Berlin: Springer-Verlag, 1991.

Jaluria, Y., and K. E. Torrance, *Computational Heat Transfer*. London: Taylor & Francis, 2003.

Leveque, R. J. *Finite Volume Methods for Hyperbolic Problems*. Cambridge, UK: Cambridge, 2002.

Samarskii, A. A., and P. N. Vabischhevich *Computational Heat Transfer, Vol. I and II*, New York: Wiley, 1995.

Tannehill, J. C., D. A. Anderson, and R. H. Pletcher. *Computational Fluid Mechanics and Heat Transfer*. Phyladelphia: Taylor & Francis, 1997.

Beam, R. M., and R. F. Warming, "An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation Law Form," *J. Comp. Phys*. 22 (1976): 87–110.

Peaceman, D., and M. Rachford. "The numerical solution of parabolic and elliptic differential equations," *J. SIAM* 3 (1955): 28–41.

PROBLEMS

1. For a two-dimensional compressible flow of a Newtonian viscous fluid with all variables depending on (x, y, t), the equations are written in vector form as

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{A}}{\partial x} + \frac{\partial \boldsymbol{B}}{\partial y} = 0.$$

Rewrite the expressions (2.42), (2.43) of the vector fields $\boldsymbol{U}$, $\boldsymbol{A}$, and $\boldsymbol{B}$ for this case.

2. Can the incompressible flow equations be written in the vector form (9.4)? What are the expressions for the vector fields $\boldsymbol{U}$, $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ in this case?

3. For the two-dimensional compressible flow equations in Problem 1, write the complete MacCormack scheme of second order in space and time. Follow the rules described in section 9.2.2 to derive the discretization of internal derivatives in $\boldsymbol{A}$ and $\boldsymbol{B}$.

4. A three-dimensional flow of air is modeled using the MacCormack scheme. The flow velocity is estimated to be between 0 and 200 m/s. The computational grid has $\Delta x = \Delta y = 10^{-2}$ m and variable step in the z-direction $10^{-3} \leq \Delta z \leq 10^{-2}$ m. Taking the air properties as $\rho = 1.2$ kg/m^3, $\mu = 1.81 \times 10^{-5}$ kg/m s, and $a = 340$ m/s (approximately the properties at 20° C), find the maximum time step that guarantees numerical stability of solution.

5. Considering that the Beam-Warming scheme is unconditionally stable, would it be justified to use very large time steps?

6. For the Beam-Warming scheme, what is the order of error introduced by linearization and by approximate factorization? Does the accuracy of the scheme deteriorate in the result of using these approximations?

7. A rectangular bar of dimensions $L_x \times L_y \times L_z$, constant material properties κ, ρ, C, and initial temperature T_0 is immersed in cold water maintained at constant temperature $T_w < T_0$.

a) Write the complete PDE problem (heat equation, initial and boundary conditions) for conduction heat transfer within the bar. For the boundary conditions, use Newton's cooling law (2.52) with constant heat transfer coefficient h.

b) Develop the simple explicit and Crank-Nicolson schemes. Include the finite difference approximations of the boundary conditions.

c) What are the truncation error and stability conditions for each scheme?

8. Rewrite the finite difference approximations developed in Problem 7 for the case when the material properties κ, ρ, C are functions of temperature.

9. Rewrite the approximate factorization method of section 9.3.2 for the case of three-dimensional heat conduction equation.

10. Compare the properties of the approximate factorization method, Crank-Nicolson method (9.31), and simple explicit method (9.29), all applied to solution of two-dimensional heat conduction problems. Discuss relative advantages and disadvantages of each method.

11. Show that the ADI scheme for two-dimensional heat equation can be obtained from the Crank-Nicolson scheme (9.31) by approximate factorization. Find the factorization error.

12. The two-dimensional heat equation is solved in a rectangular domain $0 < x < A$ and $0 < y < B$ with Dirichlet boundary conditions $u(x = 0, y, t) = f(y, t)$, $u(x = A, y, t) = g(y, t)$, $u(x, y = 0, t) = h(x, t)$, $u(x, y = B, t) = p(x, t)$. The ADI scheme is applied. Derive the numerical boundary conditions for the intermediate solution $\tilde{u}$. See the text for a hint on how this can be done.

Programming Exercises

1. Implement the simple explicit algorithm for two-dimensional heat equation and apply it to solve unsteady heat conduction in a rectangular plate of dimensions $L_x \times L_y = 0.1 \times 0.1$ m. The initial temperature is uniform and equal to 293 K. The boundary conditions are constant temperature of 293 K at $x = L_x$ and $y = 0$, perfectly insulated boundary at $x = 0$, and constant heat flux of $1000\,\mathrm{W/m^2}$ into the plate at $y = L_y$. Use the material properties of pure aluminum ($\rho \approx 2.7$ kg/m^3, $C \approx 900$ J/kg·K, $\kappa \approx 204$ W/m·K). Conduct the solution until the asymptotic steady state is found. Use uniform grids of 10×10, 50×50, and 100×100 points. Compare the results.

10

INCOMPRESSIBLE FLOWS

10.1 GENERAL CONSIDERATIONS

10.1.1 Introduction

No fluid can be considered perfectly incompressible. Even in the most carefully controlled situations, minute variations of density are present because of inevitable variations of temperature and for other reasons. The incompressible fluid model is, therefore, only an approximation. However, the approximation is quite accurate for flows, in which the local velocity is much smaller than the local speed of sound.

In this section, we consider methods developed specifically for the equations that describe flows in the approximation of incompressible fluid. In addition to constant density, we assume that the fluid properties, such as viscosity or heat conductivity, are constant.

The governing Navier-Stokes and continuity equations simplify to

$$\rho \frac{D\boldsymbol{V}}{Dt} = -\nabla p + \mu \nabla^2 \boldsymbol{V} + \rho \boldsymbol{f} \tag{10.1}$$

$$\nabla \cdot \boldsymbol{V} = 0, \tag{10.2}$$

where $\boldsymbol{f}$ is an external body force.

$$\frac{D\boldsymbol{V}}{Dt} \equiv \frac{\partial \boldsymbol{V}}{\partial t} + \frac{\partial (u\boldsymbol{V})}{\partial x} + \frac{\partial (v\boldsymbol{V})}{\partial y} + \frac{\partial (w\boldsymbol{V})}{\partial z} \equiv \frac{\partial \boldsymbol{V}}{\partial t} + \boldsymbol{N}(\boldsymbol{V}, \boldsymbol{V}) \tag{10.3}$$

is the material derivative in conservation form, with $\boldsymbol{N}$ being the shorthand notation for the nonlinear term. The energy equation can be reduced to the equation of convection heat transfer with, possibly, a heat source due to viscous dissipation and other effects:

$$\rho C \frac{DT}{Dt} = \kappa \nabla^2 T + \dot{Q}. \tag{10.4}$$

In the incompressible fluid model, the heat equation is decoupled from the momentum and incompressibility equations and can be solved separately.[1] We will focus on schemes designed to compute fluid flows—that is, to solve (10.1)–(10.2).

10.1.2 Role of Pressure

The incompressible flow model has certain distinctive mathematical and physical properties that require special computational techniques. The most important is the new role adopted by the pressure field. In compressible flows, pressure is defined by the equation of state as a function of other variables (e.g., density and temperature), which, in turn, are found as solutions of evolutionary equations. When a flow is incompressible, the situation is entirely different. There is no equation of state, except for $\rho = const$. The pressure, which still remains a function of space and time and has to be computed at every time step, is now determined as a part of the general solution of the momentum and incompressibility equations (10.1), (10.2).

The main difficulty arises here. We cannot find the pressure field by simple advancement of a marching scheme or by evaluating a function of other thermodynamic variables. Furthermore, the mass conservation condition (10.2) is not an independent evolutionary equation for density, as in the case of compressible flows, but a kinematic constraint on velocity. How can we find a velocity field, which follows the evolution prescribed by the momentum equation (10.1) and satisfies the incompressibility condition (10.2) at the same time?

The answer to these questions is to construct the pressure field in such a way that its gradient in (10.1) enforces the incompressibility. Let us

[1] An exception is the Boussinesq model of natural thermal convection, in which the incompressibility condition is relaxed to allow for buoyancy forces caused by temperature variations. Evidently, the heat and momentum equations are coupled in that case. We mention the Boussinesq model here because, apart from the buoyancy force, the momentum and incompressibility equations are the same as in the purely incompressible flow. They can be solved by the numerical methods described in this chapter.

see how this can be done. We apply the divergence operator $(\nabla\cdot)$ to the momentum equation. Requiring that $\nabla \cdot \boldsymbol{V} = 0$ and taking into account that the operator $(\nabla\cdot)$ commutes with $\partial/\partial t$ in the time-derivative term and with ∇^2 in the viscous term we find that the pressure field satisfies the Poisson equation

$$\nabla \cdot (\nabla p) \equiv \nabla^2 p = \rho \nabla \cdot [\boldsymbol{f} - \boldsymbol{N}(\boldsymbol{V}, \boldsymbol{V})] = \rho \nabla \cdot \boldsymbol{F}, \tag{10.5}$$

where we used the notation $\boldsymbol{F} = \boldsymbol{f} - \boldsymbol{N}(\boldsymbol{V}, \boldsymbol{V})$.

From the mathematical viewpoint, this means that the incompressible flow equations have some features of an elliptic system. We can say that the equations are of the mixed hyperbolic (convective terms), parabolic (viscous terms), and elliptic (pressure and incompressibility) type. The elliptic nature of the pressure solution has a physical meaning. It shows that, in an incompressible flow, the pressure field in the entire flow domain adjusts instantaneously to any, however localized, perturbation. This is in perfect agreement with the fact that weak perturbations, for example sound waves, propagate at infinite speed in incompressible fluids.

The special features of incompressible flows require special computational tools. We cannot dismiss the peculiar role of pressure as a mere technicality and apply the methods developed for compressible flows. An attempt to do so is likely to lead to disappointing results. For example, let us assume that we could adapt the MacCormack method (9.5)–(9.6) to the case of an incompressible flow. The CFL stability criterion (9.8) would require zero time step Δt since the speed of sound a is infinite. In fact, many established explicit schemes for compressible flows fail when compressibility is weak (the Mach number M is much smaller than 1). The required time step becomes too small when $M \to 0$.

The pressure equation poses new questions that have to be answered by any scheme designed for incompressible flows. Some of them are of familiar kind and refer to discretization of (10.5). The others are quite unique. They concern the organization of steps of a time-marching or iteration procedure. Every such step should combine solutions of two equations: the momentum equation (10.1) and the pressure equation (10.5). As we show in this Chapter, the task is nontrivial.

10.2 DISCRETIZATION APPROACH

We start with general comments concerning the discretization approach. Finite difference approximations of the momentum and pressure equations can be obtained using the formulas derived earlier for model equations. For

example, in the momentum equations, we can use central or upwind-bias schemes for convective terms and central differences for viscous terms. The pressure gradient term in (10.1) can be approximated by various schemes, not necessarily coinciding with the schemes used for the other terms and not necessarily on the same set of grid points (we will discuss this later).

In finite volume methods, the surface integrals corresponding to convective and viscous terms of the momentum equation are interpolated as convective and diffusive flux integrals, respectively (see section 5.2.2). The pressure gradient in the momentum equation is usually treated as a surface force. For example, for the x-momentum equation, we use

$$\int_{\Omega} \frac{\partial p}{\partial x} d\Omega = \int_{\partial\Omega} p \boldsymbol{e}_x \cdot \boldsymbol{n} dS = \int_{\partial\Omega} p n_x dS \tag{10.6}$$

and apply one of interpolation schemes to approximate the surface integral.

The pressure equation in the integral form is

$$\int_{\Omega} \nabla \cdot (\nabla p) d\Omega = \int_{\Omega} \nabla \cdot \boldsymbol{F} d\Omega. \tag{10.7}$$

The volume integrals are transformed into surface integrals using the divergence theorem:

$$\int_{\partial\Omega} \nabla p \cdot \boldsymbol{n} dS = \int_{\partial\Omega} \boldsymbol{F} \cdot \boldsymbol{n} dS \tag{10.8}$$

and approximated by interpolation formulas.

One important aspect of the system should be taken into account if a numerical scheme with exact conservation of mass is desired. The discretization schemes for the Laplace operator in (10.5), the pressure gradient term in the momentum equation (10.1), and the divergence operator in the right-hand side of (10.5) cannot be chosen independently of each other. Mathematically, the Laplace operator and the right-hand side of (10.5) are the results of application of the divergence operator to the pressure gradient and to the rest of the momentum equation (10.1). If the scheme resulting from the sequential application of the schemes used to discretize ∇p and $\nabla \cdot \boldsymbol{F}$ is equivalent to the scheme used for $\nabla^2 p$, the computed velocity field is exactly (up to the computer round-off error) incompressible: $\nabla \cdot \boldsymbol{V} = 0$. If, however, the equivalency is absent, the divergence of the computed velocity field deviates from zero. The error has the order of magnitude of the truncation error of the scheme. The effect is as if a source (positive or negative) of mass were created by the numerical scheme. However small, this artificial mass source is, in many cases, undesirable, since its effect may accumulate with time and lead

to significant mass imbalance and to poor conservation of kinetic energy by the solution. There is an additional rather unpleasant effect. In many cases, the errors of the energy balance are not strictly dissipative, which results in numerical instability.

10.2.1 Colocated and Staggered Grids

When designing a finite difference or finite volume scheme, we have to choose whether to use the same or different sets of grid points for velocity and pressure. The obvious choice seems to have a single set of points, at which all the variables and all the equations are discretized. Such a grid has the name of a *colocated or regular grid*. Albeit simple and easy in operation, the colocated grids were out of favor for long time because of their tendency to generate spurious oscillations in the solution.

Let us understand the origins of the oscillations. The phenomenon itself is quite general. It appears for colocated variable arrangements on two- and three-dimensional, structured and unstructured, uniform and nonuniform grids. For simplicity, we illustrate it on the example of a finite volume grid with uniform rectangular cells shown in Figure 10.1. The grid points are the cell midpoints marked by capital letters: P, E, N, etc. The same points can be considered as forming a rectangular finite difference grid. It will be clear from the following discussion that the discretized equations resulting from the two approaches are equivalent.

We start by approximating the divergence operator. The finite volume approach gives, by the divergence theorem,

$$\int_{\Omega} \nabla \cdot \boldsymbol{V}\, d\Omega = \int_{\partial\Omega} \boldsymbol{V} \cdot \boldsymbol{n}\, dS. \tag{10.9}$$

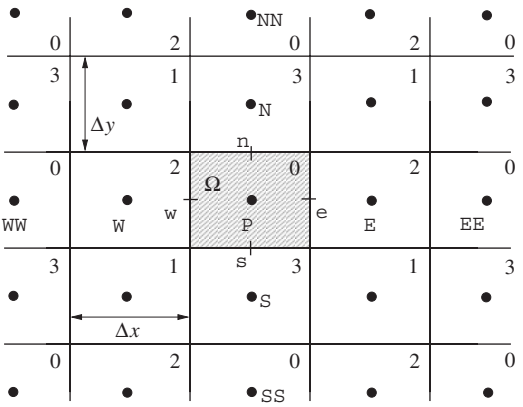


Figure 10.1 Finite volume colocated grid.

On each of the faces marked by e, w, s, and n, the outward-facing normal $\boldsymbol{n}$ is given by a positive or negative unit vector in the x- or y-direction. Using the mean value theorem, we represent the surface integral in (10.9) using the Cartesian velocity components u and v as

$$\int_{\partial\Omega} \boldsymbol{V} \cdot \boldsymbol{n}\, dS \approx u_{\mathrm{e}}\Delta y - u_{\mathrm{w}}\Delta y + v_{\mathrm{n}}\Delta x - v_{\mathrm{s}}\Delta x. \tag{10.10}$$

Dividing by the cell volume $\Delta x\,\Delta y$, we obtain the approximation of the second order

$$\frac{1}{\Delta x\,\Delta y}\int_{\Omega} \nabla\cdot\boldsymbol{V}\, d\Omega \approx \frac{u_{\mathrm{e}} - u_{\mathrm{w}}}{\Delta x} + \frac{v_{\mathrm{n}} - v_{\mathrm{s}}}{\Delta y} = \left.\frac{\delta u}{\delta x}\right|_{\mathrm{P}} + \left.\frac{\delta v}{\delta y}\right|_{\mathrm{P}}, \tag{10.11}$$

where we used the symbolic expressions $\delta u/\delta x$ and $\delta v/\delta y$ to denote discretizations of partial derivatives $\partial u/\partial x$ and $\partial v/\partial y$. The velocities u and v are only defined at the grid points. Their values at the face points have to be obtained by interpolation. Willing to retain the second order of accuracy, we use the linear (CD) interpolation:

$$u_{\mathrm{e}} \approx \frac{u_{\mathrm{E}} + u_{\mathrm{P}}}{2},\ u_{\mathrm{w}} \approx \frac{u_{\mathrm{W}} + u_{\mathrm{P}}}{2},\ v_{\mathrm{n}} \approx \frac{v_{\mathrm{N}} + v_{\mathrm{P}}}{2},\ v_{\mathrm{s}} \approx \frac{v_{\mathrm{S}} + v_{\mathrm{P}}}{2}. \tag{10.12}$$

Substitution into (10.11) gives the familiar formula

$$\left.\frac{\delta u}{\delta x}\right|_{\mathrm{P}} + \left.\frac{\delta v}{\delta y}\right|_{\mathrm{P}} = \frac{u_{\mathrm{E}} - u_{\mathrm{W}}}{2\Delta x} + \frac{v_{\mathrm{N}} - v_{\mathrm{S}}}{2\Delta y}, \tag{10.13}$$

which could be obtained as a finite difference approximation by direct application of central difference formulas to the derivatives.

The next step is to find an approximation of the pressure gradient ∇p. The second-order approximation of integrals of pressure derivatives, such as (10.6), can be derived similarly to (10.10) and (10.11):

$$\frac{1}{\Delta x\,\Delta y}\int_{\partial\Omega} p n_x\, dS \approx \frac{p_{\mathrm{e}} - p_{\mathrm{w}}}{\Delta x} = \left.\frac{\delta p}{\delta x}\right|_{\mathrm{P}}, \tag{10.14}$$

$$\frac{1}{\Delta x\,\Delta y}\int_{\partial\Omega} p n_y\, dS \approx \frac{p_{\mathrm{n}} - p_{\mathrm{s}}}{\Delta y} = \left.\frac{\delta p}{\delta y}\right|_{\mathrm{P}}, \tag{10.15}$$

where we, again, use symbolic notation for approximations of pressure derivatives.

The values at the face points are obtained by linear interpolation as in (10.12), which results in the formulas

$$\left.\frac{\delta p}{\delta x}\right|_{\mathtt{P}} = \frac{p_{\mathtt{E}} - p_{\mathtt{W}}}{2\Delta x}, \quad \left.\frac{\delta p}{\delta y}\right|_{\mathtt{P}} = \frac{p_{\mathtt{N}} - p_{\mathtt{S}}}{2\Delta y}. \tag{10.16}$$

This could also be obtained as central difference approximations of the components of $\nabla p|_{\mathtt{P}}$ on a finite difference grid.

We now consider the pressure equation in the form of (10.5) or (10.8) and recall that the exact mass conservation by a numerical solution requires two conditions: (i) that the divergence operators in the left-hand and right-hand sides are approximated by the same scheme and (ii) that the gradient operator in the left-hand side is approximated by the same scheme as the pressure gradient in the momentum equation. Applying the approximation of divergence (10.13) to the components of ∇p and $\boldsymbol{F}$, we obtain

$$\begin{aligned}\frac{(\delta p/\delta x)_{\mathtt{E}} - (\delta p/\delta x)_{\mathtt{W}}}{2\Delta x} + \frac{(\delta p/\delta y)_{\mathtt{N}} - (\delta p/\delta y)_{\mathtt{S}}}{2\Delta y} \\ = \frac{F_{xE} - F_{xW}}{2\Delta x} + \frac{F_{yN} - F_{yS}}{2\Delta y}.\end{aligned} \tag{10.17}$$

The pressure derivatives are approximated as in (10.16), except that the formulas should be taken for the cells with the grid points E, W, S, and N instead of P. For example, we should use

$$\left.\frac{\delta p}{\delta x}\right|_{\mathtt{E}} = \frac{p_{\mathtt{EE}} - p_{\mathtt{P}}}{2\Delta x}.$$

The final approximation of the pressure equation is

$$\frac{p_{\mathtt{EE}} - 2p_{\mathtt{P}} + p_{\mathtt{WW}}}{(2\Delta x)^2} + \frac{p_{\mathtt{NN}} - 2p_{\mathtt{P}} + p_{\mathtt{SS}}}{(2\Delta y)^2} = \frac{F_{xE} - F_{xW}}{2\Delta x} + \frac{F_{yN} - F_{yS}}{2\Delta y}. \tag{10.18}$$

The expression on the left-hand side is the familiar five-point operator, which has the second order of accuracy in x and y. It has a strange feature, however, that grid steps of double size $2\Delta x$ and $2\Delta y$ are taken. Albeit not especially troubling at first glance, this may lead to dangerous behavior in the form of spurious oscillations of the pressure field. The reason is the splitting of the system of discretization equations into four subsystems, which are only coupled with each other in weak sense via the right-hand sides. Each such subsystem contains the equations that connect the pressure values at cells marked by only one of the numbers 0, 1, 2,

or 3 in Figure 10.1. For example, it is easy to see that the left-hand side of (10.18) connects the values of p at the 0 cells. Any equation written for any other 0 cell connects only 0 cells. Similarly, equations for a 1 cell connect 1 cells but not the others, and so on.

As a particularly striking example of the splitting effect, let us assume that the numbers 0, 1, 2, and 3 in Figure 10.1 represent actual values of pressure in the corresponding cells. Such a bizarre distribution would be a solution of the pressure equation with zero right-hand side. The approximations (10.16) would ignore the pressure variations and show the gradient ∇p as identically zero.

The source of trouble is, of course, the use of double grid steps $2\Delta x$ and $2\Delta y$ in the approximations of the first derivatives in the gradient and divergence operators. Can this be avoided? The positive answer was first given by Harlow and Welch in 1965 in the form of a *staggered grid*.

The idea is simple and based on understanding that we do not have to use the same set of grid points for all variables. In particular, we avoid double step differences, such as (10.13) and (10.16) by evaluating velocity components directly at the corresponding face points instead of the grid points located at the centers of the cells. The u-component is calculated at the midpoints $\mathtt{e}$ and $\mathtt{w}$ of the eastern and western faces of each cell, while the midpoints $\mathtt{n}$ and $\mathtt{s}$ of the northern and southern faces are used for the v-component.

The same arrangement is used for the components of vector $\boldsymbol{F}$. The grid points at the cell centers are still used for the pressure. The formulas (10.11) can now be applied directly without need for an interpolation. The main benefit, however, is removing the splitting problem. If we use (10.11) instead of (10.13) as an approximation of the divergence operator, the consistent approximation of the pressure equation becomes, instead of (10.17),

$$\begin{aligned}\frac{(\delta p/\delta x)_{\mathtt{e}} - (\delta p/\delta x)_{\mathtt{w}}}{\Delta x} + \frac{(\delta p/\delta y)_{\mathtt{n}} - (\delta p/\delta y)_{\mathtt{s}}}{\Delta y} \\ = \frac{F_{x\,\mathtt{e}} - F_{x\,\mathtt{w}}}{\Delta x} + \frac{F_{y_{\mathtt{n}}} - F_{y_{\mathtt{s}}}}{\Delta y}.\end{aligned} \tag{10.19}$$

The partial derivatives of pressure at the face points can be approximated by single step differences:

$$\left.\frac{\delta p}{\delta x}\right|_{\mathtt{e}} = \frac{p_{\mathtt{E}} - p_{\mathtt{P}}}{\Delta x}, \quad \left.\frac{\delta p}{\delta x}\right|_{\mathtt{w}} = \frac{p_{\mathtt{P}} - p_{\mathtt{W}}}{\Delta x},$$

$$\left.\frac{\delta p}{\delta y}\right|_{\mathtt{n}} = \frac{p_{\mathtt{N}} - p_{\mathtt{P}}}{\Delta y}, \quad \left.\frac{\delta p}{\delta y}\right|_{\mathtt{s}} = \frac{p_{\mathtt{P}} - p_{\mathtt{S}}}{\Delta y}.$$

Substitution into (10.19) results in the standard five-point scheme

$$\frac{p_{\mathrm{E}} - 2p_{\mathrm{P}} + p_{\mathrm{W}}}{(\Delta x)^2} + \frac{p_{\mathrm{N}} - 2p_{\mathrm{P}} + p_{\mathrm{S}}}{(\Delta y)^2} = \frac{F_{x_{\mathrm{e}}} - F_{x_{\mathrm{W}}}}{\Delta x} + \frac{F_{y_{\mathrm{n}}} - F_{y_{\mathrm{S}}}}{\Delta y}. \quad (10.20)$$

Equations for every grid point are now strongly coupled by the left-hand sides. Unphysical oscillations like those shown in Figure 10.1 are registered by the pressure gradient and reacted upon by the velocity field.

To be able to evaluate the components of $\boldsymbol{u}$ and $\boldsymbol{F}$ directly at the face points we have to introduce additional sets of grid points or, in finite volume methods, cells. This leads to the staggered grid arrangements illustrated by two-dimensional examples in Figure 10.2. Generalization to the three-dimensional case is trivial.

In finite difference methods (see Figure 10.2*a*), we evaluate pressure and approximate the pressure equation (satisfy the incompressibility condition) at the integer grid points (x_i, y_j) shown by solid circles. These points are also used for other scalar fields (e.g., temperature). The velocity components are evaluated and momentum equations are solved at points of half-integer grids obtained from the integer grid by shifting in the x or

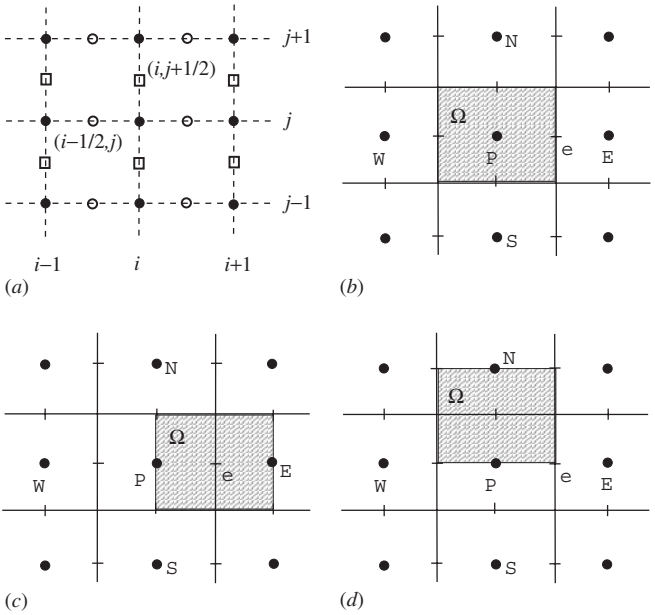


Figure 10.2 Examples of two-dimensional staggered grids. *(a)* finite difference grid. *(b)–(d)* finite volume grid; Separate figures present cells used to satisfy mass balance *(b)*, *x*-momentum balance *(c)*, and *y*-momentum balance *(d)*.

y direction by half a grid step. The points $(x_{i+1/2}, y_j)$ shown by hollow circles are used to evaluate u, while v is computed at points $(x_i, y_{j+1/2})$ shown by squares.

In the finite volume method, additional sets of cells are created. Examples of cells used to solve the pressure or other scalar equation, u-momentum equation, and v-momentum equation are shown in Figure 10.2*b*, *c*, and *d*, respectively.

The staggered arrangement increases complexity of a scheme. Programming becomes more difficult, since it requires accounting for three (or four in the three-dimensional case) indexing systems. Interpolations must be used to compute nonlinear terms of momentum equations. Further complications arise when the grid is nonuniform. All these difficulties, however, can be relatively easily handled in computations with structured grids such as those shown in Figure 10.2. For this reason and because of the benefit of removing the splitting problem, the staggered arrangement was by far the most popular choice during early years of CFD.

The difficulties of handling a staggered arrangement increase significantly when unstructured grids are used. When such grids started to be broadly applied in general-purpose codes in recent years, colocated arrangements returned to favor. This area of CFD is still evolving and, in general, requires discussion on a more advanced level than appropriate for this book. We only mention that methods have been developed to cure the splitting problem. Some of them are based on filtering out the oscillating component of pressure field or periodic averaging of pressure values at neighboring points. Others avoid the problem by fully or partially neglecting the requirement that $\nabla^2 p$ in the pressure equation, $\nabla \cdot \boldsymbol{V}$ in the incompressibility condition, and ∇p in the momentum equation are discretized consistently. The property of exact mass and kinetic energy conservation is lost if this approach is taken. It was, however, shown, for example by Morinishi et al in 1998, that the schemes can be arranged so that the error is relatively small and does not induce numerical instability.

10.3 PROJECTION METHOD FOR UNSTEADY FLOWS

In this section, we consider the procedure commonly used to compute time-dependent flows of incompressible flows. The procedures for steady-state equations, which we review in the next section, follow the same basic principle, although they are implemented differently. The details of spatial discretization are of little importance at this moment and are, therefore, omitted. We assume in the following discussion that spatial derivatives are discretized using a finite difference, finite volume or, perhaps, spectral scheme.

The problem of pressure calculation in unsteady solutions is formulated as follows: *Given the solution* p^n, $\boldsymbol{V}^n$ *at the previous time layer* t^n *find the next time-layer pressure* p^{n+1} *and velocity* $\boldsymbol{V}^{n+1}$ *such that they together satisfy the momentum equation, and the velocity is divergence-free* $\nabla \cdot \boldsymbol{V}^{n+1} = 0$ *and satisfies the boundary conditions.*

A widely used and well-developed approach is based on the *pressure-correction* or *projection* method. The general strategy is to decompose each time step into two substeps. On the first substep, the momentum equation is solved for the velocity components. The pressure gradient is either removed from the equation or approximated by an estimate. The obtained velocity field cannot be considered a solution at the new time level since it does not satisfy the incompressibility condition. The second substep is, therefore, needed, at which the correct pressure distribution is found and the correction of velocity is made. The term *projection* reflects the fact that we find a preliminary solution, which is not divergence-free, and then project it onto the space of divergence-free vector functions. Versions of the projection method were proposed by Harlow and Welch in 1965 (the marker-and-cell method), Chorin in 1968, Temam in 1969, and Yanenko in 1971 (the fractional-step method) and further developed in 1980s and 1990s.

10.3.1 Explicit Schemes

As the first illustration, we solve a marching problem using the simple explicit scheme

$$\frac{\boldsymbol{V}^{n+1} - \boldsymbol{V}^n}{\Delta t} = -\frac{1}{\rho}\nabla p^{n+1} + \left[-\boldsymbol{N}(\boldsymbol{V}, \boldsymbol{V}) + \nu\nabla^2 \boldsymbol{V} + \boldsymbol{f}\right]^n =$$

$$-\frac{1}{\rho}\nabla p^{n+1} + \boldsymbol{F}^n, \tag{10.21}$$

$$\nabla \cdot \boldsymbol{V}^{n+1} = 0, \tag{10.22}$$

where $\boldsymbol{F}^n$ is the shorthand notation for the combination of all terms of the right-hand side except the pressure gradient. The pressure term is marked as belonging to the time layer t^{n+1} to stress its role in ensuring incompressibility at that time layer.

As we already saw, the incompressibility condition is fulfilled if the pressure field satisfies the Poisson equation (10.5). In the case of the simple explicit scheme (10.21)–(10.22), the equation takes the form

$$\nabla^2 p^{n+1} = \rho \nabla \cdot \boldsymbol{F}^n, \tag{10.23}$$

which can be easily derived by applying the divergence operator directly to the discretized momentum equation (10.21) and requiring that $\nabla \cdot \boldsymbol{V}^n = \nabla \cdot \boldsymbol{V}^{n+1} = 0$.

The solution procedure is as follows. Instead of the original system (10.21), (10.22) we solve the equivalent system (10.21), (10.23). To advance from the time layer t^n to the layer t^{n+1}, we first calculate $\boldsymbol{F}^n$. It is based on the known velocity $\boldsymbol{V}^n$ and does not include the pressure term. We then solve the pressure equation (10.23) and use ∇p^{n+1} and $\boldsymbol{F}^n$ to update the velocity according to (10.21).

The procedure can be transformed into a two-step cycle of the form common for all projection methods. To do so, we split the velocity update into two substeps. On the first, $\boldsymbol{F}^n$ is taken into account to produce the intermediate velocity field $\boldsymbol{V}^*$:

$$\text{Predictor: } \frac{\boldsymbol{V}^* - \boldsymbol{V}^n}{\Delta t} = \boldsymbol{F}^n \text{ or } \boldsymbol{V}^* = \boldsymbol{V}^n + \Delta t \boldsymbol{F}^n. \tag{10.24}$$

The pressure gradient is added at the second substep to satisfy the incompressibility constraint:

$$\text{Corrector: } \frac{\boldsymbol{V}^{n+1} - \boldsymbol{V}^*}{\Delta t} = -\frac{1}{\rho}\nabla p^{n+1} \text{ or } \boldsymbol{V}^{n+1} = \boldsymbol{V}^* - \frac{\Delta t}{\rho}\nabla p^{n+1}. \tag{10.25}$$

It is easy to verify that summing (10.24) and (10.25) generates the original scheme (10.21). The pressure equation is solved either in the beginning of the cycle or between the substeps. It can be rewritten as

$$\nabla^2 p^{n+1} = \frac{\rho}{\Delta t}\nabla \cdot \boldsymbol{V}^*, \tag{10.26}$$

which follows from (10.23) and (10.24) or from direct application of the divergence operator to (10.25).

An interesting and important question arises as to what boundary conditions should be used for the pressure field. Such conditions are required at every point of the boundary for the Poisson problem (10.23) or (10.26) to be well posed. The conditions, however, do not naturally follow from the flow physics for the boundaries between a fluid and solid walls, unless, of course, a full fluid-structure interaction problem is solved. Since the latter option is, in most cases, an unnecessary complication, we have to find a way to derive the pressure boundary conditions from the equations themselves.

To do so, we observe that the velocity field should satisfy the impermeability condition $\boldsymbol{V} \cdot \boldsymbol{n}|_{\partial\Omega} = 0$, where $\boldsymbol{n}$ is the normal to the solid wall boundary $\partial\Omega$. In the case of an explicit scheme, this results in a simple boundary condition for pressure. Taking the projection of the momentum equation (10.21) on $\boldsymbol{n}$ we find

$$\left(-\frac{1}{\rho}\nabla p^{n+1} \cdot \boldsymbol{n} + \boldsymbol{F}^n \cdot \boldsymbol{n}\right)_{\partial\Omega} = 0 \quad \text{or} \quad \left.\frac{\partial p^{n+1}}{\partial n}\right|_{\partial\Omega} = \rho \boldsymbol{F}^n \cdot \boldsymbol{n}\big|_{\partial\Omega} = \left.\frac{\rho}{\Delta t}\boldsymbol{V}^* \cdot \boldsymbol{n}\right|_{\partial\Omega}. \tag{10.27}$$

The wall-normal component of $\boldsymbol{F}^n$ is, generally, nonzero. For example, let us consider a solid wall located at $x = 0$. The wall-normal component is

$$F_x^n\big|_{x=0} = \left(\nu\frac{\partial^2 u^n}{\partial x^2} + f_x^n\right)_{x=0} \neq 0.$$

In deriving these formulas we have used the no-slip condition satisfied by the velocity field: $u^n = v^n = w^n = 0$ at $\partial\Omega$.

Note that, for an explicit method, the no-slip conditions for the wall-tangential components of velocity $\boldsymbol{V}^{n+1}$ are not enforced by the procedure and have to be imposed after every time step.

The scheme outlined here remains valid for other fully explicit methods based, for example, on the Adams-Bashfort or Runge-Kutta time integration algorithms. The only modification is in the organization of the predictor step (10.24).

Solving the Poisson equation for pressure is a computationally expensive part of the solution. This is particularly true in the case of explicit methods, where the Poisson equation requires up to 90 percent of the total number of computer operations. Realizing that and taking into account that the Poisson equation is often solved by one of the iterative methods (see Chapter 8), we immediately see a way to improve efficiency of the projection algorithm. The number of iterations needed to achieve the desired accuracy is reduced, if a good initial guess of pressure distribution is employed as a starting point. In our case, the guess is readily available in the form of the pressure field from the previous time step $p_0 = p^n$ or an interpolation from several previous steps, such as $p_0 = 2p^n - p^{n-1}$. It can be used in two different ways. First, we can directly apply p_0 to start the first iteration. Alternatively, we can include the effect of p_0 into $\boldsymbol{F}^n$.

The two-step version of the algorithm (10.24)–(10.25) is modified as

$$\text{Predictor:} \qquad \boldsymbol{V}^* = \boldsymbol{V}^n + \Delta t \boldsymbol{F}^n - \frac{\Delta t}{\rho}\nabla p_0 \tag{10.28}$$

$$\text{Corrector:} \qquad \boldsymbol{V}^{n+1} = \boldsymbol{V}^* - \frac{\Delta t}{\rho}\nabla(\delta p), \tag{10.29}$$

where $\delta p = p^{n+1} - p_0$ is the pressure correction. The Poisson equation and boundary conditions for δp are expressed in terms of $\boldsymbol{V}^*$ in the same way as in (10.26) and (10.27), but the right-hand sides are closer to zero in terms of an appropriate norm. One can expect (and in many cases the expectations are fulfilled) that the number of iterations needed to solve the Poisson equation decreases.

10.3.2 Implicit Schemes

Talking about implicit methods, one has to distinguish between two approaches: the fully implicit approach, according to which all terms including the nonlinear term are approximated at the new time layer t^{n+1}, and the semi-implicit approach, in which the implicit treatment is limited to the viscous term and pressure.

The fully implicit approach leads to a system of nonlinear discretization equations, which has to be solved at every time step. Such systems have already been discussed in section 8.4. We have found that their direct solution, for example by the Newton's method, is inefficient. Efficiency can be achieved using an iteration procedure and linearization. The same general approach can be applied, after some modification, to fully implicit schemes for incompressible flows. Because of importance and certain peculiar features (primarily related to the role of pressure), it is worthwhile to present a detailed description of the method.

As an illustration, we consider the simple implicit method with time-discretized equations written as

$$\frac{\boldsymbol{V}^{n+1} - \boldsymbol{V}^n}{\Delta t} = -\frac{1}{\rho}\nabla p^{n+1} - \boldsymbol{N}(\boldsymbol{V}^{n+1}, \boldsymbol{V}^{n+1}) + \nu\nabla^2 \boldsymbol{V}^{n+1} + \boldsymbol{f}^{n+1} \tag{10.30}$$

$$\nabla \cdot \boldsymbol{V}^{n+1} = 0. \tag{10.31}$$

As before, some kind of spatial discretization is assumed but not shown.

The formal Poisson equation for pressure (10.5) can be easily rewritten for the implicit scheme. Applying the divergence operator to (10.30) we obtain, assuming that $\nabla \cdot \boldsymbol{V}^n = 0$ and requiring that $\nabla \cdot \boldsymbol{V}^{n+1} = 0$,

$$\nabla^2 p^{n+1} = \rho \nabla \cdot \left[-\boldsymbol{N}(\boldsymbol{V}^{n+1}, \boldsymbol{V}^{n+1}) + \boldsymbol{f}^{n+1} \right] = \rho \nabla \cdot \boldsymbol{F}^{n+1}. \quad (10.32)$$

We see the problem now. Unlike the explicit version (10.23), the right-hand side of the pressure equation now uses velocity $\boldsymbol{V}^{n+1}$, which should already include the pressure correction ∇p^{n+1}. The momentum and pressure equations (10.30), (10.32) are fully coupled and have to be solved simultaneously. The matter is further complicated by nonlinearity of the discretized momentum equation. We have to apply iterations and linearization to solve the system.

The iterative procedures are discussed in the next section. Here we consider the version of the method valid in the case of sufficiently small Δt. Convergence of iterations is not required in this method. In fact, we only perform one iteration at a time step. This introduces additional error, which we will estimate later. The method uses linearization about the solution obtained at the previous time layer and is arranged as a predictor-corrector procedure similar to the procedure introduced earlier for the explicit method.

We address the linearization first. The unknown velocity and pressure are represented as sums of the known values at t^n and perturbations:

$$\boldsymbol{V}^{n+1} = \boldsymbol{V}^n + \delta \boldsymbol{V}, \quad p^{n+1} = p^n + \delta p. \quad (10.33)$$

The nonlinear term is quadratic in velocity components. It can be rewritten as

$$\begin{aligned} \boldsymbol{N}(\boldsymbol{V}^{n+1}, \boldsymbol{V}^{n+1}) &= \boldsymbol{N}(\boldsymbol{V}^n, \boldsymbol{V}^n) + \boldsymbol{N}(\boldsymbol{V}^n, \delta \boldsymbol{V}) \\ &+ \boldsymbol{N}(\delta \boldsymbol{V}, \boldsymbol{V}^n) + \boldsymbol{N}(\delta \boldsymbol{V}, \delta \boldsymbol{V}). \end{aligned} \quad (10.34)$$

If the time step Δt is small, the velocity perturbations can be estimated as

$$\delta \boldsymbol{V} \sim \frac{\partial \boldsymbol{V}}{\partial t} \Delta t.$$

The last term in the right-hand side of (10.34) is quadratic in $\delta \boldsymbol{V}$ and, thus, is of the order of $O\left((\Delta t)^2\right)$. It can be neglected in comparison with other terms that are either $O(1)$ or $O(\Delta t)$. Dropping the quadratic term resolves the issue of nonlinearity at the price of introducing the error $\sim O\left((\Delta t)^2\right)$ into the solution. The price is fair, since the error is of

the higher order than the truncation error of the simple implicit formula (10.30). The linearization would still be justified if we used a second-order time discretization scheme.

The linearized version of (10.30), (10.31) is

$$\frac{\boldsymbol{V}^{n+1}-\boldsymbol{V}^{n}}{\Delta t}=-\frac{1}{\rho}\nabla p^{n}-\frac{1}{\rho}\nabla\delta p-\widetilde{\boldsymbol{N}}(\boldsymbol{V}^{n},\boldsymbol{V}^{n+1})+\nu\nabla^{2}\boldsymbol{V}^{n+1}+\boldsymbol{f}^{n+1}, \tag{10.35}$$

$$\nabla\cdot\boldsymbol{V}^{n+1}=0, \tag{10.36}$$

where $\widetilde{\boldsymbol{N}}(\boldsymbol{V}^{n},\boldsymbol{V}^{n+1})=\boldsymbol{N}(\boldsymbol{V}^{n},\boldsymbol{V}^{n})+\boldsymbol{N}(\boldsymbol{V}^{n},\delta\boldsymbol{V})+\boldsymbol{N}(\delta\boldsymbol{V},\boldsymbol{V}^{n})$ is the linearized convection term. We assume that the body force is a constant, a function of space and time, or a linear function of $\boldsymbol{V}$.

We still have to deal with the problem of coupling between momentum and pressure equations. The solution follows the predictor-corrector scheme. On the predictor substep, the linear system (10.35) is solved with ∇p^{n} used as an estimate of the pressure gradient. The perturbation δp is omitted. This produces the intermediate velocity field $\boldsymbol{V}^{*}$.

On the corrector substep, the velocity field is updated as $\boldsymbol{V}^{n+1}=\boldsymbol{V}^{*}-\rho^{-1}\Delta t\nabla\delta p$. The requirement of incompressibility of $\boldsymbol{V}^{n+1}$ gives the compact form of the pressure equation:

$$\nabla^{2}\delta p=\frac{\rho}{\Delta t}\nabla\cdot\boldsymbol{V}^{*}. \tag{10.37}$$

Note that this equation has to be solved between the predictor and corrector substeps.

The predictor part of the procedure is computationally costly, significantly more so than in the case of explicit methods, since it requires solution of a large linear system (the matrix size is three times the total number of grid points). Methods have been developed to achieve computational efficiency—for example, the methods based on the approximate factorization approach (see Chapter 9) or the efficient direct and iterative techniques for solution of matrix equations (see Chapter 8).

An alternative to the fully implicit schemes is the semi-implicit approach. In this approach, we treat the nonlinear term explicitly so no linearization is needed. The simple semi-implicit time discretization scheme is

$$\frac{\boldsymbol{V}^{n+1}-\boldsymbol{V}^{n}}{\Delta t}=-\frac{1}{\rho}\nabla p^{n+1}-\boldsymbol{N}(\boldsymbol{V}^{n},\boldsymbol{V}^{n})+\nu\nabla^{2}\boldsymbol{V}^{n+1}+\boldsymbol{f}^{n+1} \tag{10.38}$$

$$\nabla\cdot\boldsymbol{V}^{n+1}=0. \tag{10.39}$$

The projection procedure is organized as a sequence of the predictor substep, on which we solve the momentum equation with ∇p^n or without pressure at all, solution of the pressure equation (10.37) or (10.26), and the correction substep. Similarly to the fully implicit linearized solution, the predictor substep requires solving a large linear system for the components of the intermediate velocity $\boldsymbol{V}^*$.

One important difference between the fully implicit and semi-implicit methods is in their stability characteristics. The fully implicit schemes are typically unconditionally stable, while semi-implicit schemes have stability limits on the time step. This does not necessarily mean that the fully implicit schemes are more efficient. The time step should be reasonably small anyway in order to keep the linearization and truncation errors under control.

The linear elliptic problems for $\boldsymbol{V}^*$ encountered in both versions of the implicit method need boundary conditions on the boundaries $\partial\Omega$ of the computational domain. Correct formulation of these conditions and of the boundary conditions for pressure is a nontrivial matter, which is discussed in the research literature and cannot be adequately addressed here.

10.4 PROJECTION METHODS FOR STEADY-STATE FLOWS

There is a strong similarity between steady-state problems and transient problems solved by fully implicit methods. In both cases, the spatial discretization is conducted in terms of unknown variables. Almost identical systems of nonlinear discretization equations have to be solved, once for steady-state problems and at every time step for transient problems.

We learned in the last section that the solution can be simplified in the case of transient problems if time steps are small. We now turn our attention to the general case, in which a nonlinear system is actually solved using iterations and linearization. The problem under consideration is either a steady-state problem or a transient problem integrated with large time steps. The same numerical procedure is applied in both cases.

We will give an overview of several popular methods. For the sake of consistency with other textbooks (e.g., Patankar (1980) and Ferziger and Perić (2001)) and with documentation of many CFD codes, the discussion will use equations written in spatially discretized form. This should not obscure the fact that the methods are based on the same basic principles as the schemes described in the previous sections: projection and linearization of convection term.

The discretized momentum equation is written as

$$a_{\mathrm{P}}(\boldsymbol{u})u_{i,\mathrm{P}} + \sum_{\ell} a_{\ell,\mathrm{P}}(\boldsymbol{u})u_{i,\ell} = Q_{\mathrm{P}}(\boldsymbol{u}) - \left(\frac{\delta p}{\delta x_i}\right)_{\mathrm{P}}, \tag{10.40}$$

where u_i is the velocity component and, as before, we use symbolic expressions, such as $\delta p/\delta x_i$, for discretized partial derivatives. The equation is a result of spatial discretization of the fully implicit formula (10.30) or of a steady-state momentum equation. The difference between the two cases is in the form of $a_{\mathrm{P}}(\boldsymbol{u})$ and $Q_{\mathrm{P}}(\boldsymbol{u})$. The discretization is conducted for the grid point P (in finite difference methods) or cell P (in finite volume methods). The left-hand side represents the discretized nonlinear and viscous terms. The summation index ℓ runs over all neighboring nodes used by the discretization formulas. The equation is written in the general way, which does not imply any particular grid or any particular method of discretization. Since the methods discussed here are commonly applied by the general purpose CFD codes, however, it is useful to view (10.40) as a result of discretization on an unstructured finite volume grid.

Note that, as a result of the quadratic nonlinearity in the momentum equations, the coefficients a_{P} and $a_{\ell,\mathrm{P}}$ in the left-hand side of (10.40) are functions of velocity $\boldsymbol{u}$. The body force and other terms, which are linear functions of $\boldsymbol{u}$, are lumped together into the source Q. The last term in the right-hand side is the spatially discretized pressure gradient.

The solution has to satisfy the incompressibility condition

$$\left.\frac{\delta u_i}{\delta x_i}\right|_{\mathrm{P}} = 0, \tag{10.41}$$

where summation over repeating indices is assumed.

The system of coupled equations (10.40), (10.41) is solved in a sequence of converging approximations, the so-called *outer iterations*. An iteration is based on the projection method and linearization, and consists of the following principal substeps:

Step 1: Use the best currently available approximations of velocity $u_i^{(m)}$ and pressure $p^{(m)}$ to evaluate the coefficients a_{P} and $a_{\ell,\mathrm{P}}$ and the terms Q_{P} and $(\delta p/\delta x_i)_{\mathrm{P}}$ in the right-hand side of (10.40).

Step 2: Solve the linearized momentum equations

$$a_{\mathrm{P}}(\boldsymbol{u}^{(m)})u^*_{i,\mathrm{P}} + \sum_{\ell} a_{\ell,\mathrm{P}}(\boldsymbol{u}^{(m)})u^*_{i,\ell} = Q_{\mathrm{P}}(\boldsymbol{u}^{(m)}) - \left(\frac{\delta p^{(m)}}{\delta x_i}\right)_{\mathrm{P}}. \tag{10.42}$$

This is a linear system usually characterized by a sparse coefficient matrix. It can be efficiently solved by the method of sequential iterations. The steps of this procedure are called *inner iterations*. At the end, we obtain the intermediate velocity field u_i^*, which does not satisfy the incompressibility condition.

Step 3: Solve the pressure equation to find the new pressure field $p^{(m+1)}$. The exact form of the pressure equation varies with the implementation of the method. Several forms are presented in sections 10.4.1 and 10.4.2.

Step 4: Use the new pressure field to update velocity. The result is the new approximation $u_i^{(m+1)}$.

Step 5: Test convergence. If desired accuracy is not achieved, repeat the procedure, starting at Step 1.

Several variations of this general procedure have been developed. They are widely used today. We will discuss the most popular among them.

10.4.1 SIMPLE

One of the first versions of the method is SIMPLE (Semi-Implicit Method for Pressure Linked Equations) (see Caretto et al. (1972) and Patankar and Spalding (1972)). The method was originally designed for finite volume approximation with staggered grid arrangement but can be straightforwardly extended to other discretization techniques. Our description is not tied to any particular discretization.

In the method, the new values of velocity and pressure are represented as

$$u_i^{(m+1)} = u_i^* + u_i', \quad p^{(m+1)} = p^{(m)} + p', \tag{10.43}$$

where u_i^* is the solution of the linearized momentum equation (10.42) with $p^{(m)}$ used as an estimate of pressure.

To find the relation between u_i' and p' we require that $u_i^{(m+1)}$ and $p^{(m+1)}$ satisfy the linearized momentum equation

$$a_{\mathrm{P}}(\boldsymbol{u}^{(m)})u_{i,\mathrm{P}}^{(m+1)} + \sum_{\ell} a_{\ell,\mathrm{P}}(\boldsymbol{u}^{(m)})u_{i,\ell}^{(m+1)} = Q_{\mathrm{P}}(\boldsymbol{u}^{(m)}) - \left(\frac{\delta p^{(m+1)}}{\delta x_i}\right)_{\mathrm{P}}. \tag{10.44}$$

Subtracting (10.42) from (10.44), we find

$$a_{\mathrm{P}}(\boldsymbol{u}^{(m)})u_{i,\mathrm{P}}' + \sum_{\ell} a_{\ell,\mathrm{P}}(\boldsymbol{u}^{(m)})u_{i,\ell}' = -\left(\frac{\delta p'}{\delta x_i}\right)_{\mathrm{P}}. \tag{10.45}$$

This is, again, a linearized system, which, at the moment, cannot be solved for either u_i' or p', since both fields are yet unknown. We formally represent the solution as

$$u_{i,\mathrm{P}}' = \tilde{u}_{i,\mathrm{P}}' - \frac{1}{a_{\mathrm{P}}(\boldsymbol{u}^{(m)})}\left(\frac{\delta p'}{\delta x_i}\right)_{\mathrm{P}}, \tag{10.46}$$

where $\tilde{u}_{i,\mathrm{P}}'$ is an unknown function defined by (10.46).

We are now in a position to formally derive the pressure equation. Substituting the first expansion of (10.43) into the incompressibility condition

$$\left.\frac{\delta u_i^{(m+1)}}{\delta x_i}\right|_{\mathrm{P}} = 0 \tag{10.47}$$

and using (10.46), we obtain

$$\left(\frac{\delta u_i^*}{\delta x_i}\right)_{\mathrm{P}} + \left(\frac{\delta \tilde{u}_i'}{\delta x_i}\right)_{\mathrm{P}} - \frac{\delta}{\delta x_i}\left(\frac{1}{a_{\mathrm{P}}(\boldsymbol{u}^{(m)})}\frac{\delta p'}{\delta x_i}\right)_{\mathrm{P}} = 0$$

or

$$\frac{\delta}{\delta x_i}\left(\frac{1}{a_{\mathrm{P}}(\boldsymbol{u}^{(m)})}\frac{\delta p'}{\delta x_i}\right)_{\mathrm{P}} = \left(\frac{\delta u_i^*}{\delta x_i}\right)_{\mathrm{P}} + \left(\frac{\delta \tilde{u}_i'}{\delta x_i}\right)_{\mathrm{P}}. \tag{10.48}$$

We see a familiar problem. The equation contains $\tilde{u}_i'$ and p', both of which are unknown. In the SIMPLE method, the problem is resolved in a bold move. The unknown terms with $\tilde{u}_i'$ are simply removed from (10.46) and (10.48). There is no fully satisfying justification of such a drastic simplification except that the SIMPLE schemes have been found to converge in many cases. It has to be noted, however, that the convergence is slower than with more sophisticated methods discussed next.

To summarize, an outer iteration of the SIMPLE method consists of the following substeps:

Step 1: Evaluate the coefficients a_{P} and $a_{\ell,\mathrm{P}}$ and the terms of the right-hand side of the linearized momentum equation (10.42) using the velocity and pressure fields from the previous iteration.

Step 2: Solve (10.42) to find the intermediate velocity u_i^*

Step 3: Solve the approximate pressure equation (note the difference with the full equation (10. 48)):

$$\frac{\delta}{\delta x_i}\left(\frac{1}{a_{\mathrm{P}}(\boldsymbol{u}^{(m)})}\frac{\delta p'}{\delta x_i}\right)_{\mathrm{P}} = \left(\frac{\delta u_i^*}{\delta x_i}\right)_{\mathrm{P}}. \tag{10.49}$$

Step 4: Update velocity and pressure fields as

$$u'_{i,\mathrm{P}} = -\frac{1}{a_{\mathrm{P}}(\boldsymbol{u}^{(m)})}\left(\frac{\delta p'}{\delta x_i}\right)_{\mathrm{P}}, \quad u_{i,\mathrm{P}}^{(m+1)} = u_{i,\mathrm{P}}^* + u'_{i,\mathrm{P}},$$
$$p_{\mathrm{P}}^{(m+1)} = p_{\mathrm{P}}^{(m)} + p'_{\mathrm{P}}. \tag{10.50}$$

Step 5: Check convergence and start the next iteration if needed.

The convergence of SIMPLE algorithm can be accelerated by using successive underrelaxation; that is, by updating pressure and velocity as

$$u_{i,\mathrm{P}}^{(m+1)} = u_{i,\mathrm{P}}^* + \alpha_u u'_{i,\mathrm{P}}, \quad p_{\mathrm{P}}^{(m+1)} = p_{\mathrm{P}}^{(m)} + \alpha_p p'_{\mathrm{P}},$$

where $0 < \alpha_u \leq 1$ and $0 < \alpha_p \leq 1$ are underrelaxation coefficients. Rather small value of α_p is recommended on the basis of empirical studies. It can be shown that the fastest convergence is obtained when $\alpha_u = 1 - \alpha_p$.

10.4.2 SIMPLEC, SIMPLER, and PISO

The drastic simplification of completely neglecting the terms with $\tilde{u}'_i$ results in slow convergence of SIMPLE. Improved versions of the algorithm have been developed and are widely applied in modern CFD codes.

One such version is SIMPLEC (SIMPLE Consistent) proposed by van Doormal and Raithby in 1984. The unknown terms with $\tilde{u}'_i$ are approximated rather than neglected. To derive the approximation, we start with formally expressing $\tilde{u}'_i$ in terms of u'_i. From (10.45), we obtain

$$u'_{i,\mathrm{P}} = -\frac{\sum_\ell a_{\ell,\mathrm{P}}(\boldsymbol{u}^{(m)})u'_{i,\ell}}{a_{\mathrm{P}}(\boldsymbol{u}^{(m)})} - \frac{1}{a_{\mathrm{P}}(\boldsymbol{u}^{(m)})}\left(\frac{\delta p'}{\delta x_i}\right)_{\mathrm{P}}.$$

In combination with (10.46), this leads to

$$\tilde{u}'_{i,\mathrm{P}} = -\frac{\sum_\ell a_{\ell,\mathrm{P}}(\boldsymbol{u}^{(m)})u'_{i,\ell}}{a_{\mathrm{P}}(\boldsymbol{u}^{(m)})}. \tag{10.51}$$

As a next step, we approximate u_i' at every grid point by a weighted average of the values at neighboring points as

$$u'_{i,\mathrm{P}} \approx \frac{\sum_\ell a_{\ell,\mathrm{P}}(\boldsymbol{u}^{(m)}) u'_{i,\ell}}{\sum_\ell a_{\ell,\mathrm{P}}(\boldsymbol{u}^{(m)})} \tag{10.52}$$

or

$$\sum_\ell a_{\ell,\mathrm{P}}(\boldsymbol{u}^{(m)}) u'_{i,\ell} \approx u'_{i,\mathrm{P}} \sum_\ell a_{\ell,\mathrm{P}}(\boldsymbol{u}^{(m)}).$$

Substitution of the last expression into the right-hand side of (10.51) gives the desired approximation of $\tilde{u}_i'$

$$\tilde{u}'_{i,\mathrm{P}} \approx -u'_{i,\mathrm{P}} \frac{\sum_\ell a_{\ell,\mathrm{P}}(\boldsymbol{u}^{(m)})}{a_\mathrm{P}(\boldsymbol{u}^{(m)})}. \tag{10.53}$$

It can be used in the velocity correction formula (10.46) and in the pressure equation (10.48), which become, respectively,

$$u'_{i,\mathrm{P}} = -\frac{1}{a_\mathrm{P}(\boldsymbol{u}^{(m)}) + \sum_\ell a_{\ell,\mathrm{P}}(\boldsymbol{u}^{(m)})} \left(\frac{\delta p'}{\delta x_i}\right)_\mathrm{P} \tag{10.54}$$

and

$$\frac{\delta}{\delta x_i}\left[\frac{1}{a_\mathrm{P}(\boldsymbol{u}^{(m)}) + \sum_\ell a_{\ell,\mathrm{P}}(\boldsymbol{u}^{(m)})} \left(\frac{\delta p'}{\delta x_i}\right)\right]_\mathrm{P} = \left(\frac{\delta u_i^*}{\delta x_i}\right)_\mathrm{P}. \tag{10.55}$$

The iteration strategy remains the same as in SIMPLE.

Another approach is to add extra corrector substeps to the SIMPLE routine. One such method called SIMPLER (SIMPLE Revised) was proposed by Patankar in 1980. Every outer iteration is implemented as follows.

Steps 1 and 2: Evaluate the coefficients and solve the approximate momentum equation, which is obtained by deleting the pressure gradient term $\left(\delta p^{(m)}/\delta x_i\right)_\mathrm{P}$ from (10.42). The result is the intermediate velocity $\hat{u}_i^{(m+1)}$ different from u_i^*.

Step 3: Solve the pressure equation

$$\frac{\delta}{\delta x_i}\left(\frac{1}{a_\mathrm{P}(\boldsymbol{u}^{(m)})} \frac{\delta p^{(m+1)}}{\delta x_i}\right)_\mathrm{P} = \left(\frac{\delta \hat{u}_i^{(m+1)}}{\delta x_i}\right)_\mathrm{P}. \tag{10.56}$$

The solution is the pressure field at the next iteration level, so no further pressure correction is necessary.

Step 4: Substitute $p^{(m+1)}$ into the momentum equation (10.42) and solve them to obtain new intermediate velocity u_i^*.

Step 5: Solve yet another pressure equation for p' identical to equation (10.49) of SIMPLE.

Step 6: Use p' to update velocity u_i^* to $u_i^{(m+1)}$ as in (10.50). Pressure is not updated.

Every iteration of SIMPLER requires roughly twice the number of operations of the original SIMPLE algorithm, since two Poisson pressure equations and two sets of momentum equations have to be solved. This disadvantage is typically outweighed by significantly faster convergence, so SIMPLER is, in general, more efficient.

The last version to be reviewed is PISO (Pressure Implicit with Splitting Operators) algorithm (Issa 1986). It is also based on the use of additional corrector steps. The first two substeps are the same as in SIMPLE. We solve the momentum equation (10.42) to find u_i^* and the Poisson pressure equation (10.49) for p'. As in SIMPLE, $\tilde{u}_i'$ is neglected at this stage. As a next step, we make the second correction, which takes into account the effect of $\tilde{u}_i'$. It uses the full correction formula (10.46) modified as

$$u_{i,\mathrm{P}}'' = \tilde{u}_{i,\mathrm{P}}' - \frac{1}{a_{\mathrm{P}}(\boldsymbol{u}^{(m)})}\left(\frac{\delta p''}{\delta x_i}\right)_{\mathrm{P}}, \tag{10.57}$$

where we can calculate $\tilde{u}_i'$ on the basis of already found u_i' using the relation (10.51). The incompressibility condition leads to the Poisson equation for the second pressure correction

$$\frac{\delta}{\delta x_i}\left(\frac{1}{a_{\mathrm{P}}(\boldsymbol{u}^{(m)})}\frac{\delta p''}{\delta x_i}\right)_{\mathrm{P}} = \left(\frac{\delta \tilde{u}_i'}{\delta x_i}\right)_{\mathrm{P}}. \tag{10.58}$$

10.5 OTHER METHODS

10.5.1 Vorticity-Streamfunction Formulation for Two-Dimensional Flows

The methods discussed so far can be applied to arbitrary incompressible flows. If the flow is two-dimensional, with velocity and pressure fields being functions of only two space coordinates and time, $\boldsymbol{V} = u(x, y, t)\boldsymbol{i} +$

$v(x,y,t)\boldsymbol{j}$, $p = p(x,y,t)$, the problem can be simplified and solved in a completely different way. The governing equations become

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{10.59}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\nabla^2 u \tag{10.60}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\nabla^2 v. \tag{10.61}$$

Formulation: The simplification is achieved if instead of *primitive variables* $\boldsymbol{V}$ and p we use *streamfunction* and *vorticity*. The streamfunction of an incompressible two-dimensional flow is a function $\psi(x,y,t)$ such that

$$u = \frac{\partial \psi}{\partial y}, \quad v = -\frac{\partial \psi}{\partial x}. \tag{10.62}$$

The name of ψ is related to the fact that the velocity vector at every point of space and every moment of time is tangential to the line $\psi = const$ passing through this point. The lines $\psi = const$ thus represent the *streamlines* of the flow.

Vorticity is a vector field (in general, three-dimensional) $\boldsymbol{\omega} = \nabla \times \boldsymbol{V}$. For a two-dimensional flow in the x-y-plane, only the z-component of vorticity is nonzero:

$$\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}. \tag{10.63}$$

An important result of using the streamfunction instead of velocity components is that the incompressibility condition (10.59) is satisfied automatically. This is easy to verify by substitution of expressions (10.62) for u and v.

We can now transform the system of governing equations (10.59)–(10.61). Applying the operator $(\nabla\times)$ to the momentum equation and taking into account the incompressibility (10.59) and the mathematical identity

$$\frac{\partial}{\partial y}\left(\frac{\partial p}{\partial x}\right) - \frac{\partial}{\partial x}\left(\frac{\partial p}{\partial y}\right) = 0,$$

we obtain the so-called *transport equation for vorticity*

$$\frac{\partial \omega}{\partial t} + u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y} = \nu\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right) \tag{10.64}$$

or, in a short form,

$$\frac{D\omega}{Dt} = \nu\nabla^2\omega. \tag{10.65}$$

The second equation is obtained by substituting expressions (10.62) for u and v into (10.63). This results in a connection between vorticity and streamfunction:

$$\nabla^2\psi = -\omega. \tag{10.66}$$

Note that the equations (10.65) and (10.66) form a coupled system. The coupling occurs via appearance of velocity components in the left-hand side of (10.65).

We have replaced the original system of three partial differential equations (10.59)–(10.61) by just two equations (10.65) and (10.66). Such a reduction in the number of equations to be solved is a remarkable result. The incompressibility condition is satisfied automatically by the velocity field. The pressure field does not explicitly appear in (10.65) and (10.66) and, in principle, is not needed for the solution. If, for some reason, knowledge of pressure field is required, p can be evaluated after the velocity field is found by solving the pressure equation (10.5).

The system (10.65)–(10.66) requires boundary conditions on ψ and ω. For the streamfunction, imposing physically plausible conditions is not difficult. One has to write the proper boundary conditions for the velocity components and use (10.62) to represent them as conditions for ψ and its derivatives. The situation is more difficult in the case of vorticity. There are no natural conditions on ω. They can, however, be derived from the conditions on ψ applying the equation (10.66) at the boundary. Since this typically results in expressions containing second derivatives, special numerical treatment is required.

Methods of Solution: The vorticity transport equation (10.65) and Poisson equation (10.66) can be discretized using the schemes already presented in previous Chapters. For example, we can apply finite difference discretization on a uniform grid and use the simple explicit scheme for (10.65):

$$\frac{\omega_{i,j}^{n+1} - \omega_{i,j}^n}{\Delta t} = -u_{i,j}^n \frac{\omega_{i+1,j}^n - \omega_{i-1,j}^n}{2\Delta x} - v_{i,j}^n \frac{\omega_{i,j+1}^n - \omega_{i,j-1}^n}{2\Delta y} + \nu\left[\frac{\omega_{i+1,j}^n - 2\omega_{i,j}^n + \omega_{i-1,j}^n}{(\Delta x)^2} + \frac{\omega_{i,j+1}^n - 2\omega_{i,j}^n + \omega_{i,j-1}^n}{(\Delta y)^2}\right]. \tag{10.67}$$

For (10.66), the standard five-point scheme of the second order gives

$$\frac{\psi_{i+1,j}^{n+1} - 2\psi_{i,j}^{n+1} + \psi_{i-1,j}^{n+1}}{(\Delta x)^2} + \frac{\psi_{i,j+1}^{n+1} - 2\psi_{i,j}^{n+1} + \psi_{i,j-1}^{n+1}}{(\Delta y)^2} = -\omega_{i,j}^{n+1}. \tag{10.68}$$

We also need discretized expressions for u and v. Keeping the second order of approximation, we use

$$u_{i,j}^{n+1} = \frac{\psi_{i,j+1}^{n+1} - \psi_{i,j-1}^{n+1}}{2\Delta y}, \quad v_{i,j}^{n+1} = -\frac{\psi_{i+1,j}^{n+1} - \psi_{i-1,j}^{n+1}}{2\Delta x}. \tag{10.69}$$

The system of coupled equations (10.67)–(10.69) can be solved by a multistep procedure conceptually similar to the projection method developed in section 10.3 for explicit methods. Each time step consists of the following substeps:

Step 1: Use the known values of velocity, vorticity, and streamfunction at $t = t^n$ to solve (10.67) and find $\omega_{i,j}^{n+1}$ at the interior points of the computational domain.

Step 2: Solve the Poisson equation (10.68) together with the boundary conditions on ψ to find $\psi_{i,j}^{n+1}$. This can be done by a direct or iterative method.

Step 3: Find values $u_{i,j}^{n+1}$ and $v_{i,j}^{n+1}$ of the velocity components at the new time layer using (10.69) and velocity boundary conditions.

Step 4: Update $\omega_{i,j}^{n+1}$ at the boundary points using $\psi_{i,j}^{n+1}$ and the discretization of the vorticity boundary conditions derived from (10.66).

The scheme (10.67)–(10.69) has the truncation error T.E. $= O[(\Delta x)^2, (\Delta y)^2, \Delta t]$ and is stable if

$$\Delta t \le \frac{1}{2\nu}\left[\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2}\right]^{-1} \quad \text{and} \quad \Delta t \le \frac{2\nu}{(u_{i,j}^2 + v_{i,j}^2)}. \tag{10.70}$$

The second of these conditions must be satisfied at all grid points (x_i, y_j). This means that the maximum of $u^2 + v^2$ must be either estimated a-priori or evaluated at every time layer with subsequent corresponding adjustment of Δt.

Of course, the algorithm just described is only one of many possible solutions. Other methods may use implicit schemes for the vorticity transport equation or a pseudo-transient representation of the Poisson equation (10.66).

If the flow is steady-state, the vorticity equation becomes

$$\boldsymbol{V} \cdot \nabla\omega + \nu\nabla^2\omega = 0. \tag{10.71}$$

We can solve the resulting system (10.66), (10.71) as an equilibrium problem. Iterative methods for systems of coupled nonlinear equations, such as the method of sequential iterations discussed in section 8.4.3, can be applied.

10.5.2 Artificial Compressibility

Yet another approach to solution of incompressible flow equations is to modify them so that the methods developed for compressible flows can be applied. The approach can be used to compute steady-state flows. Introducing fictitious time τ and adding the term $\partial \boldsymbol{V}/\partial\tau$ to the left-hand side of the momentum equation, we also replace the incompressibility condition

$$\nabla \cdot \boldsymbol{V} = 0 \tag{10.72}$$

by

$$\frac{\partial p}{\partial \tau} + a^2 \nabla \cdot \boldsymbol{V} = 0. \tag{10.73}$$

Because in the limit $\tau \to \infty$ p converges to a steady field independent of τ, (10.73) converges to (10.72). The incompressibility is, thus, satisfied for the final steady state. Equation (10.73) reminds the continuity equation in the case of isentropic compressible flows, with a playing the role of the speed of sound. For this reason, this method introduced by Chorin in 1967 bears the name of the *artificial compressibility* approach.

REFERENCES AND SUGGESTED READING

Ferziger, J. H., and M. Perić. *Computational Methods for Fluid Dynamics*, 3rd ed. New York: Springer, 2001.

Fletcher, C. A. J. *Computational Techniques for Fluid Dynamics*, *Vol. 2: Specific Techniques for Different Flow Categories*. Berlin: Springer-Verlag, 1991.

Patankar, S. V. *Numerical Heat Transfer and Fluid Flow*, 1st ed. London: Taylor & Francis, 1980.

Yanenko, N. N. *The Method of Fractional Steps*. New York: Springer-Verlag, 1971.

Caretto, L. S., A. D. Gosman, S. V. Patankar, and D. B. Spalding. "Two calculation procedures for steady, three-dimensional flows with recirculation," *Lecture Notes in Physics* **19** (1973): 60.

Chorin, A. J. "A numerical method for solving incompressible viscous flow problems," *J. Comp. Phys.* **2** (1967): 12–26.

Chorin, A. J. "Numerical solution of the Navier-Stokes equations," *Math. Comp.* **22** (1968): 745–762.

Ghia, U., K. N. Ghia, and C. T. Shin. "High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method," *J. Comp. Phys.* **48** (1982): 387–411.

Harlow, F. H., and J. E. Welsh. "Numerical calculation of time dependent viscous incompressible flow with free surface," *Phys. Fluids* **8** (1965): 2182–2189.

Issa, R. I., "Solution of implicitly discretized fluid flow equations by operator-splitting," *J. Comp. Phys.* **62** (1986): 40–65.

Morinishi, Y., T. S. Lund, O. V. Vasilyev, and P. Moin. "Fully conservative higher order finite difference schemes for incompressible flows," *J. Comp. Phys.*, **143** (1998): 90–124.

Patankar, S., & D. Spalding. "A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows," *Int. J. Heat Mass Transf.*, **15** (1972): 1787–1807.

Temam, R., "Sur l'approximation de la solution des equations de Navier-Stokes par la méthode des pas fractionnaires," *Arch. Rational Mech. Anal.* **32** and **33** (1969): 135–153; 377–385.

Van Doormal, J. P., and G. D. Raithby. "Enhancement of the SIMPLE method for predicting incompressible fluid flows," *Numer. Heat Transfer* **7** (1984): 147–163.

PROBLEMS

1. What is the difference between colocated and staggered grid arrangements? Discuss the comparative advantages of each approach.

2. If your course involves exercises with a CFD code, study the manual to determine whether the discretization uses staggered or colocated grids. Does the manual say anything about the exact mass and energy conservation when the schemes are applied to incompressible flows?

3. Derive the approximations (10.14) and (10.15) of the pressure gradient terms.

4. Describe the staggered grid arrangement of finite difference and finite volume structured grids in the three-dimensional case.
5. The simple explicit scheme and the projection method (see section 10.3.1) are applied to compute the flow of an incompressible viscous fluid in a rectangular box $0 < x < A$, $0 < y < B$, $0 < z < C$. All boundarie are solid walls. Write the boundary conditions for pressure. Why do we need them? Would we still need them if the flow were compressible?
6. Modify the predictor-corrector formulas (10.24)–(10.27) for the method that uses the second order Adams-Bashfort time integration scheme (see section 7.4.1) instead of the simple explicit scheme.
7. Show that the decomposition (10.34) of the nonlinear term is correct. Use direct substitution of (10.33) into the expression for one component of vector $\boldsymbol{N}$.
8. Consider the explicit, fully implicit, and semi-implicit methods discussed in section 10.3. Order them by the amount of computations required at every time step. Explain your answer.
9. Develop the formula (10.40) for the case of the x-momentum equation of a two-dimensional incompressible flow discretized on a structured uniform finite difference grid. Use central differences of the second order and staggered grid arrangement. Derive expressions for all the coefficients: $a_{\mathrm{P}}(\boldsymbol{u})$, $a_{\ell,\mathrm{P}}(\boldsymbol{u})$, $Q_{\mathrm{P}}(\boldsymbol{u})$.
10. If your course involves exercises with a CFD code, study the manual to determine which of the projection schemes discussed in section 10.4 (SIMPLE, SIMPLEC, SIMPLER, PISO) are implemented. Are there other schemes available for incompressible flows? Does the manual provide any recommendations concerning the choice of the scheme?

Programming Exercise The lid-driven cavity flow has long been used as a benchmark for numerical methods. The simplest version is the two-dimensional flow of an incompressible fluid in a square cavity $x < 0 < L$, $0 < y < L$. The walls at $x = 0$, $x = L$, and $y = 0$ are stationary, while the wall at $y = L$ (the lid) is moving with constant velocity U in the tangential direction. If your course involves exercises with a CFD code, calculate the flow at several values of the Reynolds number $Re = UL/\nu$. For example, try $Re = 10$, 100, and 1,000. Experiment with different projection schemes and different grid sizes. Compare your results with the results available in literature (e.g., Ghia et al. 1982).

Part III

ART OF CFD

11

TURBULENCE

11.1 INTRODUCTION

Most flows around us are turbulent. This is true for what we observe in nature, technology, and everyday life. Among the possible examples, we name, quite arbitrarily, flows in the earth atmosphere (the weather) and liquid core (this flow causes the terrestrial magnetic field by the dynamo effect), wakes behind moving bodies, such as airplanes and automobiles, a flow within a cylinder of an internal combustion engine, and a flow in a cup of coffee, to which stirring by a spoon is applied to facilitate mixing and dissolution of sugar. These flows are generated by different mechanisms and have sizes ranging from the size of a coffee cup to the size of the earth, but they have one feature in common. The Reynolds number $Re \equiv UL/\nu$ is sufficiently large, so that the flow can only exist in turbulent form. In the definition of the Reynolds number, U and L are the typical velocity and length scales and $\nu = \mu/\rho$ is the kinematic viscosity of the fluid.

11.1.1 A Few Words About Turbulence

A complete, rigorous, and short definition of turbulence seems impossible. The usual approach is to define turbulence by its following main traits:

Irregularity, Time-dependence, and Three-dimensionality: A turbulent flow field may have a component with smooth large-scale

regular structure. It is usually identified as the mean flow with velocity $\langle \boldsymbol{u} \rangle$, where brackets stand for statistical averaging or some form of time or space averaging (a more precise definition of the mean flow is given in section 11.3). In addition to the mean component, however, the flow field necessarily contains fluctuations

$$\boldsymbol{u}' = \boldsymbol{u} - \langle \boldsymbol{u} \rangle, \tag{11.1}$$

which are irregular, time-dependent, and essentially three-dimensional. Figure 11.1 shows an example of an irregular turbulent flow field, as well as the corresponding mean flow.

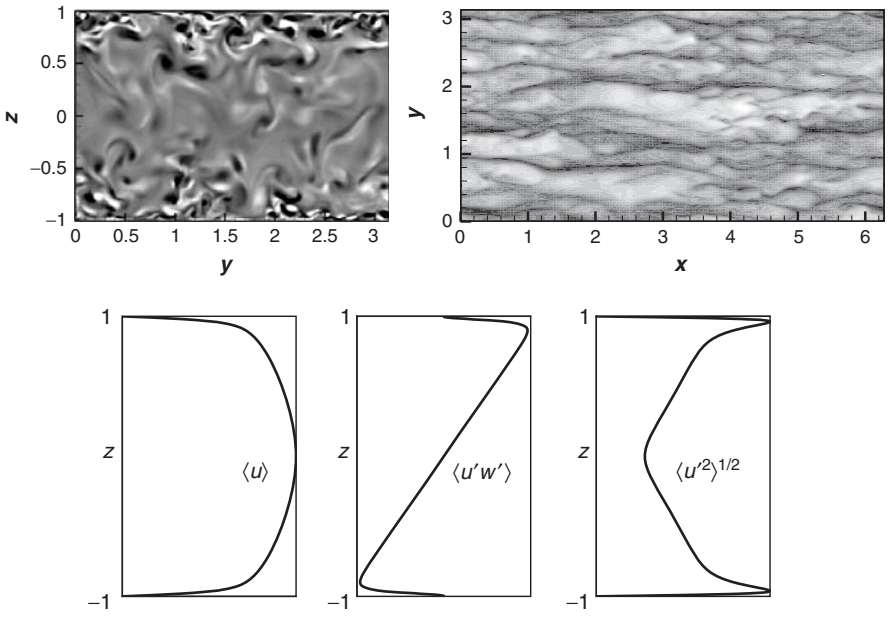


Figure 11.1 Example of a computed turbulent flow in a channel (courtesy of D. Krasnov, Ilmenau University of Technology). The Reynolds number based on the channel width and the channel-averaged velocity is 13,333. x, y, and z are the nondimensional coordinates obtained from the physical coordinates by scaling with the half of the channel width. Top from left to right: Instantaneous snapshots of the x-component of vorticity $\boldsymbol{\omega} = \nabla \times \boldsymbol{u}$ in the channel cross-section and of the x-component of velocity in the wall-parallel plane at $z = 0.95$. Bottom from left to right: profiles of horizontally and time-averaged characteristics: mean flow $\langle u \rangle$, turbulent Reynolds stress $\langle u'w' \rangle$, and root-mean-square amplitude of velocity fluctuation component $\langle u'^2 \rangle^{1/2}$.

(Pseudo-)chaoticity: A seemingly obvious conclusion can be obtained from a plain visual observation of a turbulent flow. The flow behaves chaotically.[1] The change of flow pattern in space and time seems random. However, a turbulent flow is still a solution of the Navier-Stokes equations, which do not contain any stochastic terms. The flow should follow a fully predictable evolution determined by the equations, boundary, and initial conditions. The reason for the observed pseudo-randomness is that small perturbations that are constantly added to a turbulent solution are enhanced exponentially in time. In this sense, the turbulent flows are called *constantly unstable*. Practical CFD analysis usually disregards the subtleties and treats the turbulent fluctuations as truly chaotic.

Broad Range of Length and Time Scales: A turbulent flow consists of motions with typical length and time scales that continuously fill a very broad range. This phenomenon can be explained by the nonlinearity of the Navier-Stokes equations or, from another viewpoint, by hydrodynamic instabilities and interaction between flow structures. It is formalized using the concept of energy cascade introduced by L. F. Richardson in 1922. According to the concept, large flow structures (also called *eddies* or *vortices* to stress the role of vorticity in their dynamics) constantly generated by the hydrodynamic instability of the flow are unstable themselves and generate smaller eddies. The smaller eddies are also unstable and break into even smaller ones. The kinetic energy is thus constantly transferred from large-scale to small-scales motions. The cascade stops at the level where the structures are so small that strong velocity gradients lead to complete dissipation of transferred kinetic energy into heat. The pattern is schematically represented in Figure 11.2.

The cascade concept and certain physical assumptions led A. N. Kolmogorov in 1941 to develop the phenomenological picture of turbulence, which remains one of the most profound results of the turbulence theory. Essentially for us, the Kolmogorov phenomenology also provides the basis for understanding the numerical methods used to compute turbulent flows. In particular, it allows us to estimate the ranges of active scales of the flow. The estimates are strictly valid for isotropic homogeneous turbulence, but are qualitatively correct in general case. The typical length and time scales of the smallest eddies η and τ are related to the typical length and time scales of the largest eddies as

$$\eta/L \sim Re^{-3/4}, \quad \tau/T \sim Re^{-1/2}. \tag{11.2}$$

[1]Drawings indicating attempts to make such observations were made by Leonardo da Vinci. This drawings have been heavily used by the turbulence research community in a bid to add age and respectability to the discipline (see e.g. Frisch 1995).

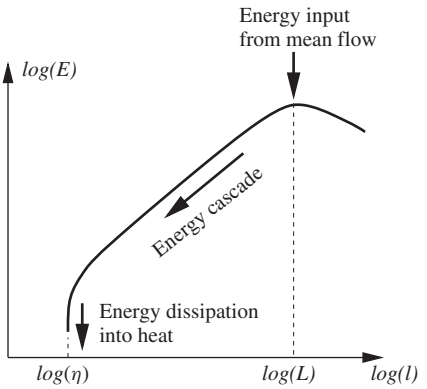


Figure 11.2 Schematic representation of the distribution of energy of velocity fluctuations over the length scales in a turbulent flow. L is the typical size of the largest and most energetic fluctuations. η is the Kolmogorov scale, the typical size of the smallest fluctuations, in which the viscous dissipation into heat primarily occurs.

The kinetic energy is distributed very unevenly over the active length scales. As illustrated in Figure 11.2, the small-scale fluctuations have much lower (orders of magnitude lower) energy than the large-scale ones.

Strong Mixing: The turbulent fluctuations provide a much more efficient mechanism of mixing than molecular diffusion. This concerns all kinds of mixing: of momentum, internal energy (in flows with heat transfer), or dissolved admixture. The actual mixing—that is, the transport of any of these fields between two neighboring fluid particles—is still accomplished by molecular diffusion. The role of turbulent fluctuations is to add intensive stirring action that brings particles with different concentrations of the mixed field into contact with each other.

Coherent Structures: One of the significant recent advances of the turbulence research is the understanding of the role played by localized *coherent structures*, which have well-defined shapes, are repeatedly generated by the flow, and persist for relatively long time. The structures have typical sizes in a wide range and take various forms, such as regions of strong vorticity or high- or low-velocity streaks (see the top two plots in Figure 11.1 for an illustration). They play an important, sometimes dominant, role in turbulent mixing. We should note that the existence of coherent structures does not annul the pseudostochastic nature of turbulence. Size, time of occurrence, location, and shape of the structures follow the pseudorandom pattern.

This description of the traits of turbulence is deliberately brief. Our intention is merely to introduce the concepts needed for the following discussion of computational methods and to draw the reader's attention to the fascinating and challenging subject of turbulence. Several of the many good books available for a thorough study of the subject are listed at the end of the chapter.

11.1.2 Why Is the Computation of Turbulent Flows Difficult?

Before we answer this question, it is necessary to say that, so far, nobody has found a way to describe and predict turbulent flows mathematically in the form of explicitly written solutions of Navier-Stokes or some other equations. A complete characterization of a turbulent flow by experimental techniques is also impossible. The situation with numerical methods seems more promising. Here, at least, we can rely on constantly growing computational power. The optimism is, unfortunately, not fully justified, as shown by the following simple analysis.

Let us estimate the size of the computational grid needed to accurately calculate a turbulent flow. It is obvious that the grid step should not be larger than the size of the smallest turbulent eddy η. If this condition is not satisfied, the fluctuations "fall through the gaps" between the grid points. They are not detected and their effect on the flow is ignored. We assume, for simplicity, that the grid step is about the same in every direction and $\Delta x \sim \Delta y \sim \Delta z \sim \eta$.

The computational domain does not have to cover the entire flow domain. Using artificial boundary conditions, we can limit computations to a fraction of the flow (see the discussion in section 2.10). However, to reproduce the flow dynamics, the computational domain has to be at least several times larger than the largest turbulent eddies. For simplicity, we assume that the dimensions of the computational domain are $L_x \sim L_y \sim L_z \sim L$. The number of the grid points in every direction is estimated using the Kolmogorov scaling (11.2) as

$$N_x \sim N_y \sim N_z \sim N = L/\eta \sim Re^{3/4}. \tag{11.3}$$

In three dimensions, the total size of the computational grid is

$$N^3 \sim Re^{9/4}. \tag{11.4}$$

This appears a stunningly large number if we consider that the Reynolds number is usually large, anywhere between 10^4 and 10^{12} or higher. Even at the lower end, the required grid consists of 10^9 points or cells.

An estimate of the required computational effort should also take into account that the time step has to be not larger than the time scale of the smallest turbulent eddies τ. The result, discussed, for example, in Pope (2000), is that even in the simplest case of an incompressible flow without boundary layers, the product of the number of grid nodes and the number of time steps needed to simulate a typical evolution of a turbulent flow is approximately $160Re^3$.

The number of floating point operations per node per time step varies, depending on the effectiveness of the method, but is unlikely to be less than 100. Assuming this lower bound, we obtain the final estimate that the total number of floating point operations needed for a meaningful simulation of a turbulent flow is, in any case, not smaller than $\sim 10^4 Re^3$. To complete the discussion, we consider an example of a moderately high Reynolds number $Re = 10^6$ (exceeded in many common flows). The total number of operations is, at least, $\sim 10^{22}$. Assuming that we work on a hypothetical multiprocessor workstation with performance of 100 Gflops (10^{11} floating point operations per second—a decent number at the moment of writing this book), the computations would take at least 10^{11} seconds, or about 3,000 years.

The example illustrates the main difficulty of numerical simulations of turbulent flows. The requirement of computational accuracy leads to unrealistically large computational grids. Even if we adopt the most optimistic predictions of the future growth of computational power, direct computations of realistic turbulent flows will remain unfeasible for a long time.

11.1.3 Overview of Numerical Approaches

There are different ways to overcome the computational challenge and acquire useful information on turbulent flows from numerical solutions. Let us overview the landscape before moving to descriptions of particular methods.

The approaches to numerical analysis of turbulence can be divided into two groups: *simulations* and *modeling*. In *simulations*, we calculate an actual realization of the flow (simulate it). The methods of this group are the direct numerical simulation (DNS) and large eddy simulations (LES) methods. The DNS, which we discuss in section 11.2, is the most honest approach. We solve the Navier-Stokes equations without any modifications or modeling assumptions. The result is a complete picture of the evolution of a time-dependent flow field $\boldsymbol{u}(\boldsymbol{x}, t)$, $p(\boldsymbol{x}, t)$. This is the approach we have assumed to follow so far in this book. The disadvantage of this approach in the case of turbulent flows is that, as discussed above, the

requirement of accurate approximation of flow features at small length scales leads to unrealistically large grids.

In the LES approach introduced in section 11.4, we solve the equations for spatially filtered variables $\overline{\boldsymbol{u}}(\boldsymbol{x},t)$, $\overline{p}(\boldsymbol{x},t)$ representing time-dependent behavior of flow features with large and moderate length scales. The effect of small-scale fluctuations appears in the form of additional terms of the equations, which cannot be calculated directly and have to be substituted by model approximations.

In *modeling*, we do not try to compute an actual realization of the flow. Instead, the problem is recast as a system of equations for mean flow quantities, such as mean velocity and pressure $\langle \boldsymbol{u} \rangle$, $\langle p \rangle$, Reynolds stresses $\langle u_i u_j \rangle$, and so on. The results correspond to our expectations of the flow characteristics that would be obtained after averaging over many realizations. This approach is called the Reynolds-averaged Navier-Stokes (RANS) method. As discussed in section 11.3, the method is computationally very efficient. However, the results are often inaccurate because of the large error introduced by the approximations included into the RANS equations.

Which of the three methods (DNS, LES, or RANS) should we use? The answer depends on the purpose of the analysis and the kind of the flow. There are several factors to consider:

Accuracy: Extra terms that appear in the LES and RANS equations have to be approximated. As we discuss in sections 11.3 and 11.4, the turbulence models used for the approximation are inherently imprecise and based on rather weak physical assumptions. They introduce the model error, whose magnitude varies depending on the type of the model, grid step, and flow characteristics. In general, $0 = \epsilon_{\text{model,DNS}} < \epsilon_{\text{model,LES}} < \epsilon_{\text{model,RANS}}$.

Level of Description: The three approaches describe the flow on different levels. In DNS, we find a complete flow field $\boldsymbol{u}(\boldsymbol{x},t)$, $p(\boldsymbol{x},t)$, which can be used to find all the desired characteristics related to behavior at all length scales. LES also produces a flow field, but the computed variables $\overline{\boldsymbol{u}}(\boldsymbol{x},t)$, $\overline{p}(\boldsymbol{x},t)$ do not provide information on motions at small scales. At last, RANS methods only produce the mean flow $\langle \boldsymbol{u} \rangle$, $\langle p \rangle$. Characteristics of turbulent fluctuations remain unknown, except for the basic properties, such as their kinetic energy and dissipation rate, which are estimated (with an error) by the turbulence models.

Computational Cost: The computational cost is highest for DNS and lowest for RANS. This is in accordance with the size of structures that

have to be resolved by a computational grid: the smallest turbulent eddies in DNS and large-scale features of mean flow in RANS. The computational cost of LES varies with the model and the desired accuracy. In general, it lies between the costs of DNS and RANS.

It is easy to see that the three methods are optimal for applications in different areas. The ability of DNS to accurately simulate the complete flow behavior makes it a powerful and irreplaceable tool for fundamental turbulence research. Of course, high computational cost means that DNS analysis is limited to flows with simplified domains and small-to-moderate Reynolds numbers. The majority of existing flows are beyond reach for DNS.

The practical engineering analysis is traditionally conducted using RANS models. The advantages are obvious. Mean flow characteristics are often sufficient for engineering problems. The computations can be conducted in a few hours or even minutes. Unfortunately, RANS provides no data on turbulent fluctuations and predicts the mean flow with significant error.

The LES approach occupies an intermediate position between DNS and RANS. It can be used for fundamental science, albeit with due awareness of the model error it introduces. The trend of modern CFD development is the growing role of LES in practical engineering computations. Although more expensive to conduct than RANS (this problem is being gradually taken care of by increasing computer power), LES analysis adds important, sometimes crucial, information on large- and moderate-scale fluctuations and is more accurate than RANS in predicting mean flow properties.

11.2 DIRECT NUMERICAL SIMULATION (DNS)

In principle, any numerical method—for example, finite difference or finite volume—can be used in DNS. The most popular choice is, however, the spectral methods (see section 3.3.1). The reasons for that become clear when we recall that spectral methods, while limited to domains of simple geometric shapes, such as a rectangular box, a cylinder, or a sphere, are computationally very efficient. DNS are typically conducted in studies of fundamental turbulence properties, where the choice of simple flow domains is allowed and even commendable.

11.2.1 Homogeneous Turbulence

The first DNS of a flow with realistic turbulent features appeared in the paper by Orszag and Patterson in 1972. The numerical approach pioneered

in that paper still serves, with some modifications, as an important tool of fundamental turbulence research. The method is based on the assumption that turbulent fluctuations are *spatially homogeneous*, which means that their statistically averaged properties are the same at every point of the flow domain. The assumption excludes flows with solid walls and other realistic boundaries but still leaves plenty of interesting behavior to look at. The situation nearest to homogeneous is achieved in wind tunnel experiments, where turbulence is generated by a uniform grid perpendicular to the mean flow direction. As an approximation, the state of turbulence in many other flows can be considered homogeneous, if we consider a small zone far from the walls.

In DNS, a homogeneous turbulent flow is simulated by a flow in a rectangular box of dimensions $L_x \times L_y \times L_z$ with periodic (cyclic) boundary conditions in all three directions:

$$\boldsymbol{u}(\boldsymbol{x},t) = \boldsymbol{u}(\boldsymbol{x} + L_x\boldsymbol{e}_x, t) = \boldsymbol{u}(\boldsymbol{x} + L_y\boldsymbol{e}_y, t) = \boldsymbol{u}(\boldsymbol{x} + L_z\boldsymbol{e}_z, t).$$

The periodicity allows us to apply the efficient Fourier spectral method introduced by Orszag and Patterson and later developed by others, most notably by Rogallo in 1981. The flow fields are approximated by three-dimensional Fourier series

$$\boldsymbol{u}(\boldsymbol{x},t) = \sum_{\boldsymbol{k}} \hat{\boldsymbol{u}}(\boldsymbol{k},t) e^{\imath \boldsymbol{k}\cdot\boldsymbol{x}}, \tag{11.5}$$

where $\boldsymbol{k} = k_x\boldsymbol{e}_x + k_y\boldsymbol{e}_y + k_z\boldsymbol{e}_z$ are the wavenumber vectors and $\hat{\boldsymbol{u}}(\boldsymbol{k},t)$ are complex-valued expansion coefficients. The exponential term is the shorthand notation $e^{\imath \boldsymbol{k}\cdot\boldsymbol{x}} \equiv e^{\imath k_x x} e^{\imath k_y y} e^{\imath k_z z}$, where, for example, $e^{\imath k_x x} = \cos(k_x x) + \imath \sin(k_x x)$.

The wavenumbers are related to the dimensions of the box, which are also periodicity lengths, as

$$\boldsymbol{k}_x = \frac{2\pi m_x}{L_x}, \quad \boldsymbol{k}_y = \frac{2\pi m_y}{L_y}, \quad \boldsymbol{k}_z = \frac{2\pi m_z}{L_z},$$

where $m_x = -N_x/2, \ldots, N_x/2$, $m_y = -N_y/2, \ldots, N_y/2$, and $m_z = -N_z/2, \ldots, N_z/2$ are integer indices. Since $\boldsymbol{u}(\boldsymbol{x},t)$ is real, the complex Fourier expansion coefficients must satisfy the conjugate symmetry $\hat{\boldsymbol{u}}(\boldsymbol{k},t) = \hat{\boldsymbol{u}}^*(-\boldsymbol{k},t)$.

An important and nontrivial question concerns the values of N_x, N_y and N_z. How many terms of the expansion (11.5) should we use in a simulation? One necessary condition is that (11.5) accurately reproduces

the turbulent fluctuations of the smallest length scale, which is the Kolmogorov scale η. The numerical resolution of a spectral method based on (11.5) approximately corresponds to the resolution on a grid with steps

$$\Delta x \sim \frac{L_x}{N_x} = \frac{\pi}{k_{x,\max}}, \quad \Delta y \sim \frac{L_y}{N_y} = \frac{\pi}{k_{y,\max}}, \quad \Delta z \sim \frac{L_z}{N_z} = \frac{\pi}{k_{z,\max}},$$

where $k_{x,\max} = 2\pi\,(N_x/2)\,/L_x = \pi N_x/L_x$, etc. are the maximum wavenumbers in the expansion. The small-scale resolution requirement is $\Delta x \sim \Delta y \sim \Delta z \sim \eta$, which implies $k_{x,\max} \sim k_{y,\max} \sim k_{z,\max} \sim \pi/\eta$. Numerical studies have shown that this formula is a slight overestimation. Unless the dynamics of turbulent fluctuations with the length scales approaching η is of special interest, the resolution requirement can be relaxed to

$$k_{x,\max} \sim k_{y,\max} \sim k_{z,\max} \sim \frac{1.5}{\eta}$$

There is another numerical resolution requirement that is related to the fact that the periodic box is an artificial construction. The periodicity introduces unphysical perfect correlations over the distances L_x, L_y, and L_z. Meaningful flow dynamics can, therefore, only be observed if the box dimensions are significantly (at least seven to eight times) larger than the size of the largest turbulent flow structures.

Combining the two requirements and using the Kolmogorov scaling $L/\eta \sim Re^{3/4}$, we see that the numerical resolution is fully determined by the Reynolds number. The first periodic box DNS were conducted by Orszag and Patterson in 1972 using $N_x = N_y = N_z = 32$ for a flow with a very low Reynolds number. At the moment of writing this book, the most ambitious simulations have used $N_x = N_y = N_z = 4{,}096$ (see Ishihara et al. 2009) and achieved the Reynolds numbers comparable to those in high-Re laboratory experiments.

A peculiar aspect of the periodic box turbulence is that the flow does not have a mechanism to sustain itself. The energy is constantly transferred into small-scale structures and dissipated into heat, so the flow experiences decay. The decay is a legitimate and interesting situation, which is similar to the situation in wind tunnel experiments, except that the distance from the grid in a wind tunnel plays the role of time in DNS. Alternatively, we can maintain the flow in a statistically steady state by adding energy to large-scale motions. This can be done in a purely artificial way (the artificial forcing) or by imitating a natural mechanism, for example, by imposing a mean flow with constant shear.

The implementation of the Fourier spectral method in the case of a periodic box flow is not difficult and, in its main features, is similar to the procedure outlined in section 3.3.1 for a one-dimensional model equation. We substitute the expansion (11.5) and the similar expansion for pressure into the Navier-Stokes equations and derive a system of ordinary differential equations for the Fourier coefficients $\hat{\boldsymbol{u}}(\boldsymbol{k}, t)$. The equations are solved using one of the established time integration schemes, for example, the Runge-Kutta or Adams-Bashfort method.

The main difficulty and computational challenge of the procedure are due to the nonlinearity of the momentum equation. To see the problem, let us consider one product in the nonlinear term, for example, uv. After substituting the Fourier expansions for u and v, we obtain

$$\sum_{\boldsymbol{k}_1} \sum_{\boldsymbol{k}_2} \hat{u}(\boldsymbol{k}_1, t)\hat{v}(\boldsymbol{k}_2, t) e^{\imath\left(\boldsymbol{k}_1 \cdot \boldsymbol{x} + \boldsymbol{k}_2 \cdot \boldsymbol{x}\right)}, \tag{11.6}$$

which should be rewritten as a series

$$\sum_{\boldsymbol{k}} \hat{f}(\boldsymbol{k}, t) e^{\imath \boldsymbol{k} \cdot \boldsymbol{x}},$$

with $\hat{f}$ being expressed in terms of $\hat{u}$ and $\hat{v}$. This can be done by direct calculation of the convolution sums in (11.6), but at the cost of approximately N^6 operations, where N is an estimate of N_x, N_y and N_z. Much more efficient way is to compute the sums (11.5) for u and v at certain grid points, calculate the product uv at these points, and use the resulting field to compute its Fourier coefficients $\hat{f}(\boldsymbol{k}, t)$. With appropriate choice of grid points, the evaluation of grid point values and Fourier coefficients can be performed using direct and inverse Fast Fourier Transforms. Each transform requires approximately $N^3 \log N$ operations. The linear part of the equations requires $\sim N^3$ operations per time step. The total number of operations per time step is, thus, estimated as $\sim N^3 \log N$.

There are other nontrivial aspects of the procedure, which we do not consider here. An interested reader can find a detailed description of the method in research literature, for example, in the papers listed at the end of this chapter.

11.2.2 Inhomogeneous Turbulence

If turbulence is inhomogeneous in one or several directions—for example, if the flow domain has realistic boundaries, such as solid walls—the spectral method based on the three-dimensional Fourier expansion cannot be

used. We can apply finite difference or finite volume methods. High-order schemes are desirable in this case, since strong numerical dissipation of low-order schemes leads to unphysical suppression of small-scale fluctuations.

Another approach is possible if the flow is inhomogeneous in only one or two directions and homogeneous in the others. A classical example is the fully developed turbulent flow in a channel (see Figure 11.1). We can assume homogeneity and apply periodic boundary conditions in the streamwise (x) and spanwise (y) directions. This means that the Fourier spectral scheme can be used for discretization in x and y. The Fourier expansion becomes

$$\boldsymbol{u}(x,y,z,t) = \sum_{k_x,k_y} \hat{\boldsymbol{u}}\left(k_x,k_y,z,t\right) e^{\imath\left(k_x x + k_y y\right)}. \tag{11.7}$$

Upon substitution into the Navier-Stokes equations and performing the standard spectral method transformations in the x- and y-coordinates, we obtain a system of partial differential equations for coefficients $\hat{\boldsymbol{u}}\left(k_x,k_y,z,t\right)$ considered as functions of z and t. The discretization in the inhomogeneous (z) direction can be achieved using a finite difference scheme with proper grid clustering near the walls (we discuss clustering in Chapter 12) or by a different version of spectral method. The expansion over Chebyshev polynomials $T_{m_z}(z) = \cos(m_z \arccos z)$ is particularly convenient for the latter, since the polynomials satisfy the no-slip boundary conditions at the walls, provide good resolution of boundary layers, and their series can be computed and inverted by a Fast Fourier Transform.

The first high-resolution DNS of the channel flow published in 1987 by Kim, Moin, and Moser used the Fourier-Chebyshev spectral method. The results have had strong and lasting impact on the turbulence research and CFD in general by creating a complete and reliable set of flow characteristics. The flow presented in Figure 11.1 were also computed using a version of this method. The number of the expansion terms in the series was $N_x = N_y = 512$ and $N_z = 256$.

11.3 REYNOLDS-AVERAGED NAVIER-STOKES (RANS) MODELS

RANS is the oldest method of turbulence modeling, which still remains a primary tool of practical CFD analysis. Its advantages are simplicity, low computational cost (relative to DNS and LES), a broad selection of models

readily available in general purpose CFD codes, and significant experience accumulated in application to different kinds of turbulent flows. Its disadvantages are the low level of description (no information is available beyond the mean flow characteristics), necessity to fine-tune the models to specific features of the flow, and relatively large modeling error. The first disadvantage is often acceptable in engineering analysis. The second and third, however, should be taken very seriously. Two questions should always be asked when applying a RANS model: *How well does the model capture the flow physics, and how accurate are the quantitative predictions obtained in the analysis?*

We begin with a precise definition of the mean flow fields. The universally applicable definition is the *ensemble-averaged* field

$$\langle \boldsymbol{u} \rangle(\boldsymbol{x}, t) = \lim_{M \to \infty} \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{u}^{(m)}(\boldsymbol{x}, t), \tag{11.8}$$

where $\boldsymbol{u}^{(m)}$ are the realizations of the flow in M identical experiments.

If the flow conditions are time-independent, the mean flow can be considered a result of time averaging:

$$\langle \boldsymbol{u} \rangle(\boldsymbol{x}) = \lim_{T \to \infty} \frac{1}{T} \int_{t}^{t+T} \boldsymbol{u}^{(m)}(\boldsymbol{x}, t). \tag{11.9}$$

If the flow experiences slow nonturbulent variations, but the typical variation time T_V is much large than the typical time scale T_t of the largest and slowest turbulent fluctuations, the time averaging definition remains valid in the form

$$\langle \boldsymbol{u} \rangle(\boldsymbol{x}, t) = \frac{1}{T} \int_{t}^{t+T} \boldsymbol{u}^{(m)}(\boldsymbol{x}, t), \tag{11.10}$$

where $T_t \ll T \ll T_V$.

The operations (11.8)–(11.10) are linear and commute with space derivatives. They also commute with time derivative, which is obvious for (11.8)–(11.9) and can be derived for (11.10) as an approximation that assumes T as the new infinitesimal time step. Other relevant properties include

$$\langle \langle f \rangle \rangle = \langle f \rangle, \quad \langle f \langle g \rangle \rangle = \langle f \rangle \langle g \rangle, \tag{11.11}$$

where f and g are arbitrary functions.

The flow fields can be seen as sums of mean and fluctuating parts:

$$\boldsymbol{u}(\boldsymbol{x},t) = \langle \boldsymbol{u} \rangle(\boldsymbol{x},t) + \boldsymbol{u}'(\boldsymbol{x},t), \quad p(\boldsymbol{x},t) = \langle p \rangle(\boldsymbol{x},t) + p'(\boldsymbol{x},t). \quad (11.12)$$

Taking the mean of the left-hand and right-hand sides of this formula immediately shows that

$$\langle \boldsymbol{u}' \rangle = 0, \quad \langle p' \rangle = 0. \quad (11.13)$$

11.3.1 Reynolds-Averaged Equations

Applying the averaging operation to the Navier-Stokes system results in the *Reynolds-averaged* equations for the mean fields. We will show the derivation for the incompressible and Newtonian flow equations in conservation form:

$$\rho \frac{\partial u_i}{\partial t} + \rho \frac{\partial}{\partial x_j}\left(u_i u_j\right) = -\frac{\partial p}{\partial x_i} + \mu \nabla^2 u_i, \quad \frac{\partial u_i}{\partial x_i} = 0, \quad (11.14)$$

where, as usual, we assume summation over repeating indices. The result of averaging is

$$\rho \frac{\partial \langle u_i \rangle}{\partial t} + \rho \frac{\partial}{\partial x_j} \langle u_i u_j \rangle = -\frac{\partial \langle p \rangle}{\partial x_i} + \mu \nabla^2 \langle u_i \rangle, \quad \frac{\partial \langle u_i \rangle}{\partial x_i} = 0. \quad (11.15)$$

Using the linearity and properties (11.11) of the averaging operation, we find that

$$\rho \langle u_i u_j \rangle = \rho \langle (\langle u_i \rangle + u_i')(\langle u_j \rangle + u_j') \rangle = \rho \langle u_i \rangle \langle u_j \rangle + \rho \langle u_i' u_j' \rangle. \quad (11.16)$$

The second term in the right-hand side is the *Reynolds stress tensor*

$$\tau_{ij} \equiv \rho \langle u_i' u_j' \rangle = \rho \langle u_i u_j \rangle - \rho \langle u_i \rangle \langle u_j \rangle. \quad (11.17)$$

Using the new notation, we rewrite the RANS (Reynolds-averaged Navier-Stokes) equations (11.15) as

$$\rho \frac{\partial \langle u_i \rangle}{\partial t} + \rho \frac{\partial}{\partial x_j}\left(\langle u_i \rangle \langle u_j \rangle\right) = -\frac{\partial \langle p \rangle}{\partial x_i} + \mu \nabla^2 \langle u_i \rangle - \frac{\partial \tau_{ij}}{\partial x_j}, \quad \frac{\partial \langle u_i \rangle}{\partial x_i} = 0. \quad (11.18)$$

The system is not closed because the components of the Reynolds stress tensor are unknown and cannot be expressed as functions of $\langle \boldsymbol{u} \rangle$ and $\langle p \rangle$. There are more variables than equations. The system can only be solved numerically after we find a way to approximate τ_{ij} in terms of the mean flow quantities. The closure models providing such approximations are discussed in the rest of this section.

11.3.2 Eddy Viscosity Hypothesis

There is an apparent similarity between the momentum transport by molecular viscosity and by turbulent fluctuations. One can hypothesize that the similarity also exists on the level of functional relations and assume that the turbulent transport depends on the mean velocity gradients in the same way as molecular transport depends on the gradients of the full velocity field.[2] This is formalized as the *eddy viscosity hypothesis*. The Reynolds stress tensor is assumed to satisfy

$$\tau_{ij} \equiv \rho \langle u_i' u_j' \rangle = -2\mu_t \langle S_{ij} \rangle + \frac{2}{3}\rho \delta_{ij} k, \tag{11.19}$$

where

$$\langle S_{ij} \rangle = \frac{1}{2}\left(\frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial \langle u_j \rangle}{\partial x_i} \right)$$

is the rate of strain tensor of the mean flow, $\mu_t(\boldsymbol{x}, t)$ is the *eddy viscosity*, and

$$k \equiv \frac{1}{2}\langle u_i' u_i' \rangle = \frac{1}{2}\langle u_x' u_x' + u_y' u_y' + u_z' u_z' \rangle \tag{11.20}$$

is the kinetic energy of turbulent fluctuations (the turbulent kinetic energy).

The justification of the eddy viscosity hypothesis is questionable at best. There is no theoretical explanation beyond the generally unfounded analogy between the motion of molecules and motion of turbulent eddies. Experiments and DNS studies show that the alignment between the tensors in the left-hand and right-hand sides of (11.19) is good in simple parallel shear flows but disappears in flows of more complex structure. Nevertheless, the eddy viscosity model is widely applied in CFD analysis. The main reasons seem to be tradition, model's simplicity, and the fact that reasonably accurate results can be obtained with appropriately defined $\mu_t(\boldsymbol{x}, t)$.

On the simplest level of description, turbulence is fully characterized by kinetic energy of fluctuations k or their root-mean-square velocity

[2]The hypothesis has two components. One is the assumption of local equilibrium, according to which the local production and dissipation of turbulence are balanced, and the stress tensor depends solely on the local turbulence conditions. Another component is the actual functional form (11.19) of the dependency.

$q \equiv (2k/3)^{1/2}$ and by a certain typical length scale ℓ. Dimensionality analysis shows that in this case

$$\mu_t = C_\mu \rho q \ell, \tag{11.21}$$

where C_μ is a dimensionless proportionality constant.

11.3.3 Algebraic Models

The simplest closure models are the algebraic models, in which the turbulent eddy viscosity is approximated by an algebraic function of space, time, and simple characteristics of the mean flow. We will consider the models based on the Prandtl's mixing length theory.[3] Such models were successfully applied to parallel shear flows: attached boundary layers, mixing layers, jets, and wakes. In these flows, there is one dominant mean velocity component $\langle u \rangle$, which depends on only the cross-flow coordinate y. It can be hypothesized that the length scale ℓ and velocity scale q of (11.21) are estimated as

$$\ell \approx \ell_m, \quad q \approx \ell_m \left| \frac{d\langle u \rangle}{dy} \right|, \tag{11.22}$$

where ℓ_m is the Prandtl mixing length. Substituting into (11.21) and neglecting the proportionality constant, we obtain

$$\mu_t = \rho \ell_m^2 \left| \frac{d\langle u \rangle}{dy} \right|. \tag{11.23}$$

This expression for the eddy viscosity can be used in (11.19). The turbulent kinetic energy $k = 3q^2/2$ can be estimated as in (11.22) or, for an incompressible fluid, included into the modified pressure field. To fully close the RANS equations, however, we need an approximation of the mixing length ℓ_m. Such approximations were proposed and fine-tuned for various types of parallel shear flows. For example, for a wake, mixing layer, or jet, the approximation is $\ell_m \approx \alpha \delta(x)$, where x is the coordinate in the direction of the flow, $\delta(x)$ is the flow width, and α is the nondimensional adjustment coefficient, which takes a special value in each case. In the attached boundary layer, the mixing length can be approximated

[3]The mixing length theory was proposed by L. Prandtl in 1925. The theory introduces the so-called mixing length ℓ_m, an analog of the mean free-path in the kinetic theory of gases. The mixing length can be viewed as the typical maximum distance over which a lump of fluid particles moving in a turbulent flow retains its momentum.

on the basis of properties of the logarithmic layer, which we discuss in section 11.3.5, as $\ell_m \approx \kappa y$, where $\kappa = 0.41$ is the von Karman constant and y is the wall-normal coordinate.

There exist advanced versions of the mixing length model, which use more accurate approximations of ℓ_m (see, e.g. Wilcox 2006). The model can also be nominally extended to the general three-dimensional flow configuration. For example, the version proposed by Smagorinsky (1963) is

$$\mu_t = \rho \ell_m^2 \left(2\langle S_{ij}\rangle \langle S_{ij}\rangle\right)^{1/2}.$$

Such extended models reduce to models of the form (11.23) in parallel shear flows.

The algebraic models are very attractive because of their simplicity and computational efficiency. Their applicability, however, is limited to parallel shear flows. In all other cases, acceptably accurate prediction of the mixing length ℓ_m and, thus, of the eddy viscosity is impossible.

11.3.4 Two-Equation Models

In the two-equation models, the velocity and length scales of turbulence (q and ℓ of the eddy viscosity expression (11.21)) are determined as solutions of two additional partial differential equations. The advantage of this approach in comparison to the algebraic models is that no specification of ℓ_m is required as a precondition. The velocity and length scales are evaluated as functions of space and time on the basis of the local flow state. As a result, the two-equation models are applicable, at least theoretically, to any flow configuration.

We will consider the most commonly used $k - \epsilon$ model. The equations of this model are the transport equations for the turbulent kinetic energy $k(\boldsymbol{x}, t)$ and the rate of viscous dissipation (the rate at which the kinetic energy of small-scale fluctuations is converted into heat by viscous friction, see Figure 11.2) $\epsilon(\boldsymbol{x}, t)$. The velocity scale is defined as

$$q = k^{1/2}. \tag{11.24}$$

To determine the length scale we employ properties of turbulent energy cascade. The kinetic energy of fluctuations with characteristic length scale ℓ is related to this scale and to the total rate of dissipation as

$$\epsilon \approx \frac{k^{3/2}}{\ell}. \tag{11.25}$$

In the model, we apply this relation at the length scale of the largest fluctuations, which are known to contain a dominant part of turbulent kinetic energy. Assuming that ℓ gives the length scale of the eddy viscosity formula, we obtain

$$\mu_t = C_\mu \rho \frac{k^2}{\epsilon}. \tag{11.26}$$

Starting with the momentum conservation equation, we can derive the equation for k:

$$\rho \frac{\partial k}{\partial t} + \rho \langle u_j \rangle \frac{\partial k}{\partial x_j} = -\tau_{ij} \frac{\partial \langle u_i \rangle}{\partial x_j} - \rho\epsilon + \frac{\partial}{\partial x_j}\left[\mu \frac{\partial k}{\partial x_j}\right] - \frac{\partial}{\partial x_j}\left[\frac{\rho}{2}\langle u_i' u_i' u_j' \rangle + \langle p' u_j' \rangle\right]. \tag{11.27}$$

The terms in the left-hand side form the material derivative of k, which is the rate of change of k in a fluid particle transported by the mean flow. The terms in the right-hand side represent the mechanisms, by which the energy within a particle can be changed. The first term is the rate of energy production (the rate, at which energy is transferred to fluctuations from the mean flow). It can be approximated using the eddy viscosity hypothesis (11.19)

$$P_k \equiv -\tau_{ij} \frac{\partial \langle u_i \rangle}{\partial x_j} = -\rho \langle u_i' u_j' \rangle \frac{\partial \langle u_i \rangle}{\partial x_j} = 2\mu_t \langle S_{ij} \rangle \frac{\partial \langle u_i \rangle}{\partial x_j}. \tag{11.28}$$

Note that the second term in the right-hand side of (11.19) disappears in incompressible flow, since summation over i and j gives $\delta_{ij}(\partial \langle u_i \rangle / \partial x_j) = \partial \langle u_i \rangle / \partial x_i = 0$.

The second term in the right-hand side of (11.27) represents the effect of viscous dissipation at small scales. The rate of dissipation ϵ is found as a solution of the second partial differential equation (see (11.31)). The third term corresponds to molecular diffusion of k. It does not need any approximation. The last term also represents diffusion, but by turbulent motions. It evidently needs an approximation, the common choice of which is the gradient diffusion model:

$$\frac{\rho}{2}\langle u_i' u_i' u_j' \rangle + \langle p' u_j' \rangle \approx -\frac{\mu_t}{\sigma_k} \frac{\partial k}{\partial x_j}, \tag{11.29}$$

where σ_k is the new parameter, which is called the turbulent Prandtl number and usually assumed to be unity.

The final form of the k-equation is

$$\rho\frac{\partial k}{\partial t}+\rho\langle u_j\rangle\frac{\partial k}{\partial x_j}=2\mu_t\langle S_{ij}\rangle\frac{\partial\langle u_i\rangle}{\partial x_j}-\rho\epsilon+\frac{\partial}{\partial x_j}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right]. \tag{11.30}$$

The equation for the dissipation rate ϵ is also derived from the Navier-Stokes equation and then transformed in a series of drastic simplifying assumptions. The assumptions are, in fact, strong and unfounded to the degree that the final equation can be considered only loosely connected to the dynamics of averaged fields determined by the Navier-Stokes equations. Anyway, the final equation for ϵ is

$$\rho\frac{\partial \epsilon}{\partial t}+\rho\langle u_j\rangle\frac{\partial \epsilon}{\partial x_j}=C_{\epsilon 1}P_k\frac{\epsilon}{k}-C_{\epsilon 2}\rho\frac{\epsilon^2}{k}+\frac{\partial}{\partial x_j}\left[\left(\mu+\frac{\mu_t}{\sigma_\epsilon}\right)\frac{\partial \epsilon}{\partial x_j}\right], \tag{11.31}$$

where P_k is the rate of turbulent kinetic energy production determined by (11.28) and $C_{\epsilon 1}$, $C_{\epsilon 2}$, and σ_ϵ are model constants.

Let us summarize. The $k-\epsilon$ model consists of two partial differential equations (11.30) and (11.31) and the algebraic expression (11.26) for the eddy viscosity. The equations for k and ϵ have to be solved simultaneously with the momentum and mass conservation equations for the mean flow (11.18) as parts of one PDE system. The model contains five constant parameters, which have little theoretical footing and have to be determined through comparison with DNS and experimental results in the process of fine-tuning the model. The most commonly used set of values is:

$$C_\mu=0.09,\ C_{\epsilon 1}=1.44,\ C_{\epsilon 2}=1.92,\ \sigma_k=1.0,\ \sigma_\epsilon=1.3. \tag{11.32}$$

Numerous other two-equation models have been proposed and tested during the long history of RANS development. Many of them use the k-equation to determine the velocity scale, while the choices of the second equations and of the modeling approximations of the equation terms vary. The $k-\epsilon$ model remains most broadly used. The second place in the popularity contest belongs to the $k-\omega$ model, in which an equation for the field $\omega=\epsilon/k$ is used instead of the ϵ-equation. A detailed description of this and other models, as well as an excellent in-depth discussion of various aspects of RANS modeling can be found in the book of Wilcox, reference to which is provided at the end of the chapter.

11.3.5 Numerical Implementation of RANS Models

Since numerical implementation of algebraic models is trivial, we will focus on models that require solution of additional PDEs. The $k - \epsilon$ model will serve as an example.

The RANS equations, such as (11.30) and (11.31), are solved on the same computational grid as the mean flow. The numerical methods presented in the previous chapters of the book are applied directly to the extended system. For example, a steady-state problem for incompressible fluid is likely to be solved using a projection algorithm, iteration approach, linearization, and either finite volume or finite difference method (see Chapters 4, 5, 8, and 10). Several modifications are recommended in the RANS case to make the solution more efficient and robust.

One modification is related to the fact that turbulence has shorter response time than the mean flow. This means that the extended PDE system is numerically stiff (parts of the solution evolve at strongly different time scales), which usually leads to slow convergence of an iteration procedure. The convergence can be accelerated if the iterations for the mean flow and turbulence properties are separated in the manner similar to the sequential iterations discussed in section 8.4.3. Every iteration is divided into two substeps. On the first, an outer iteration for the mean flow is performed with the eddy viscosity taken from the previous iteration. On the second, the obtained approximation of the mean flow is used to conduct an outer iteration of the k and ϵ equations.

An important fact to be kept in mind is that, by definition, k and ϵ cannot be negative. Violation of this condition may lead to a bizarre situation, in which the eddy viscosity and the turbulent stress tensor have the wrong signs and the work of turbulent stresses $\tau_{ij} \langle S_{ij} \rangle$ acts as a source of energy for the mean flow instead of a sink. Even if this occurs locally and at an intermediate stage of the solution, for example after an intermediate iteration, there is a danger of numerical instability. We should take into account that the small response time of turbulence properties makes an overshoot into negative values of k and ϵ a likely event. The natural and efficient way to avoid the troubles is to apply successive underrelaxation (see section 8.3.4). The relaxation parameter ω about 0.6 to 0.8 is typically recommended.

The RANS equations require boundary conditions for computed turbulent fields, such as k and ϵ. Setting them and arranging near-wall treatment involve interesting and not always easy questions. We will briefly discuss some of them.

The first question concerns the actual formulation of the boundary conditions. At solid walls, we require

$$k_{\text{wall}} = 0. \tag{11.33}$$

The dissipation rate does not have to be zero. Exact boundary conditions on ϵ are impossible in the RANS framework, but we can use approximations such as, for example,

$$\epsilon_{\text{wall}} = \nu \left(\frac{\partial^2 k}{\partial n^2} \right)_{\text{wall}} \quad \text{or} \quad \epsilon_{\text{wall}} = 2\nu \left(\frac{\partial k^{1/2}}{\partial n} \right)^2_{\text{wall}}. \tag{11.34}$$

The conditions at symmetry and periodic boundaries and at exits are typically set in the same way as for other flow variables (see section 2.10). The situation is more difficult at the inlets, where we often do not know the state of the flow. The common approach is to assume a certain level of turbulence and prescribe the turbulent kinetic energy as a fraction of the kinetic energy of the mean flow. This is done in terms of *turbulence intensity* defined as the ratio between the root-mean square velocity of turbulent fluctuations and the absolute value of mean velocity:

$$I \equiv \frac{(2k/3)^{1/2}}{\left(\langle u_x \rangle^2 + \langle u_y \rangle^2 + \langle u_z \rangle^2 \right)^{1/2}}. \tag{11.35}$$

Recommended values of I vary, depending on the specific situation. $I \approx 10^{-2}$ is used for weakly turbulent inlets—for example, in computations of flows past moving bodies. Strongly turbulent inlets, such as the inlets into segments of heat exchangers or turbomachinery, require $I \approx 10^{-1}$ or even higher. To determine the inlet values of ϵ, we apply the relation (11.25). A plausible assumption is used to estimate the turbulence length scale ℓ, typically as a fraction of the inlet width.

A comment should be made about the initial conditions applied in a marching problem or initial guess of the iteration procedure in a steady-state problem. The general advice to choose the initial fields as close to the actual state of the flow as possible fully pertains to the turbulence fields k and ϵ. There is another important rule. The initial values of k and ϵ should never be zero. The danger is easy to see if we inspect the right-hand sides of the eddy viscosity equation (11.26) and k and ϵ equations (11.30) and (11.31). Taking $k = \epsilon = 0$ leads to $\mu_t = 0$ and to zero right-hand sides of (11.30) and (11.31). k and ϵ would remain zero. No turbulence would be

generated in the flow. This is a reflection of the fact that, unlike nature, the RANS models are unable to produce turbulence from a laminar high-*Re* flow. Moreover, calculations starting from such initial conditions would, most likely, lead to numerical instability.

On first glance, there should be no special resolution requirements in RANS computations. After all, we only compute averaged turbulence properties, which have approximately the same typical length scale as the mean flow. Certain requirements are, however, imposed by the behavior of turbulence near solid walls. As illustrated in Figure 11.1, amplitudes of turbulent fluctuations have strong and narrow peaks near the wall. Mean flow velocity has strong gradient in the immediate vicinity of the wall, in the so-called viscous sublayer. The typical length scale of the flow characteristics computed by a RANS model, therefore, decreases near the walls. This requires reduction of the grid steps, especially the steps in the wall-normal direction.

There is another feature of near-wall turbulence that requires special treatment. The actual turbulent eddy viscosity defined by (11.19) becomes smaller as the wall is approached. This phenomenon is poorly reproduced by the RANS models. Even at an adequate numerical resolution, use of an unmodified RANS model near the wall would result in overprediction of turbulent stresses. The error would not be limited to the near-wall zone. Inaccurate estimate of the near-wall flux of momentum would inevitably lead to an incorrect picture of the entire flow. Various algorithms have been developed to deal with this so-called low Reynolds number effect[4] by appropriately reducing the turbulent stresses in the near-wall zone. A review and discussion can be found in Wilcox (2006).

In high Reynolds number flows, the viscous sublayer is so thin that it becomes inefficient or even unfeasible to resolve it by a fine grid. An alternative approach can be used, in which computations are performed on a relatively coarse grid, and turbulence modeling is only applied up to a certain distance to the wall. Near the wall, the solution is "patched up" by the so-called *wall functions*, which imply universal boundary layer behavior and imitate the effect of a solution corresponding to that behavior.

The wall functions are based on two assumptions: the flow is in a local equilibrium, so turbulence production and dissipation are nearly equal, and the wall-parallel mean velocity satisfies the logarithmic law

$$\langle u \rangle_{\mathrm{P}} = u_\tau \left[\frac{1}{\kappa} \ln \left(\frac{u_\tau z_{\mathrm{P}}}{\nu} \right) + B \right]. \tag{11.36}$$

[4]The term refers to the fact that near a wall the behavior of a turbulent flow is dominated by molecular viscosity and the intensity of turbulence decreases to zero as in the case of low Reynolds number flows.

In this formula, u_τ is the wall shear velocity defined as $u_\tau = \left(|\tau_w|\rho^{-1}\right)^{1/2}$, where τ_w is the viscous shear stress at the wall, and z_P is the wall-normal coordinate of the wall-nearest grid point. The two constants are the von Karman constant $\kappa = 0.41$ and the empirical constant B, which is about 5.5 at a smooth flat plate but may take different values in other cases. For the logarithmic law to be correct, the distance to the first grid point z_P should approximately satisfy $30 < u_\tau z_\mathrm{P}/\nu < 300$.

The logarithmic law can be used to find the wall shear stress τ_w as a function of $\langle u \rangle_\mathrm{P}$. After that, the local equilibrium assumption allows us to find the turbulence properties at z_P. For the k-ϵ model, we have

$$k_\mathrm{P} = \frac{u_\tau^2}{C_\mu^{1/2}}, \quad \epsilon_\mathrm{P} = \frac{C_\mu^{3/4} k_\mathrm{P}^{3/2}}{\kappa z_\mathrm{P}}. \tag{11.37}$$

The wall function approach is strictly valid for an attached unidirectional boundary layer. Its generalization to the case of a more complex pattern of wall-parallel mean flow is straightforward. The situation becomes more difficult when the boundary layer experiences separation. The logarithmic layer ceases to exist in the separation zone and the wall functions become invalid. Other approaches must be used: RANS with near-wall resolution and low Reynolds number modification, or LES.

11.4 LARGE-EDDY SIMULATION (LES)

In LES, we directly calculate the mean flow and the unsteady large-scale and intermediate-scale motions. The effect of small-scale fluctuations on the rest of the flow is modeled. This introduces a modeling error, which, although generally smaller than in RANS, should not be disregarded. Avoiding the requirement of accurate resolution of small-scale motions significantly reduces the computational cost in comparison to DNS and makes it possible to simulate flows in realistically complex geometries at realistically high Reynolds numbers. On the other hand, the computational cost of LES is much higher than the cost of the RANS methods. Current use of LES in practical engineering analysis is limited to the flows for which representation of large-scale turbulent fluctuations, and not just the mean flow, is essential, although the area of applicability is expanding rapidly following the growth of computing power.

In this section, we provide an outline of the LES approach and introduce some popular models. A broader and more thorough discussion can be found in specialized texts and research literature.

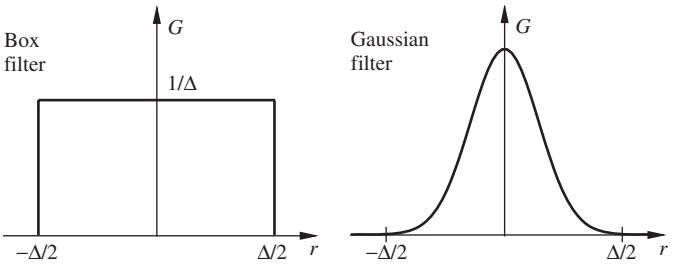


Figure 11.3 Box and Gaussian filters.

11.4.1 Filtered Equations

The key concept of LES is that of filtering. Instead of actual flow, we simulate the behavior of fields obtained in the result of application of a low-pass filter. In the formal way, the filtered velocity is

$$\bar{\boldsymbol{u}}(\boldsymbol{x},t) \equiv \int G(\boldsymbol{r},\boldsymbol{x})\boldsymbol{u}(\boldsymbol{x}-\boldsymbol{r},t)d\boldsymbol{r}, \tag{11.38}$$

where the integral is over the flow domain. G is a filter function that satisfies the normalization condition $\int G(\boldsymbol{r},\boldsymbol{x})d\boldsymbol{r} = 1$ and is nearly zero at $r = |\boldsymbol{r}| > \Delta/2$. The parameter Δ is called the filter width.

Various specific forms of the filtering function have been suggested. We will present two popular versions for the simple case when the filter is uniform (G does not depend on $\boldsymbol{x}$) and isotropic (the dependence on $\boldsymbol{r}$ is limited to the dependence on its absolute value r). Figure 11.3 shows the box filter

$$G(r) = \begin{cases} 1/\Delta & \text{if} \quad r < \Delta/2 \\ 0 & \text{if} \quad r > \Delta/2 \end{cases}$$

and the Gaussian filter

$$G(r) = \left(\frac{6}{\pi\Delta^2}\right)^{1/2} \exp\left(-\frac{6r^2}{\Delta^2}\right).$$

Evidently, the isotropic filtering operation is simply a weighted and localized (in a sphere of diameter Δ) averaging. Fluctuations with length scales significantly smaller than Δ are smoothed out. This is illustrated in Figure 11.4 using a one-dimensional example. The smallest scales remaining in the filtered signal are of the order of Δ. This explains the

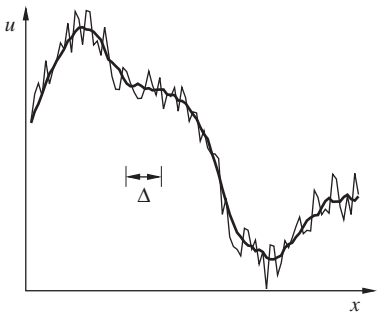


Figure 11.4 Effect of filtering on a turbulent signal. Bold curve shows the result of box filtering with filter width Δ.

computational efficiency of LES in comparison with DNS. Since the fluctuations with scales smaller than Δ are absent, it is sufficient to use a computational grid with steps approximately equal to Δ rather than to the much smaller Kolmogorov scale η.

Sometimes, the relation between the filter width and grid steps is taken as far as assuming near identity between them. The filtered fields, such as $\bar{\boldsymbol{u}}$, are called *resolved* fields, and the term *subgrid-scale* is used for the filtered out fluctuations

$$\boldsymbol{u}' = \boldsymbol{u} - \bar{\boldsymbol{u}}. \tag{11.39}$$

We will follow the less restrictive approach, in which the possibility that the grid steps are a different size than the filter width is not rejected. The appropriate terms in this case are *filtered* fields for $\bar{\boldsymbol{u}}$ and *residual* fields for $\boldsymbol{u}'$.

Our next step is to apply the filtering operation to the Navier-Stokes equations. The filtering operation is linear and commutes with partial derivatives. For example, applying it to $\partial u/\partial t$, we obtain

$$\begin{aligned}\overline{\left(\frac{\partial u}{\partial t}\right)} &= \int G(\boldsymbol{r},\boldsymbol{x})\frac{\partial u}{\partial t}(\boldsymbol{x}-\boldsymbol{r},t)\,d\boldsymbol{r} \\ &= \frac{\partial}{\partial t}\int G(\boldsymbol{r},\boldsymbol{x})u(\boldsymbol{x}-\boldsymbol{r},t)\,d\boldsymbol{r} = \frac{\partial \bar{u}}{\partial t}.\end{aligned}$$

For the Navier-Stokes system (11.14), the result of filtering is

$$\rho\frac{\partial \bar{u}_i}{\partial t} + \rho\frac{\partial}{\partial x_j}\left(\overline{u_i u_j}\right) = -\frac{\partial \bar{p}}{\partial x_i} + \mu\nabla^2\bar{u}_i, \quad \frac{\partial \bar{u}_i}{\partial x_i} = 0. \tag{11.40}$$

The main challenge of LES appears in these equations. The reason is the nonlinearity of the momentum equation. The filtered product $\overline{u_i u_j}$ cannot be expressed as a function of filtered velocity. In particular,

$$\overline{u_i u_j} \neq \bar{u}_i \bar{u}_j .$$

The filtered equations are usually rewritten as

$$\rho \frac{\partial \bar{u}_i}{\partial t} + \rho \frac{\partial}{\partial x_j} \left(\bar{u}_i \bar{u}_j \right) = -\frac{\partial \bar{p}}{\partial x_i} + \mu \nabla^2 \bar{u}_i - \frac{\partial \tau_{ij}^R}{\partial x_j}, \quad \frac{\partial \bar{u}_i}{\partial x_i} = 0, \qquad (11.41)$$

where

$$\tau_{ij}^R \equiv \rho \overline{u_i u_j} - \rho \bar{u}_i \bar{u}_j \qquad (11.42)$$

is the *residual stress tensor*.

Despite the similarity in appearance, there is a significant difference between this tensor and the Reynolds stress tensor (11.17) of RANS. The Reynolds stresses show the momentum flux due to *all* turbulent fluctuations, while the residual stresses only include the effect of fluctuations with length scales smaller than the LES filter width.

Further transformation is made by expanding τ_{ij}^R as a sum of anisotropic and trace parts:

$$\tau_{ij}^R = \tau_{ij}^r + \frac{1}{3} \delta_{ij} \tau_{ii}^R$$

and introducing the modified pressure field

$$\bar{p}^* = \bar{p} + \frac{1}{3} \tau_{ii}^R .$$

The final system of LES equations is

$$\rho \frac{\partial \bar{u}_i}{\partial t} + \rho \frac{\partial}{\partial x_j} \left(\bar{u}_i \bar{u}_j \right) = -\frac{\partial \bar{p}^*}{\partial x_i} + \mu \nabla^2 \bar{u}_i - \frac{\partial \tau_{ij}^r}{\partial x_j}, \quad \frac{\partial \bar{u}_i}{\partial x_i} = 0. \qquad (11.43)$$

The trace part of the residual stress tensor is now a part of the modified pressure field that can be found by a projection technique described in Chapter 10 (this would not work for LES of a compressible flow, in which case a special model of the trace part is required). The anisotropic residual stress tensor τ_{ij}^r remains in the equation in any case. It is unknown and cannot be reduced to a function of $\bar{\boldsymbol{u}}$ and $\bar{p}^*$. The system of LES equations (11.43) is not closed. Its numerical solution is impossible unless we find a way to model τ_{ij}^r in terms of the filtered flow fields.

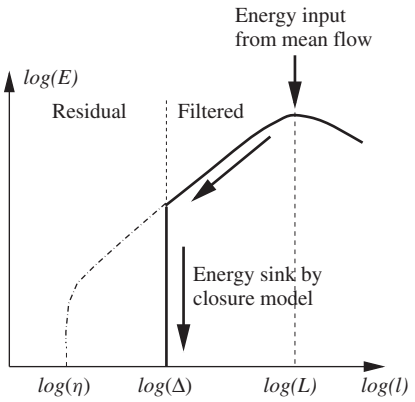


Figure 11.5 Schematic representation of the energy distribution and cascade in LES. L is the typical size of the largest and most energetic fluctuations. Δ is the filter width.

11.4.2 Closure Models

The models used to approximate τ_{ij}^r are called *closure* or *subgrid-scale* models. The first name is evidently due to the fact that the models close the LES equations (11.43) and allow them to be solved numerically. The second name reflects the fact that τ_{ij}^r represents the effect of residual (subgrid-scale) fluctuations on the filtered part of the flow.

What should we require from a model? One obvious requirement is that, to be a true closure, an approximation of τ_{ij}^r should depend only on filtered flow fields and known parameters, such as the filter width. There is a geometric requirement that a model is invariant to principal coordinate transformations. From the physical viewpoint, a model should represent the effect of residual fluctuations as accurately as possible. Yet another view of the role of a closure model is illustrated in Figure 11.5. Since the small-scale fluctuations are not simulated in LES, the turbulent energy cascade cannot proceed in its natural way, as shown in Figure 11.2. A closure model should provide an energy sink of appropriate strength at the length scale approximately equal to the filter width.

Among the numerous existing closure models, we describe the simplest and oldest—the Smagorinsky model. Proposed in 1963, the model has become the most popular choice of LES analysis and served as a basis for many other, more advanced models.

The Smagorinsky model uses the *eddy viscosity hypothesis:*

$$\tau_{ij}^r = -2\mu_t \bar{S}_{ij}, \tag{11.44}$$

where $\mu_t(\boldsymbol{x}, t)$ is the turbulent eddy viscosity and

$$\bar{S}_{ij} \equiv \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right)$$

is the rate of strain tensor of the filtered flow. The eddy viscosity is modeled, on the basis of the Prandtl's mixing length theory, as

$$\mu_t(\boldsymbol{x}, t) = \rho \ell_S^2 |\bar{S}| = \rho (C_S \Delta)^2 |\bar{S}|. \tag{11.45}$$

In this formula, $|\bar{S}| = \left(2\bar{S}_{ij}\bar{S}_{ij}\right)^{1/2}$ (summation over i and j is assumed). The characteristic length scale ℓ_S is approximated as a product of the filter width Δ and the empirical Smagorinsky constant C_S. The rate of energy sink by the closure model is

$$P_t = -\tau_{ij}^r \bar{S}_{ij} = 2\mu_t \bar{S}_{ij}\bar{S}_{ij} = \mu_t |\bar{S}|^2.$$

The numerical implementation of the Smagorinsky model is not difficult. We use the filtered flow computed at the previous time step or extrapolate from several previous time steps to evaluate $\bar{S}_{ij}$ and $|\bar{S}|$. The results are substituted into (11.45) and (11.44) to compute the eddy viscosity and residual stress tensor. Note that the filtering operation (11.38) does not appear in the Smagorinsky algorithm. We only use the filter width Δ, which is usually assumed to correspond to the typical size of the grid cell estimated as $\Delta = (\Delta x \Delta y \Delta z)^{1/3}$.

The popularity of the Smagorinsky model is fully justified by its simplicity and computational efficiency. The model, however, has significant drawbacks that make it less accurate and flexible than its more complex counterparts. The main problem is the empirical Smagorinsky constant C_S. There is no theoretical derivation that would justify a single optimal value or a functional dependency of C_S. The Kolmogorov theory of isotropic homogeneous turbulence leads to the estimate $C_S \approx 0.17$. This becomes an overestimate in flows with strong mean shear—for example, in the channel flow shown in Figure 11.1. The optimal value of the channel flow C_S providing the closest agreement with DNS and experiments was found to be about 0.1.

In general, the optimal value of C_S depends strongly and in a complex fashion on flow characteristics (strength of mean shear, Reynolds number, presence of density stratification, etc.). There is always the danger of using the wrong constant and, as a result, getting inaccurate results. One striking example is the model behavior in simulations of laminar flows. The residual stress τ_{ij}^r is practically zero in such flows, while the

filtered strain rate $\bar{S}_{ij}$ is not. The Smagorinsky model applied to a laminar flow would predict nonzero residual stress term and add a false dissipation mechanism leading to an incorrect flow picture. This feature of the model makes particularly dangerous its application to the flows that evolve between laminar and turbulent states or have laminar and turbulent zones.

Advanced LES models provide higher accuracy and flexibility, although at the cost of higher complexity and larger amount of computations. Some of them are improved versions of the Smagorinsky model, while others follow different approaches (e.g., not based on the concept of eddy viscosity). A detailed discussion of these models is beyond the scope of this book but available in monographs and research papers, some of which are listed at the end of the chapter.

We only mention the dynamic Smagorinsky model proposed by Germano et al. (1991) and extended by Lilly (1992). The method remedies the drawbacks of the original Smagorinsky model by providing an elegant and effective mechanism of automatic adaptation of C_S to the local conditions of the flow. After every time step, the filtered flow field is subjected to an additional *test* filter, which has the width $\tilde{\Delta} = 2\Delta$. The Smagorinsky constant C_S, which becomes a function of space and time, is estimated using the algebraic identities based on the assumption that the formulas (11.44) and (11.45) remain true at both filter widths.

11.4.3 Implementation of LES in CFD Analysis: Numerical Resolution and Near-Wall Treatment

Discussing the resolution requirements and, correspondingly, computational cost of LES analysis, we have to consider separately the inner part of the flow and the near-wall zone. For the inner part, the Kolmogorov description of turbulence can be adopted, at least in its most essential aspects: the energy of turbulent fluctuations decreases rapidly with the length scale, and the small-scale motions serve as a nearly passive receiver of the energy cascading from the large-scale motions (see Figures 11.2 and 11.5). A plausible conclusion is that an LES model gives an accurate prediction of the filtered flow dynamics if the filtered flow contains a larger portion, say, 80 percent of the total flow energy. The effect of the residual fluctuations on the filtered flow is inevitably predicted with some error by the closure model, but, since the residual fluctuations are weak, the error is probably not very significant.

Theoretical analysis and computational studies have shown that the requirement of 80 percent of flow energy in the filtered velocity is not

especially strict. The needed computational grids are, of course, much finer than the grids needed for laminar flows or for turbulent flows computed using RANS models. They are, however, much cruder than in DNS. Of the key importance is the fact that the size of LES grids for inner zones of fully turbulent flows *is practically independent of the Reynolds number.* The amount of computations, of course, varies with the complexity of the computational domain and large-scale features of the flow. In general, however, the computations are feasible on modern computers. As an example, we mention the simplest turbulent flow without walls, the isotropic turbulence in a periodic box. It can be shown (see, e.g., Pope (2000)) that the necessary grid consists of only about 40 points in each direction.

Simulation of near-wall zones is more difficult and, typically, requires special treatment. The difficulty is not in the boundary conditions themselves. Applying the filtering operation to the no-slip boundary condition (2.46) we obtain the condition for the filtered velocity field

$$\bar{\boldsymbol{u}} = \boldsymbol{U}_{\text{wall}} \text{ at the wall.} \tag{11.46}$$

The special treatment is necessary because of the nature of turbulent boundary layers developing near the solid walls. The small-scale fluctuations cannot be treated as mere passive receivers of energy. The flow properties are largely determined by the dynamics of small-scale coherent structures. The flow is very anisotropic. It is also strongly inhomogeneous. As illustrated in Figure 11.1, the intensity of turbulent fluctuations has a strong peak at some short distance to the wall. In the close vicinity of the wall (in the viscous sublayer), turbulent motions vanish, so the residual stress must disappear, while strong viscous stress is caused by the gradient of the mean flow.

Because of these features, simple extension of an LES model and computational grid suitable for the internal flow into the near-wall zone would generate significant error. As we already discussed in the previous section, the error is not limited to the near-wall zone, but leads to an incorrect picture of the entire flow. For example, the inner part of the channel flow in Figure 11.1 could be accurately reproduced by the standard Smagorinsky model with $C_S \approx 0.1$ and the grid consisting of several tens of cells or points in every direction. The errors of the near-wall approximation would, however, lead to unacceptable inaccuracy in estimation of wall friction and strongly distorted distributions of mean flow, turbulent fluctuation intensity, and other flow properties.

A part of the solution is to reduce the grid steps near the wall. There is, however, a problem that persists even when the grid resolution of

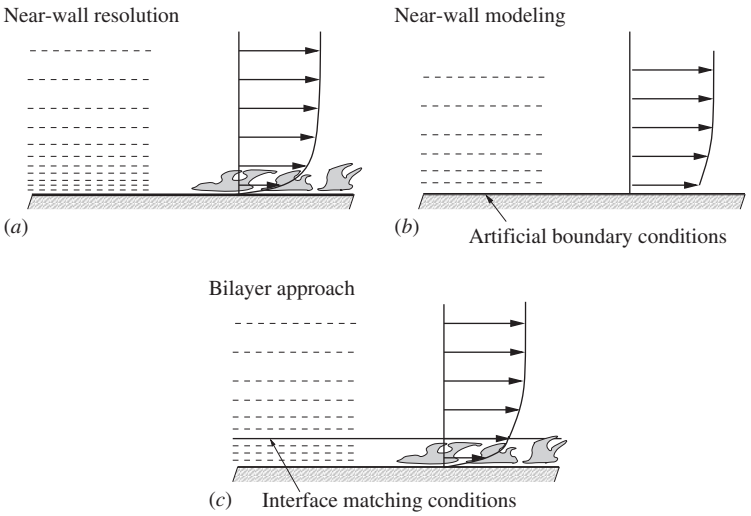


Figure 11.6 Approaches to near-wall treatment in LES.

near-wall zones is adequate. Many closure models become inaccurate and require modification in this zone. In particular, the Smagorinsky model (11.44), (11.45) significantly overestimates the residual stresses. One possible remedy is to artificially reduce the turbulent eddy viscosity using the van Driest damping

$$C_S^2 = C_{S0}^2 \left[1 - \exp\left(-(u_\tau z/\nu)/25\right)\right]^2, \tag{11.47}$$

where C_{S0} is the Smagorinsky constant in the core flow, and u_τ is the wall shear velocity defined in section 11.3.5.

There are three principally different ways to treat the near-wall problem. One of them, schematically illustrated in Figure 11.6a is to use a computational grid, in which steps decrease in the near-wall zone providing a nearly DNS-like resolution. This guarantees an accurate approximation of boundary layers, but the computational cost goes up significantly. The required number of floating point operations increases with the Reynolds number as $\sim Re^k$, where the exponent k varies depending on the situation, but, typically, is around 2. Evidently, this approach is unfeasible in practical CFD analysis of high-Re flows.

Another approach, illustrated by the scheme in Figure 11.6b is to model the effect of boundary layers instead of simulating their internal dynamics. Artificial boundary conditions are imposed in the form of model approximations for the shear components of the residual stress. These conditions

replace the no-slip conditions on tangential velocity components and imitate the momentum transport across the boundary layers. This approach works well when we can be certain that the flow has a unidirectional attached boundary layer. The artificial boundary conditions have the form of relations between the wall shear stress and the inner flow velocity at the grid point nearest to the wall. They are derived on the basis of the assumption of the logarithmic law behavior (11.36).

In the bilayer approach, illustrated in Figure 11.6c, the flow domain is divided into the core and the near-wall layer. The filtered flow in the core is simulated using LES. The near-wall flow is found using a less accurate, but computationally more efficient approach. Matching conditions are satisfied at the interface between the two zones. The popular and practical way to treat the near-wall layer is to apply a RANS model based on full Navier-Stokes or boundary layer equations. This approach has the special name *detached eddy simulations* (DES). It is implemented in many modern engineering CFD codes.

REFERENCES AND SUGGESTED READING

Davidson, P. A. *Turbulence.* New York: Oxford University Press, 2004.

Frisch, U. *Turbulence*, Cambridge, UK: Cambridge University Press, 1995.

Pope, S. B. *Turbulent Flows.* Cambridge, UK: Cambridge University Press, 2000.

Sagaut, P. *Large Eddy Simulation for Incompressible Flows: An Introduction.* New York: Springer 2001.

Tennekes, H. and J. L. Lumley. *A First Course in Turbulence.* Cambridge, MA: MIT Press, 1972.

Wilcox, D. C. *Turbulence Modeling for CFD,* 3rd ed. La Cañada, CA: DCW Ind. 2006.

Germano, M. U. Piomelli, P. Moin, and W. H. Cabot. "A dynamic subgrid-scale eddy viscosity model," *Phys. Fluids A* **3** (1991): 1760–1765.

Ishihara, T., T. Gotoh, and Y. Kaneda, "Study of high Reynolds number isotropic turbulence by direct numerical simulation," *Ann. Rev. Fluid Mech.* **41** (2009): 165–180.

Kim, J., P. Moin, and R. Moser, "Turbulence statistics in fully developed channel flow at low Reynolds number," *J. Fluid Mech.* **177** (1987): 133–166.

Launder, B. E. and D. B. Spalding, "The numerical computation of turbulent flows," *Comput. Methods Appl. Mech. Eng.* **3** (1974): 269–289.

Lilly, D. K. "A proposed modification of the Germano subgrid-scale closure method," *Phys. Fluids A* **4** (1992): 633–635.

Meneveau, C. and J. Katz. "Scale-Invariance and Turbulence Models for Large-Eddy Simulation," *Ann. Rev. Fluid Mech.* **32** (2000): 1–32.

Moin, P. and K. Mahesh. "Direct numerical simulation: A tool in turbulence research," *Ann. Rev. Fluid Mech.* **30** (1998): 539–578.

Orszag, S. A. & G. S. Patterson, "Numerical simulation of three-dimensional homogeneous isotropic turbulence," *Phys. Rev. Lett.* **28** (1972): 76–79.

Piomelli, U. and E. Balaras, "Wall-layer models for large-eddy simulations," *Ann. Rev. Fluid Mech.* **34** (2002): 349–374.

Prandtl, L. "Bericht über die Entstehung der Turbulenz," *Z. Angew. Math. Mech*. **5** (1925): 136–139.

Rogallo, R. S. "Numerical experiments in homogeneous turbulence." Technical report TM81315. NASA, 1981.

Spalart, P. R. "Detached-eddy simulation," *Ann. Rev. Fluid Mech.* **41** (2009): 181–202.

Smagorinsky, J. "General circulation experiments with the primitive equations: I. The basic equations," *Mon. Weather Rev.* **91** (1963): 99–164.

PROBLEMS

1. For the following hypothetical CFD tasks, compare DNS, LES, and RANS approaches. Discuss which of them is feasible and which is likely to produce desired accuracy and level of description.

   a) Find flow rate and drag in a rectangular duct of a residential air-conditioning system. Assume that the duct cross-section is a square of side 25 cm, and the average velocity is 1 m/s.

   b) Find turbulent drag on a slender body of length 2 m moving at zero attack angle in the air with speed 10 m/s.

   c) Predict motion of a hurricane.

   d) Analyze flow around a swimming fish. Assume the fish is 20 cm long and moves with speed of 40 cm/s.

   e) Try to understand how the local properties of turbulent fluctuations created in a cup of coffee by spoon movements affect the rate of sugar dissolution.

   f) Analyze dynamics of a solar protuberance.

2. Does the steady-state formulation of a CFD problem make sense in DNS, LES, or RANS?

3. For DNS in a periodic box, it was estimated that the typical size of large-scale eddies is 1000 larger than the Kolmogorov dissipation scale η. What are the resolution requirements? Is the analysis feasible?

4. Prove that the averaging operations (11.8)–(11.10) are linear and commute with space and time derivatives.
5. Prove the relations (11.11), (11.13), and (11.16).
6. RANS is applied to a turbulent flow with convection heat transfer. Apply the averaging operation to (2.27) to derive the RANS equation for temperature. Are there any terms that need modeling?
7. If your course involves exercises with a CFD code, study the documentation to determine which RANS models are implemented. What approach is taken to the near-wall treatment?
8. Prove that the LES filtering operation (11.38) is linear and commutes with space and time derivatives.
9. What determines the adequate grid step in LES far from the walls?
10. LES is applied to a turbulent flow with convection heat transfer. Apply the filtering operation to (2.27) to derive the LES equation for temperature. Are there any terms that need modeling?
11. If your course involves exercises with a CFD code, study the documentation to determine whether the LES option is implemented. Which closure models are used? What approach is taken to the near-wall treatment?
12. Consider the primitive LES or RANS model, in which the eddy viscosity formulas (11.19) or (11.44) are applied with constant eddy viscosity μ_t. What is the drawback of this model? Should we expect accurate results?

12

COMPUTATIONAL GRIDS

12.1 INTRODUCTION: NEED FOR IRREGULAR AND UNSTRUCTURED GRIDS

A well-constructed computational grid is an essential ingredient of CFD analysis. Having such a grid is a necessary and significant step toward an accurate, efficient, and robust numerical solution. On the contrary, as any CFD practitioner would confirm and illustrate by spectacular examples, a poorly designed grid leads to low accuracy, slow convergence, and, sometimes, numerical instability.

Our discussion has so far largely ignored the questions of grid design. For the sake of simplicity and transparency, we have presented CFD techniques on the example of a structured, uniform, and orthogonal grid, in which grid points or finite volume cells are formed by intersections of coordinate lines of a Cartesian coordinate system. Non-Cartesian and unstructured grids have been addressed (for example, in the discussion of finite volume methods in Chapter 5), but only briefly.

The general principles of CFD approximation discussed in the previous chapters remain valid if the grid is nonorthogonal, nonuniform, or unstructured. On the other hand, certain new questions arise. They, in particular, concern the effect of the type and properties of the grid on the numerical solution, methods of grid generation, and methods of solution of CFD problems on irregular grids. The issues are too complex to

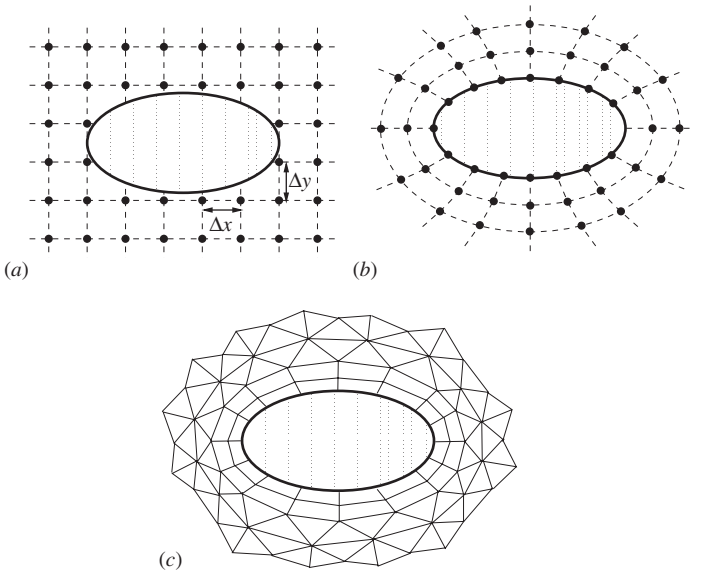


Figure 12.1 A case for irregular grids — complex geometry: *(a)* a non-rectangular geometry approximated poorly by a regular Cartesian grid; *(b)* the same geometry approximated by a boundary-fitting curvilinear structured grid; *(c)* the same geometry approximated by a boundary-fitting unstructured grid.

be fully addressed in this book. For a detailed discussion, the reader is referred to specialized texts, some of which are listed at the end of the chapter. Here, we only discuss the most basic concepts and provide a few recommendations.

There are two often coexisting features of practical engineering flows and heat transfer processes that make the use of a uniform Cartesian grid a poor choice. One is illustrated in Figure 12.1. The solution domain may have complex, possibly curvilinear boundaries, which cannot be fit with a Cartesian grid.

There are several ways to address this problem. The simplest one is to ignore it and employ a regular Cartesian grid anyway. This is illustrated in Figure 12.1a. The boundary conditions have to be set at the grid points nearest to the boundary. This means that the boundary itself is replaced by a staircase-like shape. The approach is not recommended, primarily because of the significant error it introduces into the solution. Furthermore, programming a CFD algorithm on such a grid is challenging, since the number of grid points is different for different grid lines and the approximation of Neumann and mixed boundary conditions is complicated.

The commonly used and recommendable approaches are illustrated in Figure 12.1b and 12.1c. As shown in Figure 12.1b, a curvilinear coordinated system can be introduced such that the boundaries coincide with certain coordinate lines. The grid points can be positioned directly at the boundary, which preserves accuracy. The programming difficulties are also successfully resolved. Another possible and commonly used approach is illustrated in Figure 12.1c. We use a boundary-fitting unstructured grid.

Sometimes, the boundaries of the computational domain are moving (imagine waves on the ocean surface or a flow around rotating blades of a mixer in a chemical processor). In this case, a boundary-fitting *moving grid* can be designed and used.

The second common reason for the use of non-uniform grids is that gradients of solution variables may be of strongly different amplitudes in different areas of solution domain. For example, mean velocity and turbulence variables experience large gradients in boundary layers near solid walls. On the contrary, far from walls, the gradients are small (see Figure 12.2). In all such cases, it seems reasonable to use smaller grid steps or cells in the areas, where the solution variables have strong gradients and reduce computational effort by using larger steps or cells in other areas, where the gradients are weak. This technique is called the grid *stretching* or *clustering*. As illustrated in Figure 12.2b, this can be achieved through

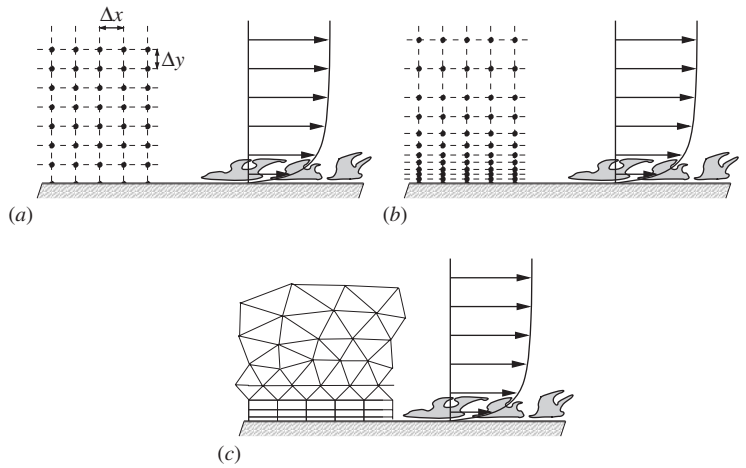


Figure 12.2 A case for irregular grids — strong spatial variability of gradients of solution: *(a)* A flow with strong velocity gradients near the wall approximated poorly on a uniform grid; *(b)* The same flow approximated on a stretched grid adapted to solution gradients; *(c)* The same flow approximated on an unstructured grid refined in the area of strong gradients.

the use of a structured grid designed on the lines of a coordinate system compressed and stretched as necessary. Alternatively, as illustrated in 12.2c, we can build an unstructured grid refined in the areas of strong gradients.

12.2 IRREGULAR STRUCTURED GRIDS

The structured grids is the only realistic choice for finite-difference and spectral methods. They can also be used in combination with finite volume discretization, although unstructured grids discussed in the next section are more common in that case.

12.2.1 Generation by Coordinate Transformation

Grid points of a structured irregular grid can be generated as points of intersection of coordinate lines of a non-Cartesian coordinate system. For simplicity, we will illustrate the method using the two-dimensional grid configurations shown in Figure 12.3. Generalization to the

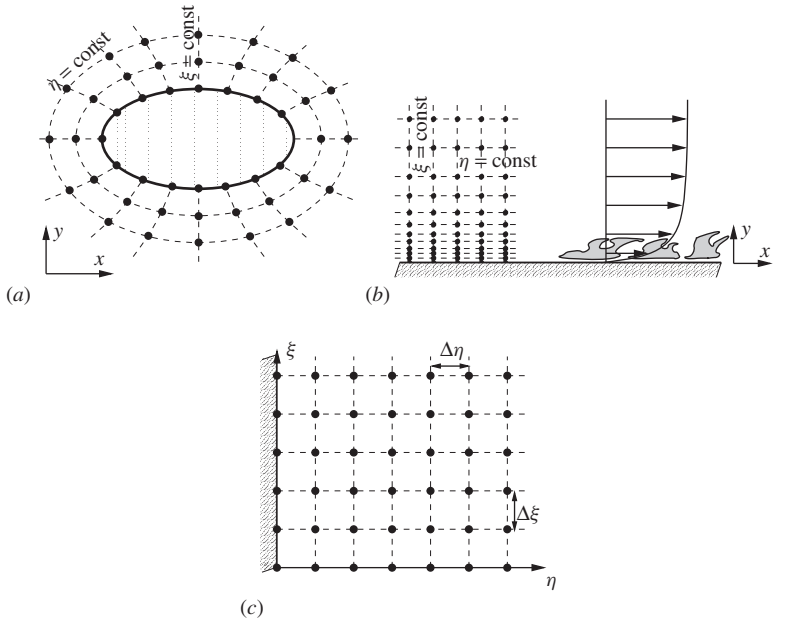


Figure 12.3 Coordinate transformation associated with irregular grids: *(a)* and *(b)* irregular coordinate systems in the physical space; *(c)* corresponding Cartesian coordinate system in the computational space.

three-dimensional case is straightforward. For our discussion, we will assume that the Cartesian (x, y) and non-Cartesian (ξ, η) coordinates are related as

$$\xi = \xi(x, y, t), \quad \eta = \eta(x, y, t) \tag{12.1}$$

and

$$x = x(\xi, \eta, t), \quad y = y(\xi, \eta, t). \tag{12.2}$$

The relation between the two coordinate systems can be viewed as a mapping (transformation) between the solution domains in the physical space shown in Figures 12.3a and 12.3b and a rectangular computational domain in the space of Cartesian coordinates (see Figure 12.3c).

The transformation should satisfy certain mathematical properties that guarantee that the correspondence between the points of physical and computational spaces is one-to-one, the irregular coordinate lines of the same family (for example lines $\eta = const$ in Figure 12.3a or 12.3b) do not cross, and any two lines of different families do not cross more than once.

After a proper coordinate transformation has been found, we replace the equations by their discretization approximations and solve them. The discretization is conveniently performed in the transformed space, since the coordinates are Cartesian. One extra step is to be done, though. We have to express the partial derivatives with respect to the physical coordinates in terms of partial derivatives with respect to the coordinates of the computational domain.

For example, let the unknown function be $u(x, y, t)$. In the transformed coordinates, it becomes $u = u\left[\xi(x, y, t), \eta(x, y, t), t\right]$. The first derivatives we obtain using the chain rule:

$$u_x = u_\xi \xi_x + u_\eta \eta_x \tag{12.3}$$

$$u_y = u_\xi \xi_y + u_\eta \eta_y \tag{12.4}$$

$$u_t = u_\xi \xi_t + u_\eta \eta_t + u_t. \tag{12.5}$$

Formulas for second derivatives can be obtained by differentiating (12.3)–(12.5) and applying the chain rule again. For example,

$$\begin{aligned} u_{xx} &= \partial\left(u_\xi \xi_x + u_\eta \eta_x\right)/\partial x \\ &= \xi_x \partial\left(u_\xi\right)/\partial x + u_\xi \xi_{xx} + \eta_x \partial\left(u_\eta\right)/\partial x + u_\eta \eta_{xx} \\ &= \xi_x\left(u_{\xi\xi}\xi_x + u_{\xi\eta}\eta_x\right) + u_\xi \xi_{xx} + \eta_x\left(u_{\eta\xi}\xi_x + u_{\eta\eta}\eta_x\right) + u_\eta \eta_{xx} \\ &= u_{\xi\xi}\xi_x^2 + 2u_{\xi\eta}\xi_x\eta_x + u_{\eta\eta}\eta_x^2 + u_\xi \xi_{xx} + u_\eta \eta_{xx}. \end{aligned} \tag{12.6}$$

After rewriting, the partial differential equations are discretized in the transformed coordinates. For example, in the finite-difference discretization, we can approximate

$$u_x|_{(\xi_i,\eta_j)} = \left(u_\xi \xi_x + u_\eta \eta_x\right)_{(\xi_i,\eta_j)}$$

using cental differences, which gives

$$u_x|_{(\xi_i,\eta_j)} = \frac{u^n_{i+1,j} - u^n_{i-1,j}}{2\Delta\xi}\xi_x + \frac{u^n_{i,j+1} - u^n_{i,j-1}}{2\Delta\eta}\eta_x, \tag{12.7}$$

where $u^n_{i,j} = u(\xi_i, \eta_j, t^n)$ and ξ_x, η_x are computed at the point (ξ_i, η_j).

The formula (12.7) illustrates the fact that the discretization equations do not require the explicit algebraic expressions (12.1) and (12.2). The only information needed is the set of values of the partial derivatives ξ_x, ξ_y, η_x, etc. at the grid points (ξ_i, η_j). They can be calculated using analytical derivatives of (12.1) or numerically. Another method useful when the only available analytical expressions for the coordinate transformation are (12.2) is to apply Jacobian-based mathematical identities, such as

$$\xi_x = -\frac{\partial(f,g)}{\partial(x,\eta)} \Big/ \frac{\partial(f,g)}{\partial(\xi,\eta)}, \tag{12.8}$$

where the coordinate transformation is represented by

$$f = x - x(\xi,\eta,t) = 0 \text{ and } g = y - y(\xi,\eta,t) = 0.$$

12.2.2 Examples

We shall give two examples of coordinate transformation. In the first example, the need for a non-Cartesian grid arises because of the nonrectangular shape of the boundary. Conduction heat transfer problem is solved in a domain that has the form of a circular ring with inner and outer radii R_i and R_o (see Figure 12.4). The steady-state temperature distribution is governed by the Laplace equation

$$T_{xx} + T_{yy} = 0.$$

The boundary conditions are of the von Neumann type (known heat flux) with the normal derivative $\partial T/\partial r$ equal to $g(r)$ and $f(r)$ at, respectively, $r = R_i$ and $r = R_o$.

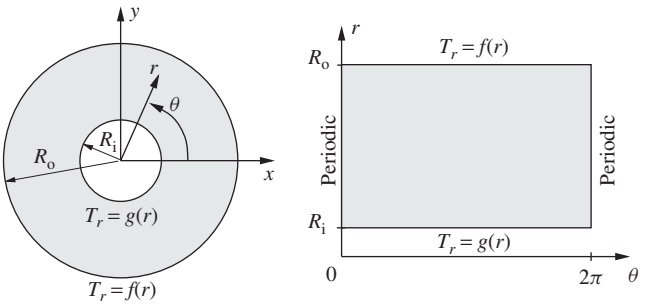


Figure 12.4 Heat transfer in a ring as an example of boundary fitting coordinate transformation.

The boundary-fitting coordinates are obviously polar coordinates r and θ, such that $x = r\cos\theta$ and $y = r\sin\theta$. The computational domain in these coordinates is the rectangle (see Figure 12.4)

$$R_i \le r \le R_o, \quad 0 \le \theta \le 2\pi.$$

The Laplace equation in the new coordinates is

$$\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r} + \frac{1}{r^2}\frac{\partial^2 T}{\partial \theta^2} = 0. \tag{12.9}$$

It is discretized as

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{(\Delta r)^2} + \frac{1}{r_i}\frac{T_{i+1,j} - T_{i-1,j}}{2\Delta r} + \frac{1}{r_i^2}\frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{(\Delta\theta)^2} = 0. \tag{12.10}$$

Indices i and j refer to the grid point (r_i, θ_j). The boundary conditions at $r = R_i$ and $r = R_o$ remain the same. At $\theta = 0$ and $\theta = 2\pi$, periodic boundary conditions are imposed.

In the second example, the coordinate transformation is used to improve the numerical resolution in the zone of strong gradient. For the boundary layer flow illustrated in Figure 12.2, the grid should be stretched in the wall-normal y direction so that the step Δy decreases near the wall and remains large far from it. There are many coordinate transformations that achieve this effect. For example, we can use exponential, logarithmic,

polynomial, or hyperbolic functions. The logarithmic transformation is (we assume that the wall is at $y = 0$)

$$\xi = x \quad \text{and} \quad \eta = \ln(y + 1). \tag{12.11}$$

Differentiating the second expression, we find the relation between the grid steps

$$\Delta\eta = \frac{1}{y+1}\Delta y \quad \text{or} \quad \Delta y = (y + 1)\Delta\eta.$$

Taking into account that $\Delta\eta$ is constant, we see that Δy has the minimum value equal to $\Delta\eta$ at the wall and increases linearly with the distance to it.

12.2.3 Grid Quality

We have to mention that not any coordinate transformation produces an acceptable computational grid. Sometimes, numerical solution of the transformed equations leads to large discretization errors, slow convergence, or numerical instability. The reasons of this behavior and the mathematical apparatus needed to describe it are discussed in specialized texts. We will only list the properties that combine into a general characteristic that can be called *grid quality*, and give some rules of good computational practice.

Distortion: This term refers to the deviation from orthogonality between the intersecting lines of the non-Cartesian coordinate system. The best results are provided by orthogonal grids. If the grid needs to be nonorthogonal, the distortion angles should be possibly close to 90°. There are no universal limits, but grids with angles smaller than 45° or larger than 135° are usually considered dangerously distorted.

Ratio of Adjacent Cell Sizes: We have to avoid strong variations of grid steps. As a general guideline, it is usually accepted that the ratio between the typical sizes of any two adjacent grid cells should not exceed two. This requirement often needs enforcing when we construct stretched grids for flows with boundary layers and other zones of strong gradient. The transition from the fine grid within such a zone to a crude grid in the outer flow should be gradual.

Cell Aspect Ratio: We have to avoid strongly anisotropic grids. In general, it is recommended that the ratio between the grid steps in different directions is neither large or small. The specific limits on the aspect ratio

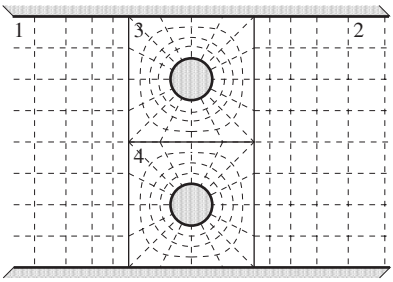


Figure 12.5 Example of a block-structured grid: A two-dimensional flow past two cylinders in a channel is computed using a grid consisting of four blocks marked by numbers 1 to 4.

vary with the nature of the flow and the type of the computational scheme. One particular case when high aspect ratio cells are acceptable is the boundary layer flows. Such flows are characterized by high velocity in one direction (let it be x) and strong velocity gradients in the other two directions. In that case, it is possible to use Δx significantly larger than Δy and Δz. In any case, the cells with aspect ratios higher than five have to be avoided.

Block-Structured Grids: Our last comment concerns the possibility of block-structured grids. They can be used in the geometries, which are too complex to be meshed by a single structured grid, but can be subdivided into several zones of relatively simple shapes. Within each block, a structured grid is constructed. Special matching conditions are imposed at the interfaces between the blocks. As an example of a block-structured arrangement, Figure 12.5 shows a grid build for a two-dimensional flow past an array of two cylinders in a channel.

12.3 UNSTRUCTURED GRIDS

One can easily imagine a complex geometry, in which a structured or block-structured grid is either impossible or very difficult to build. Our only practical choice in this case is to cover the domain with an unstructured grid and apply finite element or finite volume discretization.[1] As a typical example, Figure 12.6 shows the grids built to solve the problems

[1]Finite difference discretization on an unstructured grid is possible in principle, but has never been seriously tried because of the difficulties involved and the existence of a much simpler alternative—the finite volume method.

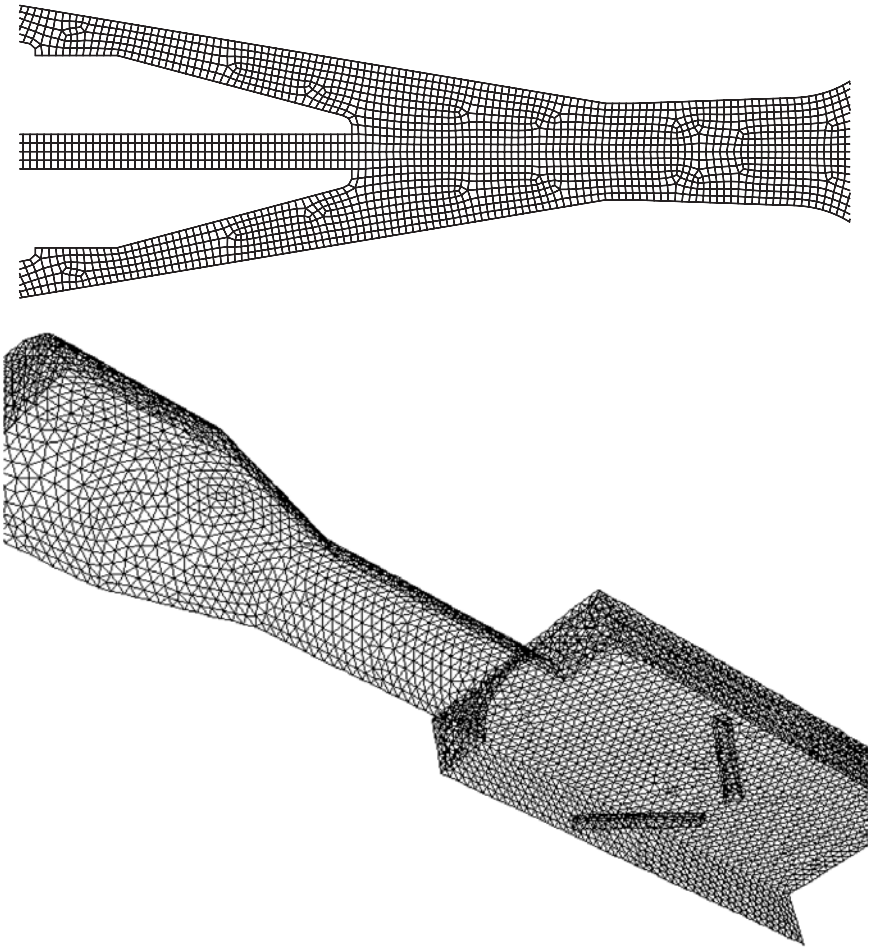


Figure 12.6 Examples of unstructured grids in complex solution domains (courtesy of Y. Wang and P. Mohanty, University of Michigan - Dearborn). The grids are automatically generated using GAMBIT 2.3, part of FLUENT 6.3. Both pictures show nozzles of plasma spray systems. On the top: the grid used for preliminary analysis of axisymmetric flow within the nozzle of SG100 plasma torch by Praxair. On the bottom: the three-dimensional grid for analysis of the novel torch design combining HVOF (High-Velocity-Oxy-Fuel) and twin-wire arc spray systems.

of turbulent gas flow and heat transfer within the nozzles of two thermal plasma spray systems used to generate special coatings.

In this section, we discuss the unstructured grids in context of their application with the finite volume method. This type of solution has become the prevalent tool of practical CFD analysis implemented in many

widely used commercial and noncommercial CFD codes. There are three main reasons for the popularity of unstructured grids:

1. Possibility of application to arbitrarily shaped computational domains
2. High level of flexibility and control of grid parameters such as cell shape and size
3. Existence of automated algorithms of grid generation and solution of discretized equations

The combination of these factors creates an attractive situation for a CFD practitioner. The same code can be used to solve a wide range of problems with different geometries, flow types, discretization accuracies, and so on.

The disadvantages of the unstructured grid approach are primarily related to the complexity of the grid description. The grid points and cells cannot be identified through a simple system of two or three indices. Instead, the information on location, shape (typically determined through locations of all vertices), and connectivity to neighbors should be specified and stored for every cell. Luckily for us, the arduous task of handling these data is usually taken care of by a CFD code. There are, however, consequences of the complex and irregular data structure that cannot be ignored. Most importantly, the matrices of the systems of discretization equations no longer have band-diagonal or block-diagonal form. This leads to slower convergence and higher computational cost of iterations. In general, solutions on unstructured grids have lower computational efficiency than solutions on structured grids of the same size. It is, therefore, recommended to use structured grids any time it is allowed by the geometry of the computational domain.

The issues involved in the solution of partial differential equations on unstructured grids and generation of such grids are complex. Taking into account that ready-to-use algorithms accomplishing these tasks are typically available in engineering CFD tools, we will limit our discussion to basic facts and a few practical recommendations.

12.3.1 Grid Generation

The common practice of modern engineering CFD is to use automated grid-generation algorithms. Commercial codes usually offer such algorithms as parts of the software packages. The task of grid buildup, which used to require days or even weeks of a CFD practitioner's time, is now accomplished with relatively low effort within hours or minutes. At the

same time, the user retains substantial control over the grid properties. This includes setting the topology and typical size of the cells and determining the areas where grid refinement is needed. It is also the user's responsibility to verify that the grid contains no "bad" cells (the characteristics of grid quality are discussed in section 12.3.4).

In the most common approach to grid generation, the process is initiated by the user, who specifies the mesh points at the boundaries of the domain. The user can exert control over the future distribution of cell sizes at this stage. The automated grid generator uses the mesh points to build the first layer of cells adjacent to the boundary. The second layer of cells is then built on the top of the first, and so on, converging toward the center of the computational domain in the manner of an advancing front.

After the grid has been built, the control is returned to the user, who checks the quality of the cells and modifies them, if necessary. Many CFD codes assist the user by providing information on essential cell characteristics and offering automated tools for fixing the cells.

12.3.2 Finite Volume Discretization on Unstructured Grids

Before we discuss the grid properties, we have to recall certain features of the finite volume discretization. The basic principles of the discretization presented in Chapter 5 fully apply in the case of unstructured grids. The integral balance equations leading to the discretization obviously remain the same. The volume and surface integrals are usually approximated by the midpoint formulas of the second order (see (5.7) and (5.8)).

The major modifications caused by the grid irregularity concern the methods of interpolation applied to approximate solution variables at the midpoints of cell faces. We will summarize the main issues, some of which have already been discussed in section 5.3.4. For convenience, Figure 12.7 reproduces the illustration of Figure 5.5.

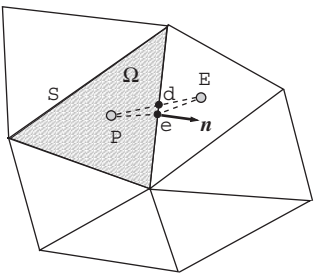


Figure 12.7 Difficulties of finite volume interpolation on unstructured grids.

A serious problem arises if we try to use interpolations of high order based on more than two grid points. As a rule, any three neighboring grid points of an unstructured grid do not form a straight line. The interpolation formulas become more complex than in the Cartesian grid case. Moreover, the complexity increases dramatically with the number of grid points involved. As a result of this, schemes of the second order remain most popular, and the schemes of the order higher than third are very rarely used.

Other problems appear even if we apply the second-order interpolation. As illustrated in Figure 12.7, the line connecting two neighboring grid points (which we identify with the cell centers such as P and E in Figure 12.7) does not necessarily pass through the face midpoint e. The linear interpolation

$$\Phi_{\mathrm{e}} = \gamma \Phi_{\mathrm{P}} + (1 - \gamma)\Phi_{\mathrm{E}},$$

where $\gamma = |\mathrm{eE}|/|\mathrm{PE}|$ and Φ is a solution variable, reduces its order of approximation from second to first. The additional first-order error is proportional to $|\mathrm{ed}|$, where d is the point of intersection of the line PE with the face. This error, can be neglected if the angle between $|\mathrm{eP}|$ and $|\mathrm{eE}|$ is close to 180°.

Yet another difficulty illustrated in Figure 12.7 is that the line $|\mathrm{PE}|$ can be nonparallel to the normal $\boldsymbol{n}$. The simple central difference formula

$$\left(\frac{\partial \Phi}{\partial n}\right)_{\mathrm{e}} \approx \frac{\Phi_{\mathrm{E}} - \Phi_{\mathrm{P}}}{|\mathrm{PE}|}.$$

does not generate a second-order approximation of the normal derivative of Φ. One method of dealing with this problem is described in section 5.3.4. Various other methods have been developed. None of them is entirely free from drawbacks in the form of loss of accuracy or reduction of computational efficiency. The drawbacks become more pronounced in grids with larger misalignment angles.

Reliable second-order accuracy and maximum computational efficiency are provided by the grids in which the lines connecting centers of neighboring cells are perpendicular to the faces separating the cells and cross these faces at their midpoints. The unstructured grids possessing this property are called *orthogonal unstructured grids*. Interestingly, the cells of such grid do not have to be rectangular. For example, as illustrated in Figure 12.8, a two-dimensional grid consisting of rectangles and equilateral triangles belongs to the class.

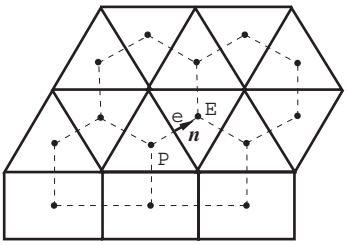


Figure 12.8 Example of a two-dimensional orthogonal unstructured grid.

12.3.3 Cell Topology

In theory, cells of arbitrary convex polygonal (in 2D) or polyhedral (in 3D) shape can be used in finite volume computations. In practice, however, the CFD codes usually limit the choice to few basic shapes shown in Figure 12.9. In the two-dimensional case, we can use quadrilateral (Figure 12.9a) or triangular (Figure 12.9b) cells. In three-dimensional grids, the most common elements are hexahedra (Figure 12.9c) and tetrahedra (Figure 12.9d), although prisms and pyramids (Figure 12.9e, f) are

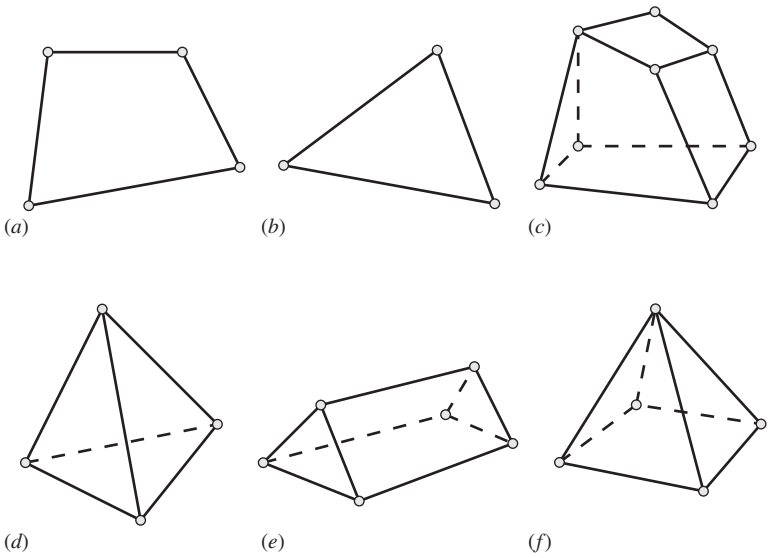


Figure 12.9 Typical shapes of unstructured grid cells. *(a)* and *(b)*, quadrilateral and triangular cells used in 2D grids; *(c)–(f)*, hexahedra, tetrahedra, prisms (wedges), and pyramids used in 3D grids.

also used. Many CFD codes allow combination of cells of different types (e.g., quadrilateral and triangular) within one grid.

The basic cell shapes are sufficient to build unstructured grids of great geometric flexibility. Any computational domain can be covered by such a grid. In many cases, this can be done relatively easily by applying an automated grid-generation code. Figure 12.6 shows two examples of automatically generated grids: a two-dimensional grid consisting of quadrilateral cells and a three-dimensional grid in which the cells are tetrahedral.

12.3.4 Grid Quality

The grid quality is a complex characteristic that determines the overall accuracy of solution at a given number of grid points and a given discretization scheme. We have already discussed the quality of structured grids in the previous section. In the case of unstructured grids, new aspects appear because of larger diversity and flexibility of cell shapes. In general, the matter of quality requires closer attention, since grids are generated automatically. Typically, an unstructured grid needs quality control and optimization after it has been built.

Some parameters of the grid quality are evident. For example, the cell sizes should not exceed the limits set by the numerical resolution requirements, which, in turn, are determined by the features of the computed solution. Other parameters are less evident and do not allow precise definitions. Furthermore, importance of various quality criteria varies with the flow type and the kind of analysis. In the result of all this, determining the grid quality is often a matter of experience and empirical knowledge rather than theoretical arguments.

We will briefly review the basic aspects of quality of unstructured grids. More detailed and specific information is available in specialized monographs, CFD research literature, and online resources (the latter should be, of course, treated with caution).

Near-orthogonality: As discussed, the best accuracy is provided for the second-order finite volume schemes if the unstructured grid is orthogonal. Building a perfectly orthogonal grid is not always feasible. The recommended strategy is to design a grid that is as close to orthogonal as possible. An example of a near-orthogonal arrangement can be seen in Figure 12.10a. On the contrary, the cell combinations shown in Figure 12.10b,c are far from orthogonal. If included into a grid, they are likely to compromise the accuracy of the entire solution and, therefore, should be avoided.

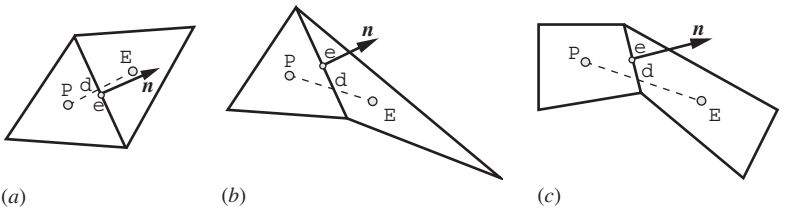


Figure 12.10 Examples of nearly orthogonal and strongly nonorthogonal (and therefore undesirable) cell combinations. In the near-orthogonal case *(a)*, the line EP that connects grids points of two neighboring cells is nearly parallel to the normal $\boldsymbol{n}$ to the face separating the cells and crosses the face at the point d close to the midpoint e. In the strongly nonorthogonal cases *(b)* and *(c)*, the angle between EP and $\boldsymbol{n}$ and the distance between d and e are large.

Undesirable Cell Shapes: Certain cell shapes have to be avoided, since they may cause large approximation errors, spurious oscillations, slow-down of convergence, or even numerical instability. One common danger is the cells of unacceptably large aspect ratio. The recommended criterion is the same as in the case of structured grids. The ratio between cell dimensions in any two orthogonal directions should not be smaller than 0.2 or larger than 5. Also, similarly to structured grids, we have to avoid strong (approximately larger than two-fold) differences between the sizes of neighboring cells.

Undesirable topological distortions may occur to nontetrahedral cells of three-dimensional unstructured grids. The list includes strongly warped and strongly sheared cells, and cells with centroids located outside the cell body. The automated grid-generation algorithms usually provide an opportunity of grid check operation, during which the topological quality of each cell is assessed and distortions are detected. It is a requirement of good CFD practice to always perform this operation and modify the grid in accordance to its results.

Choice of Cell Topology: We will consider the most common dilemma, the choice between quadrilateral and triangular cells in 2D and between tetrahedral and hexahedral cells in 3D. The triangular and tetrahedral cells provide certain geometric advantages. The automated generation of grids consisting of such cells is relatively easy. It can be accomplished for arbitrarily shaped domains at low risk of creating undesirable cell shapes. By contrast, generating a grid of quadrilateral or hexahedral elements often creates a number of topologically bad cells, especially in the cases of complex domains. Another advantage of triangular and tetrahedral cells is that we can generate a nearly orthogonal grid by simply requiring that the cell shape does not deviate

too far from equilateral and that sizes of any two neighboring cells do not differ too much. Near-orthogonality of grids consisting of quadrilateral or hexahedral cells requires that the cells are nearly rectangular, which cannot be achieved for all cells if the domain has a curvilinear boundary.

The arguments in favor of quadrilateral and hexahedral cells are related to the fact that the second-order finite volume schemes based on the midpoint approximation of cell and face integrals, linear interpolation, and central differences are most widely used with unstructured grids. The accuracy of these schemes is, in general, higher if the cells are quadrilateral or hexahedral. The reason is the partial canceling of the errors of discretization of diffusive terms at opposite cell faces. The approximation of convective terms is also achieved with higher accuracy on quadrilateral and hexahedral cells, if the computed flow is predominantly in one direction and the cells are oriented with one set of opposite faces parallel or nearly parallel to the flow. An important and commonly observed situation of this kind is within the boundary layers at solid walls.

It is recommended for flows with attached boundary layers that the grid includes a layer of thin quadrilateral (in 2D) or hexahedral (in 3D) cells adjacent to the wall. The rest of the domain can be filled with cells of any kind chosen in accordance with domain geometry and accuracy considerations. Illustrations of such arrangements can be seen in Figure 12.1c and 12.2c.

Example: To illustrate the discussion of grid quality let us visually analyze the grids in Figure 12.6. Both grids are generated automatically using the code GAMBIT 2.3. The upper two-dimensional and lower three-dimensional grids are built of quadrilateral and tetrahedral cells, respectively. The grids do not contain topologically bad cells (it was confirmed to the author that the grids had passed the examination by a grid-checking algorithm). The near-orthogonality is evident for the upper grid that consists of nearly rectangular cells. The lower grid can be assumed nearly orthogonal, since the tetrahedral cells are nearly equilateral and approximately of the same size.

Both grids in Figure 12.6 can be criticized.[2] The numerical resolution of the upper grid is inadequate. Four cells used across the channel are unlikely to be sufficient to accurately resolve the flow. In the lower grid,

[2]It should be mentioned that the grids shown in Figure 12.6 exemplify the situation when the accuracy of the CFD solution is not of the highest priority. Figure 12.6 shows only parts of the solution domains. The parts outside the nozzle, which are not shown, play more important role in the analysis of the spraying process. The error of approximation of the flow within the nozzle, albeit possibly large by CFD standards, is definitely smaller than the error introduced by the physical model of electric discharge arc and plasma formation.

the boundary layer treatment, which would insert a layer of hexahedral cells between the wall and the tetrahedral mesh, is not applied.

REFERENCES AND SUGGESTED READING

Ferziger, J. H., and M. Perić. *Computational Methods for Fluid Dynamics*, 3rd ed. New York: Springer, 2001.

Thompson, J. F., B. Soni, and N. P. Weatherrill. *Handbook of Grid Generation*. Clermont, FL: CRC Press, 1998.

Liseikin, V. D. *Grid Generation Methods*. New York: Springer-Verlag, 1999.

PROBLEMS

1. For each of the following flows, determine whether it is better to use a regular Cartesian, irregular structured, or unstructured grid. Provide arguments supporting your answer.

   a) Flow in a gap between two concentric spheres rotating with different velocities around a common axis (the so-called spherical Couette flow)

   b) Flow of air past a car moving in a tunnel (see Figure 2.5)

   c) Flow of ocean water around the supporting structure of an offshore oil drilling platform

   d) Fully developed turbulent flow in a plane channel between two parallel walls (see Figure 11.1)

2. For each of the flows in Problem 1, for which an irregular structured grid has been found necessary and possible, suggest a coordinate transformation.

3. For the coordinate transformation (12.1)–(12.2), derive the formula for u_{xy}.

4. For the coordinate transformation (12.1)–(12.2), develop the second-order finite difference approximations of u_y, u_{xx}, and u_{xy} at (ξ_i, η_j). Apply central differences for discretization of derivatives with respect to ξ and η.

5. If your course involves exercises with a CFD code, study the manual to determine which types of cell shapes are available for unstructured grids. Does the software allow combination of cells of different types within the same grid? Which characteristics of grid quality are measured by the software?

6. If your course involves exercises with a CFD code, try automated grid generation for several simple shapes: a pipe of circular cross-section, a spherical ball, a duct of rectangular cross-section, a 2D channel with a backward-facing step, and so on. In each case, create a grid with clustering near the walls. Try different cell shapes and different algorithms of grid generation, if available. Analyze the quality of each grid.

13

CONDUCTING CFD ANALYSIS

In this chapter, we consider the issues that belong to the realms of both common sense and CFD as a scientific discipline. The main question is formulated as follows: *Given the available tools (discretization methods and algorithms for solution of discretization equations and grid generation), how should we conduct the CFD analysis to obtain reliable and useful results?*

13.1 OVERVIEW: SETTING AND SOLVING A CFD PROBLEM

A CFD analysis usually involves much more than simply generating a grid and running an available CFD code. As illustrated in Figure 13.1, there are other essential steps. The actual path of the analysis is usually quite convoluted, full of loops and bifurcations. Nevertheless, the following key stages are always present in some form.

Setting the Physical Model: With the exception of few simple cases, which are rarely encountered outside the classroom, the subject of analysis is usually too complex for a direct CFD approach. Let us, for example, consider the area of more recent application of CFD, the human blood circulation system. Parts of the system, such as arteries, capillaries, and heart muscles, are all connected to each other and with the other organs within the body. The system's behavior involves complex and not always well understood effects: elastic and moving walls, complex blood rheology,

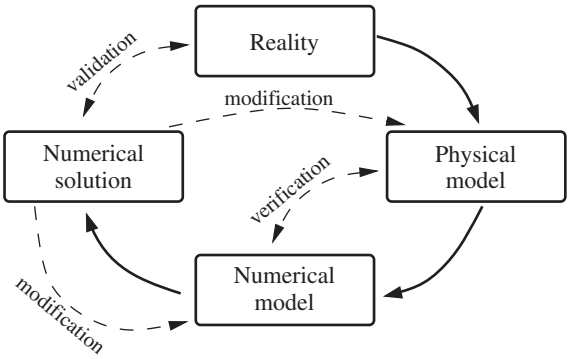


Figure 13.1 Key stages of a CFD analysis.

varying oxygen concentration, and so on. Also, since no two human bodies are the same, each blood circulation system is unique.

Let us now take a position of a researcher trying to understand a certain pathological condition. A numerical simulation of the entire blood circulation system of a sick person is technically impossible. It would also be not very useful, because the critical information on the pathology could easily be lost in the sea of data produced by such a model. Furthermore, the results would correspond to the state of a specific body in a specific period of time. It is not at all certain that this state would reveal the behavior typical for the pathology.

The proper approach to the analysis is to set a *simplified and generalized physical model*. In our example, setting the model starts with locating the particular part of the system, where the pathology—for example, a partially blocked artery—occurs, and limiting the analysis to this part. The influence of rest of the body is reproduced by, hopefully, plausible models implemented in the form of boundary conditions.

The next step is to decide which physical effects have to be included into consideration and which have to be neglected. For example, we can include the complex rheology of blood and the dynamics of plaque buildup or consider a simple flow of an incompressible fluid with constant viscosity and smooth solid walls. In CFD, where the physical model is usually set in terms of partial differential equations expressing the conservation laws, the selection of physical effects is equivalent to the selection of a certain form of the equations.

Quite often, the physical phenomena are important, but cannot be directly addressed in the computations. The most obvious example is turbulence. We can also name multiphase flows and flows with chemical reactions. The approximations used in such cases, for example, the LES

and RANS closure models of turbulence considered in chapter 11, are parts of the general physical model.

One has to assign the values of physical parameters, such as fluid viscosity and density. Furthermore, the CFD analysis has to be conducted for certain 'typical conditions'. In our example, this may mean generalizing the geometry and choosing the regime of operation of the blood system (pulsation frequency and flow rate) critical for the pathology.

The main point of the discussion above is that setting a physical model is unavoidable, often difficult, requires good understanding of the subject of analysis, and can be a significant source of error.

Selecting Numerical Approach: On this stage, we select components and features of the numerical model. This includes discretization method (finite volume, finite difference, etc.), order of approximation, discretization scheme, type of the problem (transient or steady state), type and parameters of the grid (typical space and time steps, clustering, cell shape, etc.), and boundary conditions. Evidently, it is important to make correct choices, since they determine the accuracy of the future solution and the amount of computational work this solution will require.

Developing Numerical Model: In the earlier years of CFD, this was the most difficult and time-consuming part of the analysis. Many thousand lines of code had to be written, tested, and debugged, which took, depending on the qualification of the author and difficulty of the task, from several days to several years. Development of a new algorithm to solve a particular problem still occurs nowadays, but mostly in fundamental research. Practical CFD analysis is predominantly conducted using readily available general-purpose algorithms.

This still leaves serious responsibilities to the user. One of them is the choice of the method to be applied to solution of discretization equations. For example, educated choice of one of the many available iterations methods usually has to be made to achieve reliable and rapid convergence.

Another responsibility appears when the physical model contains features, for which no numerical approximation is available in the CFD code. For example, in the analysis of a blood circulation system, such feature can be the dependence of the apparent viscosity of blood on the diameter of the vessel and flow rate. It is normally possible to implement a new physical feature in the form of a "user-defined" subroutine, an additional segment of the code, which has to be supplied and tested by the user.

Conducting Computations: This is the easiest part of the process. We let computers do the work.

Verification and Validation: There are many possible sources of error in CFD analysis. The only way to increase confidence in the results is thorough testing (verification and validation) discussed in detail in the next section. For now, we only mention that the absence of proper testing is a common and grave mistake leading to sloppy CFD analysis.

Modifications: The CFD analysis rarely succeeds on the first try. Usually, the first results are either absent (convergence is not achieved or the computed fields look physically impossible) or do not pass the verification and validation tests. Modifications of the physical and numerical models have to be made and the computations repeated. This is an iterative procedure, which, if the person conducting the analysis is skillful and lucky, eventually leads to acceptable results. Virtually the only kind of situation in which the modifications can be unnecessary is when the same system is repeatedly analyzed for slightly different regimes of operation.

13.2 ERRORS AND UNCERTAINTY

In theory, we should separate between errors and uncertainty. For example, we can say that the error is a deficiency that occurs not because we do not know something, but because of limitations of our technical (e.g., computer power) capacity. The uncertainty is then characterized as a deficiency occurring because of luck of knowledge. In the following discussion, we use a simplified, albeit theoretically imperfect, approach. We ignore the difference and use the term *error* for all inaccuracies of a CFD analysis.

13.2.1 Errors in CFD Analysis

It is clear from our discussion throughout the book that a CFD solution inevitably contains errors. How can we be confident that the errors are not so large as to render the solution meaningless? Are there ways to estimate and reduce the errors? An attempt to answer these questions is provided next. We start by discussing the types of errors and possible ways to estimate them.

The errors appearing in a CFD analysis can be classified into the following four types:

1. Errors of physical model
2. Discretization errors
3. Errors of solution of discretization equations (iteration errors)
4. Programming errors

Let's look at these error types in more detail.

Errors of Physical Model: As we discussed in section 13.1, a CFD analysis generates an approximate description of the behavior of a *physical model*, rather than of a real physical system. This substitution is inevitably a source of error. We can symbolically write that the values of a property u in the real system and the model differ by

$$\epsilon_{\text{model}} = u_{\text{real}} - u_{\text{model}}. \tag{13.1}$$

There are many reasons why a physical model may behave differently from reality. We will name several of them, which are almost always present.

The partial differential equations that usually constitute a physical model are themselves only imprecise models of real behavior. This does not concern the underlying principles of mass, momentum, and energy conservation, which are exact. The problem is that the Navier-Stokes and heat transfer equations contain the Newton's law (2.16) for viscous stresses and the Fourier law (2.23) for heat flux, both of which are empirical, albeit in many cases quite accurate, approximations. Another, typically much larger source of error, is the modeling approximations we have to use in the equations when dealing with complex phenomena unsuitable for direct numerical analysis. The list is long. It certainly includes turbulence, multiphase flows, and flows with chemical reactions.

Properties of real liquids and gases (e.g., density or viscosity) depend in a complex and not always well documented way on temperature, pressure, and concentration of admixtures. This is often neglected in CFD analysis, and the properties are assumed constant. In the cases where this assumption is abandoned, the error is still introduced, since the properties are approximated by imprecise empirical functions.

Spatially, the physical model often reproduces only a part of a real system. The influence of the rest is imitated by boundary conditions. The imitation is unavoidably artificial and imperfect. One good example is a computational flows domain with an inlet. The inlet conditions are important but usually unknown except, perhaps, for general characteristics, such as the flow rate and average temperature. In the absence of a better alternative, we assume an idealized form of the inlet flow (e.g., constant mean flow and temperature, certain turbulence intensity and dissipation rate) that probably has little in common with reality. Another example of approximate boundary conditions is the temperature conditions (2.48) and (2.50) at perfectly conducting and perfectly insulated walls. Neither of these conditions is realistic. They are only used to avoid solving the conjugate heat transfer problem in the walls.

There is no systematic way to fully predict the magnitude of error produced by physical modeling. Preliminary estimates can, however, be made

on the basis of understanding the physics of the process. For example, knowing the typical magnitudes of flow velocity and variations of temperature and pressure, we can estimate the order of magnitude of density variations and, thus, see if the incompressibility condition is justified. More comprehensive and conclusive estimates are obtained through *validation*, the process of comparison of computed results with data of real system behavior.

Discretization Errors: In the CFD approach, we do not solve the equations of the physical model exactly. Instead, an approximation is found as a solution of a system of algebraic discretization equations. The process introduces the *discretization error*. It is defined as the difference between the exact solution of the model PDE and the exact solution of the discretization equations

$$\epsilon_{\text{discr}} = u_{\text{model}} - u_{\text{discr}}. \tag{13.2}$$

The behavior of the error is determined by the order of the discretization scheme. Let the order of the discretization in the x-direction is p. If the solution of PDE is smooth and Δx is sufficiently small, so that $O((\Delta x)^p)$ is the leading-order term of the truncation error, the discretization error scales as

$$\epsilon_{\text{discr}} = O((\Delta x)^p). \tag{13.3}$$

We can rewrite it as

$$\epsilon_{\text{discr}} = C(\Delta x)^p + O((\Delta x)^{p+1}), \tag{13.4}$$

where C is a generally unknown constant, which depends on the local properties of the solution (C contains its derivatives), properties of the discretization scheme, and design of the grid. Analogous formulas can be written for other coordinates and for time.

Unfortunately, the scaling formulas do not measure the amplitude of the discretization error. The only conclusion that can be made from (13.4) is that the error reduces by approximately m^p times when the grid step is reduced m-fold.

Because of variations of C, the absolute value of the error varies strongly, sometimes by orders of magnitude within the solution. Similar uncertainty exists for integral characteristics. Two schemes of nominally the same order or two differently designed grids used with the same scheme are likely to result in significantly different discretization errors.

The only way to obtain reliable quantitative measure of the amplitude of the discretization error is to compare solutions on systematically refined grids. The logic is simple. Let us assume that a scheme of order p is applied on two grids, which are geometrically similar but have different steps Δx_1 and Δx_2. The results are the two approximate solutions that are related to the exact solution of the physical model as

$$u_{\text{model}} = u_{\text{discr},1} + \epsilon_{\text{discr},1} = u_{\text{discr},1} + C\,(\Delta x_1)^p + O((\Delta x_1)^{p+1})$$
$$u_{\text{model}} = u_{\text{discr},2} + \epsilon_{\text{discr},2} = u_{\text{discr},2} + C\,(\Delta x_2)^p + O((\Delta x_2)^{p+1}).$$

Solving this as a system of linear equations for C and u_{model}, we obtain

$$C = \frac{u_{\text{discr},1} - u_{\text{discr},2}}{(\Delta x_2)^p - (\Delta x_1)^p} + O(\Delta x_1, \Delta x_2) \tag{13.5}$$

and

$$u_{\text{model}} = \frac{(\Delta x_2)^p u_{\text{discr},1} - (\Delta x_1)^p u_{\text{discr},2}}{(\Delta x_2)^p - (\Delta x_1)^p} + O((\Delta x_1)^{p+1}, (\Delta x_2)^{p+1}). \tag{13.6}$$

The discretization error can be estimated, for example, for the solution on the grid with Δx_1, as

$$\begin{aligned}\epsilon_{\text{discr},1} &= \frac{u_{\text{discr},1} - u_{\text{discr},2}}{(\Delta x_2)^p - (\Delta x_1)^p}(\Delta x_1)^p + O((\Delta x_1)^{p+1}, (\Delta x_2)^{p+1}) \\ &\approx \frac{u_{\text{discr},1} - u_{\text{discr},2}}{(\Delta x_2)^p - (\Delta x_1)^p}(\Delta x_1)^p. \end{aligned} \tag{13.7}$$

The first term in the right-hand side of (13.6) can be used to obtain an approximation of the exact solution u_{model}. Together with (13.7), this constitutes the method known as *Richardson extrapolation*.

If the order of discretization p is a-priori unknown, it can be found by conducting computations on yet another grid with step Δx_3 and solving the system of three equations to find C, p, and u_{model}.

Albeit simple on first glance, the method based on the Richardson extrapolation is difficult to implement. The reason is that it requires computations on two or three increasingly fine grids. It is essential for the accuracy of the method that the refinement is significant. The preferred choice is $\Delta x_2 = \Delta x_1/2$. In 3D, this means that the second grid has eight times more points or cells than the first grid. At the same time Δx_1 should be small enough for the asymptotic formula (13.3) to be valid.

Although difficult and time consuming, computations on systematically refined grids have to be done every time the analysis is conducted of a new problem or using a new scheme or new type of grid. The main purpose is not so much to estimate the discretization error, but to determine the level of refinement on which this error becomes smaller than a given tolerance. This level is commonly (and somewhat incorrectly) referred to as the level of *grid independency* of solution. The meaning of the term is that further refinement would change the solution very little.

The grid independency is often determined using the simplified procedure that does not explicitly rely on the Richardson extrapolation. Solutions obtained on systematically refined grids are compared with each other using integral properties and plots of essential characteristics. The solution is declared grid-independent when further refinement does not lead to visible changes.

Errors of Solution of Discretization Equations (Iteration Errors): The algebraic discretization equations are unavoidably solved with errors, which we define as the difference between the exact and actually computed solutions

$$\epsilon_{\text{iter}} = u_{\text{discr}} - u_{\text{comp}}. \tag{13.8}$$

One component of ϵ_{iter} is the round-off error of computer operations. Fortunately, it only becomes important if the scheme is numerically unstable (see Chapter 6). If the scheme is stable, and the round-off errors do not accumulate, their magnitude is typically several orders lower than the magnitude of the errors of other types, for example, of the discretization errors.

Much more significant and constituting practically the entire ϵ_{iter} are the iteration errors that appear when a linear or linearized system of discretization equations is solved by an iteration method (see section 8.3). The iterations cannot be continued indefinitely. They should be stopped when the estimated iteration error becomes smaller than a certain small but nonzero tolerance level.

The two important questions concerning the iteration error are: what tolerance threshold is acceptable, and how can we estimate the error. The answer to the first question is case-specific, although we have to take into account that the iteration errors occur on the background of unavoidable discretization errors and, possibly, model errors. In general, there is no need to continue the iterations to the round-off level of accuracy.[1] The

[1]The only situation in which such ultra-precision may be necessary is the verification testing of an iteration algorithm when it is applied for the first time to the problem. It is desirable in that case to gain confidence that the algorithm does not have internal faults that would prevent convergence below a certain level.

commonly accepted criterion is that the iteration errors are at least one order of magnitude lower than the discretization errors.

The second question is difficult because, as we have already discussed in section 8.3.1, the error cannot be computed directly. Let us, for consistency, return to the notation of Chapter 8 and consider the iteration error $\boldsymbol{\epsilon}^{(k)} = \boldsymbol{v} - \boldsymbol{v}^{(k)}$ of the solution of the matrix equation $\mathbf{A} \cdot \boldsymbol{v} = \boldsymbol{c}$. Evidently, the error is impossible to compute since the exact solution $\boldsymbol{v}$ is unknown. We can only compute indirect characteristics, such as the difference between the results of successive iterations $\boldsymbol{\delta}^{(k)} = \boldsymbol{v}^{(k+1)} - \boldsymbol{v}^{(k)}$ or the residuals $\boldsymbol{r}^{(k)} = \boldsymbol{c} - \mathbf{A} \cdot \boldsymbol{v}^{(k)}$. The usual approach is to calculate the norm $\|\boldsymbol{\delta}^{(k)}\|$ or $\|\boldsymbol{r}^{(k)}\|$ (see (8.27) and (8.28) for definitions of possible norms) and stop the iterations when $\|\boldsymbol{\delta}^{(k)}\|$ or $\|\boldsymbol{r}^{(k)}\|$ becomes smaller than a predetermined tolerance limit ϵ_0.

How do we determine the value of ϵ_0 that secures the desirable level of iteration error? The question is nontrivial, since $\|\boldsymbol{\delta}^{(k)}\|$ or $\|\boldsymbol{r}^{(k)}\|$ do not represent the actual magnitude of $\boldsymbol{\epsilon}^{(k)}$. It can be shown theoretically and confirmed in computational experiments that, after several initial iterations, the dynamics of the error follows

$$\|\boldsymbol{\epsilon}^{(k)}\| \approx A\|\boldsymbol{\delta}^{(k)}\| \approx B\|\boldsymbol{r}^{(k)}\|, \tag{13.9}$$

where A and B are the constant coefficients related to the largest eigenvalue of the iterations matrix. Simply setting ϵ_0 to a small number, say to 10^{-4}, does not guarantee small iteration error, because A and B can be large.

The only reliable fact at our disposal is that $\|\boldsymbol{\epsilon}^{(k)}\|$, $\|\boldsymbol{\delta}^{(k)}\|$, and $\|\boldsymbol{r}^{(k)}\|$ are all reduced at approximately the same rate during the process of convergence. According to (13.9), we can write

$$\frac{\|\boldsymbol{\epsilon}^{(k)}\|}{\|\boldsymbol{\epsilon}^{(0)}\|} \approx \frac{\|\boldsymbol{\delta}^{(k)}\|}{\|\boldsymbol{\delta}^{(0)}\|} \approx \frac{\|\boldsymbol{r}^{(k)}\|}{\|\boldsymbol{r}^{(0)}\|}. \tag{13.10}$$

The initial error $\boldsymbol{\epsilon}^{(0)} = \boldsymbol{v} - \boldsymbol{v}^{(0)}$ is related to the exact solution $\boldsymbol{v}$. $\|\boldsymbol{\epsilon}^{(0)}\|$ is equal to $\|\boldsymbol{v}\|$ if zero initial conditions $\boldsymbol{v}^{(0)} = 0$ are used and is of the same order of magnitude as $\|\boldsymbol{v}\|$ if the initial conditions are attempted as an approximation of $\boldsymbol{v}$. Our final formula

$$\frac{\|\boldsymbol{\epsilon}^{(k)}\|}{\|\boldsymbol{v}\|} \approx \frac{\|\boldsymbol{\delta}^{(k)}\|}{\|\boldsymbol{\delta}^{(0)}\|} \approx \frac{\|\boldsymbol{r}^{(k)}\|}{\|\boldsymbol{r}^{(0)}\|} \tag{13.11}$$

shows that the norm of the iteration error relative to the amplitude of the solution is approximately equal to the observed total reduction of

the norms of differences $\boldsymbol{\delta}^{(k)}$ or residuals $\boldsymbol{r}^{(k)}$. *We have to define* ϵ_0 *as the level at which* $\|\boldsymbol{\delta}^{(k)}\|$ *or* $\|\boldsymbol{r}^{(k)}\|$ *becomes a predetermined fraction of* $\|\boldsymbol{\delta}^{(0)}\|$ *or* $\|\boldsymbol{r}^{(0)}\|$. For example, we can stop the iterations when the norms of residuals fall by four orders of magnitude and be confident that the norm of the iteration error is probably 10^{-4}, but definitely smaller than 10^{-3} of the norm of the solution.

We will complete the discussion by two comments. The sentence, "The residuals are changing only little from one iteration to the next, so I can stop the iterations," is meaningless. The residuals may be changing slowly because the convergence is slow, while the iteration error is still high. The second comment is that the estimate $\|\boldsymbol{\epsilon}^{(0)}\| \approx \|\boldsymbol{v}\|$ becomes an overestimate when the initial conditions are close to $\boldsymbol{v}$. This may happen, in particular, when the results of previous computation, either underconverged or conducted at slightly different parameters or on a slightly different grid, are used as initial conditions. Smaller reduction of $\|\boldsymbol{\delta}^{(k)}\|$ or $\|\boldsymbol{r}^{(k)}\|$ can be required in such cases.

Programming Errors: Programming errors are virtually inevitable and omnipresent. There are many opportunities for a CFD practitioner to make one. Even if a readily available general-purpose code is used, errors can be made while setting the problem (determining the geometry, boundary conditions, physical parameters, etc.) or preparing a user-defined code.

Another, somewhat disturbing fact should be kept in mind. Any CFD code, commercial or noncommercial, contains a significant number of algorithmic errors. The analysis of Hatton (1997) found on average about 10 faulty lines per 1,000 lines of code in more than 100 scientific and engineering codes reviewed. There is no ground to assume that the situation has significantly improved since then.

The programming errors can be roughly divided into two types. There are fatal errors that prevent the code from executing or generate evidently incorrect results. Another type includes "mild" or "sleeping" faults that generate incorrect answers in certain circumstances that may or may not appear in the course of code execution. The errors hidden in the ready-to-use CFD codes usually belong to the second type. The resulting mild incorrectness is very dangerous, since it is usually not clearly visible (one may be tempted to declare that everything is fine, seeing that the iterations have converged and the plots of solution look plausible), but nevertheless real.

There is no systematic way to estimate and control the effect of programming errors. Rather, we have to give full effort to detect as many of them as possible and reach the maximum possible confidence in the results of computations. The methods of verification and validation used for this purpose are discussed in the following section.

13.2.2 Verification and Validation

The previous discussion shows that results of a CFD analysis unavoidably contain errors, the effect of which is either difficult or impossible to determine before or in the course of computations. In this section, we will talk about the additional tools available to detect the errors and evaluate their magnitude. The ultimate goal is to increase the level of confidence in the results.

We should say in the beginning that in CFD there is no universally reliable method of achieving absolute confidence. Rather, the process reminds a criminal investigation (how it appears in books and movies). We collect direct and indirect evidences and modify the assumed picture until a reasonably high level of confidence is achieved that the picture accurately represents reality.

There is a scientific discipline dealing with the general issues of accuracy and reliability of computational modeling. The methods of this discipline were formalized and developed primarily for high-consequence systems, such as weapons or transportation systems or nuclear power stations. Full-scale consistent application of these methods in CFD is a relatively recent phenomenon still largely limited to a few specific areas, such as military systems or weather prediction. However, the basic elements of the accuracy and reliability analysis, perhaps taken informally, have always been considered an essential ingredient of CFD.

The two main tools are verification and validation. The difference between them can be seen in the following definitions, which we quote from the AIAA guidelines published in 1998.

- *Verification:* the process of determining that a model implementation accurately represents the developer's conceptual description of the model and the solution to the model.
- *Validation:* the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model.

These definitions can be rephrased using our terminology, in which *model* becomes *physical model* and *implementation* becomes *implementation of numerical model*, basically, the code and the grid we are using. Figure 13.1 provides an illustration. We can describe verification as the way to ensure that the numerical model solves the physical model accurately. Note that even a fully verified numerical model may produce unrealistic results if the physical model is incorrect. Correctness of the physical model is primarily tested in the course of

validation, which we can describe as the way to test and ensure that the results of a numerically adequate CFD analysis agree with reality.

Considering the types of the CFD errors, we can say that the iteration errors are evaluated and controlled in the course of verification. Physical model errors are addressed by validation. The algorithm errors are detected by both processes. At last, the discretization errors are primarily assessed through verification, although their part related to the effect of numerical resolution on accuracy of turbulence and other closure models requires validation.

Verification: A systematic verification of numerical model should be conducted every time a new code is developed, an essentially new grid is built, or a significant new feature is added to the model. The methodology relies on availability of *reference solutions*, exact analytical or highly accurate numerical solutions of certain benchmark problems. The logic is simple. The best and often only way to find out where and how our numerical model solves the partial differential equations incorrectly is to apply it to a problem for which an exact solution is known and compare the numerical and exact results.

To separate different sources of error, it is recommended to conduct the verification for separate modules of the model, starting at the most elementary level and working way up. For example, developing a new finite volume algorithm, we have to verify that each operation of integral approximation and interpolation is coded correctly and has the desired order of truncation error. This can be done by applying the operations to simple functions (e.g., polynomials), for which integrals and interpolations can be found analytically, and comparing these reference solutions with the numerical results obtained using the values of the same functions at grid points. Similarly, the iteration solver of the matrix equation can be verified by substituting the matrix coefficients and the right-hand side, for which an exact solution is known.

After all modules are verified and assembled or if a ready-to-use code is applied, verification of the entire numerical model is conducted. Since a reference solution is usually unavailable for the system for which we plan to conduct the CFD analysis (the analysis would be unnecessary otherwise), we have to use simplified test cases, in which geometry and flow parameters are modified to make a reference solution possible. For example, if we study a blood flow in an artery with a pathological wall deformity, the good first test cases can be a steady and a pulsating flow in a segment of a circular pipe.

Verification of a numerical model is a complex process characterized by loops and bifurcations, many of which are needed to detect algorithmic

errors. The first of the major steps not related to the algorithmic errors is the analysis of iteration errors. A test case with steady reference solution should be used for this purpose. After running the iterative procedure to round-off accuracy, thus assuring convergence, the iteration error is analyzed in its relation to residuals and differences between the results of successive iterations. This helps to establish useful, albeit not fully reliable, estimates of the convergence criteria.

Another important step is the analysis of discretization errors. This is done by solving the test case equations on successively refined grids and comparing the results with the reference solution. As the outcome of this procedure, we learn whether the discretization errors are reduced at the intended rate, estimate their amplitude as a function of the grid steps, and make conclusions regarding the grid quality.

Validation: The validation is performed after we have gained confidence that the numerical model sufficiently accurately reproduces the behavior of the physical model (this practically means that the numerical model has been successfully verified). The computed solution is compared with the results of experiments. The main purpose of this comparison is to determine the degree of accuracy with which the physical model reproduces the real world. We make the conclusion whether or not the physical model is adequate and evaluate the errors introduced by the model assumptions and approximations, such as the choice of computational domain, artificial boundary conditions, turbulence models, and so on.

Typically, experiments with the entire system considered in a CFD analysis are impossible or impractical. It is, therefore, necessary to replace the system by a simpler prototype or decompose its behavior into smaller and experimentally accessible units. Our example of a blood flow in an artery presents a good case for such *benchmark experiments*, since making detailed measurements within a functioning human body is a difficult and definitely undesirable task. The validation experiments can be conducted in a laboratory using an artificial artery segment, in which flow is generated by a pump. Information on the flow behavior in typical regimes of operation can be collected and compared with the results of computations.

Another difficulty of validation is that the experimental data are, themselves, not free from errors. Evidently, reliable validation is only possible if the experimental uncertainty is known and sufficiently small.

As an additional complication, grid properties (quality and step or cell size) affect the accuracy of certain aspects of the physical model. This concerns, in particular, LES and RANS models of turbulence. The effect cannot be analyzed in the framework of the verification procedure, where the physical model is usually simplified (e.g., assuming that the flow is

laminar) to be able to obtain an exact reference solution. On the contrary, the accuracy of physical approximations on various grids can be assessed in the validation tests through direct comparison with experiments.

13.3 ADAPTIVE GRIDS

We have already seen that the discretization error is, in fact, a local phenomenon. There may exist zones in the solution domain, within which the solution variables have particularly strong gradients. To maintain low discretization error, particularly small grid steps are required within these zones. In the rest of the domain, where the gradients are weak, the same amplitude of the discretization error can be achieved on a cruder grid. Evidently, for the sake of computational efficiency, we have to avoid using a fine grid where it is unnecessary and employ local refinement.

This concept was introduced in section 12.1. We assumed that the location of the refinement zone had been known before the problem was solved. A steady attached boundary layer was used as an example. In this situation, the necessary local refinement could be achieved by designing a correspondingly clustered grid.

There are, however, situations, in which the location of the refinement zone is a-priori unknown. It can only by found as a part of the solution. In some cases, the solution is unsteady and the strong gradient zone changes its location and shape with time.

We will name just two among many such configurations. In flows with shock waves, refinement is required around the shocks, but the shock location and structure are only determined in the course of the solution. Another important and much researched type is that of flows with interfaces between two immiscible media: gas and liquid, two liquids, or liquid and solid. Examples are the unpremixed combustion and solidification. The interface dynamics is, as a rule, a complex, nonsteady, and nonlinear phenomenon coupled with flows, heat and mass transfer in surrounding media. The solution variables usually experience strong gradients near the interface. We need a moving and deformable refinement zone that follows the interface motion and deformation.

Evidently, in the situations such as those just described, a pre-built clustered grid would be useless. We need a method, in which the refinement is made *in the course of computations in correspondence to computed flow fields*. Numerous techniques have been developed for this purpose over the last 20 to 30 years. The common names of *adaptive refinement methods* or *adaptive grid methods* can be used for them. A thorough discussion of these complex methods would only be possible on the level

significantly more advanced than appropriate for our book. The reader interested in the subject is invited to study the research literature. Some useful references are provided at the end of the chapter. Here, we only discuss the basic principles of the approach and briefly outline several established techniques.

The sequence of key steps of an adaptive refinement procedure is obvious:

1. Compute an approximate solution on a grid.
2. Analyze the discretization error and detect the zones where additional refinement is necessary.
3. Rearrange the grid, interpolate the solution onto the new grid, and repeat computations.

In steady-state solutions, the procedure is repeated in iterative manner until a desirably low level of discretization error is achieved throughout the domain. In unsteady solutions with moving refinement zones, the error analysis and grid rearrangement are done between the time steps.

It is the implementation of these obvious steps that makes the adaptive refinement a challenging and advanced technique. One serious issue is how to determine the new location of the refinement zone. We need a sufficiently accurate and robust tool to estimate the local discretization errors, the *error estimator*. This should be done by analyzing the computed solution, although, in some cases, understanding of the physics of the analyzed process can provide helpful indicators.

Perhaps the simplest, but not always reliable, error estimator is the normalized amplitude of the solution gradient. Many estimators have been proposed that evaluate the actual magnitude of the discretization error. Some of them are based on comparison between the solutions on differently refined grids and the Richardson extrapolation (13.7). In others, the necessary information is derived comparing the values of the local solution properties, for example, convective and diffusive flux integrals obtained using schemes of different orders of approximation.

The second major issue is the actual grid refinement and handling solutions on the resulting complex grids. Several distinct approaches have been established. The classification popularized in finite element computations separates p- r- and h-methods. In the p-methods, which are used exclusively in combination with finite elements, the grid remains unmodified, but the order of approximation of the scheme (the number of terms in the Galerkin expansion) is increased within the refinement zone. The

r-methods keep the constant total number of grid points or cells but redistribute them so as to achieve finer resolution within the refinement zone and cruder resolution without it.

In the *h*-method, each cell within the refinement zone is split into a number of smaller cells. For example, 2D quadrilateral cell are usually split into two or four. Several levels of refinement can be used. The traditional finite volume algorithms have to be modified to handle the complexity of the resulting grids. In particular, some faces of larger cells are in contact with faces of two smaller cells. In order to maintain the conservation properties of the method, the face of the larger cell should be treated as a combination of two faces, each representing an interface to a smaller cell. This is possible, but the method should allow a grid consisting of cells with the different numbers of faces.

If the *h*-method is used with a block-structured grid, the refinement is performed block-wise.

A completely original method has been actively applied for solution of hyperbolic systems, for example in supersonic gas dynamics or in astrophysics. The refinement is accomplished through a sequence of overlapping patches. Each patch is covered by a structured and, typically, orthogonal grid. In the solution, each patch is treated quasi-independently. Coupling between the patches and the main grid is achieved using interpolation and other operations.

REFERENCES AND SUGGESTED READING

AIAA. *Guide for the verification and validation of computational fluid dynamics simulations*, AIAA-G-077-1998. Reston, VA: American Institute of Aeronautics and Astronautics, 1998.

Plewa, T., T. Linde, and V. G. Weirs (eds.). *Adaptive Mesh Refinement—Theory and Applications*. New York: Springer, 2005.

Roache, P. J. *Verification and Validation in Computational Science and Engineering*. Hermoza Publishers, 1998.

Thompson, J. F., B. Soni, and N. P. Weatherrill. *Handbook of Grid Generation*. CRC Press, 1998.

Hatton, L. "The T experiments: errors in scientific software," *IEEE Comput. Sci. Eng*. **4** N2 (1997): 27–38.

Oberkampf, W. L., and T. G. Trucano. "Verification and validation in computational fluid dynamics," *Progr. Aerospace Sci*. **38** (2002): 209–272.

INDEX

D

E

F

G

H

I

J

K

L

M

N

P

R

S

T

U

V

W