2007

Lectures in Computational Fluid Dynamics of Incompressible Flow: Mathematics, Algorithms and Implementations

James M. McDonough
University of Kentucky, jmmcd@uky.edu

LECTURES in COMPUTATIONAL FLUID DYNAMICS of INCOMPRESSIBLE FLOW:

Mathematics, Algorithms and Implementations

J. M. McDonough

Departments of Mechanical Engineering and Mathematics
University of Kentucky

PROLOGUE

Computational fluid dynamics (CFD) can be traced to the early attempts to numerically solve the Euler equations in order to predict effects of bomb blast waves following WW II at the beginning of the Cold War. In fact, such efforts were prime drivers in the development of digital computers, and what would ultimately come to be termed *supercomputers*. Such work was motivated further by the "Space Race" with the (former) Soviet Union, beginning in the late 1950s. The terminology "computational fluid dynamics," however, was seldom, if ever, used during this early period; moreover, computational facilities were so inadequate that it was not until the late 1960s that anything even remotely resembling a modern CFD problem could be attempted.

The first book devoted to CFD was written by Patrick Roache during a year-long visit to the Mechanical Engineering Department of the University of Kentucky during 1970–71, and was published the following year [1]. Computing power at that time was still grossly inadequate for what we today would consider useful calculations, but significant efforts in algorithm development and analysis were underway in many leading research universities and national laboratories within the U.S., in Europe (especially France, Great Britain and Sweden) and in the (former) Soviet Union.

Today, at the beginning of the 21^{st} Century, CFD can be viewed as a mature discipline, at least in the sense that it has been universally recognized as essential for engineering analyses associated with transport phenomena, and with regard to the fact that numerous commercial computer codes are now available. Such codes can be executed even on PCs for fairly complex fluid flow problems. Despite this, CFD is not always a part of university curricula and, indeed, it is almost never a *required* component for a degree—primarily because faculty whose backgrounds do not include CFD have yet to retire, and consequently badly-needed curriculum renovations have not been possible.

We have offered elective courses on CFD at both graduate and undergraduate levels for the past 12 years at the University of Kentucky, but like most universities we do not have a formal "CFD Program" in place. Such a program should consist of at least one <u>required</u> undergraduate course in which students would learn to employ a well-known commercial CFD package in solving "real-world" engineering problems involving fluid flow and heat and mass transfer (just has been required in finite-element analysis of structures for many years). Then at least one (but preferably two) graduate classes in computational numerical analysis should be available—possibly through an applied mathematics program. The CFD graduate curriculum, *per se*, should consist of <u>at least</u> the following four one-semester courses: *i*) CFD of incompressible flow, *ii*) CFD of compressible flow, *iii*) turbulence modeling and simulation and *iv*) grid generation. Clearly, a fifth course on computational transport processes and combustion would be very desirable. The two computational numerical analysis courses and the first two CFD classes have been taught at the University of Kentucky since 1990 with an introduction to grid generation provided in the second of the numerical analysis classes, an advanced graduate numerical partial differential equations (PDEs) course.

The present lecture notes correspond to the first item of the above list. They are written to emphasize the mathematics of the Navier–Stokes (N.–S.) equations of incompressible flow and the algorithms that have been developed over the past 30 years for solving them. This author is thoroughly convinced that some background in the mathematics of the N.–S. equations is essential to avoid conducting exhaustive studies (and expending considerable effort in doing so) when the mathematics of the problem shows that the direction being pursued cannot possibly succeed. (We will provide specific examples of this in Chap. 2 of these lectures.) Thus, Chap. 1 is devoted to a fairly advanced presentation of the known theory of the N.–S. equations. The main theorems regarding existence, uniqueness and regularity of solutions will be presented, and put into a computational context, but without proofs. Omission of these (which tend to be extremely technical, mathematically) is for the sake of engineering students, and we will provide essential references from which mathematicians can obtain proofs and other details of the mathematics.

Chapter 2 will be devoted to presentation of a number of basically elementary topics that are specifically related to CFD but yet impact details of the numerical analysis ultimately required to solve the equations of motion (the N.–S. equations). They are often crucial to the successful implementation of CFD codes, but at the same time they are not actually a part of the mathematical aspects of numerical analysis. These topics include the different forms into which the N.–S. equations might be cast prior to discretization, the various possible "griddings" of the physical domain of the problem that can be used in the context of finite-difference, finite-element and finite-volume methods, treatment of the so-called "cell Reynolds number problem" and introduction to "checkerboarding" associated with velocity-pressure decoupling. An understanding of these subjects, along with competence in the numerical analysis of PDEs (a prerequisite for this course) will serve as adequate preparation for analysis and implementation of the algorithms to be introduced in Chap. 3. Unlike what is done in most treatments of CFD, we will view numerical analysis as an essential tool for doing CFD, but not, *per se*, part of CFD itself—so it will <u>not</u> be included in these lectures.

In Chap. 3 we present a (nearly) chronological historical development of the main algorithms employed through the years for solving the incompressible N.–S. equations, starting with the *marker-and-cell* method and continuing through the *SIMPLE* algorithms and modern *projection* methods. We remark that this is one of the main features of the current lectures that is not present in usual treatments. In particular, most CFD courses tend to focus on a single algorithm and proceed to demonstrate its use in various physical problems. But at the level of graduate instruction being targeted by this course we feel it is essential to provide alternative methods. It is said that "those who do not know history are doomed to repeat it," and one can see from the CFD literature that, repeatedly, approaches that long ago were shown to be ineffective continue to resurface—and they still are ineffective but may appear less so because of tremendous increases in computing power since the time of their last introduction. This happens because researchers simply are not aware of earlier attempts with these very "natural" methods. It is this author's opinion that the place to prevent such wasted effort is in the classroom—by presenting both "good" and "bad" (and maybe even "ugly") algorithms so they can be analyzed and compared, leading the student to a comprehensive and fundamental understanding of what will and will not work for solving the incompressible N.–S. equations—<u>and why</u>.

We note at the outset that all details of algorithms will be presented in the context of finite-difference/finite-volume discretizations, and in essentially all cases they will be restricted to two space dimensions. But the algorithms, *per se*, could easily be implemented in a finite-element setting (and in some cases, even for spectral methods). Furthermore, all details and analyses are conceptually easy to transfer to three space dimensions. The actual construction of working codes, however, is much more tedious in 3D, and students are expected to write and debug codes corresponding to various of the algorithms to be presented. So we believe the 2-D treatment is to be preferred.

As hinted above, this set of lectures is intended to be delivered during a (three-unit) one-semester course (or, essentially equivalently, during a four-unit quarter course) to fairly advanced graduate students who are expected to have had at least a first course in general graduate numerical analysis, and should also have had a second course emphasizing numerical PDEs. It is desired to spend as much time as possible focusing on topics that are specific to solution of the incompressible Navier–Stokes equations without having to expend lecture time on more elementary topics, so these are considered to be prerequisites. Lectures on these elements of numerical analysis can be obtained over the Internet as pdf files that can be downloaded by visiting the website http://www.engr.uky.edu/∼acfd and following the links to "lecture notes.".

Contents

List of Figures

Chapter 1

The Navier–Stokes Equations: a mathematical perspective

In this chapter we provide an introduction to the Navier–Stokes equations from a mainly mathematical point of view in order to lay the proper groundwork for the numerical treatments to follow in subsequent chapters. We begin with a very brief discussion of the equations, themselves, reserving details for later, and we provide an outline of the history of the analysis of these equations. But we make no claim as to completeness of this; the intent is merely to indicate the tremendous amount of work that has been done. We follow this with basic functional analysis relating to the Navier–Stokes equations, including such key ideas as "weak" and "strong" solutions. We then state, without proof, the main results concerning existence, uniqueness and regularity of solutions, and we discuss these in the framework of the numerical procedures that will be needed to actually produce solutions.

1.1 Introductory Remarks

As indicated above, this section will consist of presenting the Navier-Stokes equations with cursory remarks regarding the physics embodied in them, and a brief outline of the history of their investigation by, mainly, mathematicians. It will conclude with some remarks on why a basic mathematical study of these equations is important.

1.1.1 The Navier–Stokes equations

The equations of viscous, incompressible fluid flow, known as the Navier–Stokes (N.–S.) equations after the Frenchman (Claude Louis Marie Henri Navier) and Englishman (George Gabriel Stokes) who proposed them in the early to mid 19^{th} Century, can be expressed as

$$\rho\frac{D\boldsymbol{u}}{Dt} = -\nabla p + \mu\Delta\boldsymbol{u} + \boldsymbol{F}_{\!B} \tag{1.1a}$$

$$\nabla\cdot\boldsymbol{u} = 0\,, \tag{1.1b}$$

where ρ is density of the fluid (taken to be a known constant); $\boldsymbol{u} \equiv (u_1, u_2, u_3)^T$ is the velocity vector (which we will often write as $(u, v, w)^T$); p is fluid pressure; μ is viscosity, and $\boldsymbol{F}_{\!B}$ is a body force. D/Dt is the substantial derivative expressing the Lagrangian, or total, acceleration of a fluid parcel in terms of a convenient laboratory-fixed Eulerian reference frame; ∇ is the gradient operator; Δ is the Laplacian, and $\nabla\cdot$ is the divergence operator. We remind the reader that the first of these equations (which is a three-component vector equation) is just Newton's second law of motion applied to a fluid parcel—the left-hand side is mass (per unit volume) times acceleration, while the right-hand side is the sum of forces acting on the fluid element. Equation (1.1b) is simply conservation of mass in the context of constant-density flow. In the sequel we will provide alternative forms of these basic equations.

1.1.2 Brief history of mathematical analyses of the N.–S. equations

It is interesting to note that Navier derived Eq. (1.1a) in 1822 on a very fundamental physical basis including effects of attraction and repulsion of neighboring molecules, but did not specify the physics embodied in the coefficient that ultimately came to be taken as viscosity. Indeed, the effects of molecular interactions might be viewed as being somehow equivalent to viscosity (they are certainly related to it, but Navier could not have known the details of this), but one must not lose sight of the fact that the continuum hypothesis is actually required to make analysis of fluid flow reasonable in the context of any equations less fundamental than the Boltzmann equation (in particular, the N.–S. equations). The derivation by Stokes in 1845 was the first to explicitly introduce viscosity.

During the 20^{th} Century the N.–S. equations received increasing attention from mathematicians following the early studies of Leray [2, 3] in the 1930s in which it was proposed that solutions to these equations could possibly be singular and that this might be associated with turbulence. The work of Leray spurred considerable development of 20^{th} Century functional analysis, and it is often claimed that the N.–S. equations are one of two main progenitors of modern 20^{th} Century mathematical analysis—the other being Schrödinger's equation of quantum mechanics. Following Leray's work was the seminal contribution of Ladyzhenskaya [4] (first available in English in 1963) which provided basic direction and results for further analysis of the N.–S. equations. Beginning in the early 1970s numerous works began to appear that led to a rather different, and modern, view of the equations of fluid motion, namely as *dynamical systems.* The best known of these is probably that of Ruelle and Takens [5] in which it is proposed that the N.–S. equations are capable of describing turbulent fluid motion, that such motion is not random but instead *chaotic* and associated with a *strange attractor* of the *flow* of the dynamical system (the N.–S. equations), and furthermore such behavior must arise after only a small number (possibly as few as three) of bifurcations of the equations of motion.

It is interesting to note that this view of the N.–S. equations was in a sense foreseen in the numerical work of Lorenz [6] in 1963 and the early attempts at direct numerical simulation (DNS) of turbulence by, *e.g.*, Orszag and Patterson [7] in the late 1960s and early 70s. It has in recent years provided the context for numerous works on the Navier–Stokes equations beginning with Ladyzhenskaya [8, 9], Temam [10]–[12], Constantin and Foias [13] and many others. Recently, there have been a number of new monographs published on the Navier–Stokes equations. These include the work by Doering and Gibbon [14] and the very up-to-date volume by Foias *et al.* [15] that will be used extensively in the present chapter. We highly recommend both of these, but especially the latter, to students interested in details of the mathematics that will be somewhat diluted in the current lectures.

1.1.3 Why study mathematics of the N.–S. equations?

Before launching into a mathematical discussion of the N.–S. equations that to some may seem a needless waste of time at best (and maybe worse, completely useless impractical ramblings) we need to provide a rationale for doing so. The first point to be aware of is that in the 21^{st} Century most (probably eventually all) analyses of fluid flow will be performed via CFD. Moreover, such analyses will almost always be carried out using commercial flow codes not written by the user—and possibly (probably, in most cases) not even understood by the user. One of the main goals of these lectures is to provide the reader with sufficient understanding of what is likely to be going on "under the hood" in such codes to motivate caution in accepting computed results as automatically being accurate. At the same time it is hoped that the reader will also develop an ability to test the validity and accuracy of computed results. It will be apparent as we proceed that a fair amount of mathematical understanding of the N.–S. equations, and the nature of their solutions, is required to do this.

Beyond this is the fact, underscored by Eqs. (1.1), that the N.–S. equations comprise a system of PDEs requiring the user of commercial software to prescribe sufficient data associated with any given physical situation to produce a mathematically *well-posed* problem. Commercial CFD codes tend generally to be quite robust, and they will often produce results even for problems that are not properly posed. These can range from easily recognized "garbage out" due to "garbage in" to solutions that look "physically correct" but which are wrong in subtle ways. Clearly, the latter are far more dangerous, but in any case it behooves users of commercial software to know sufficient mathematics of the N.–S. equations to reliably construct well-posed problems—and, sometimes what is required to do so can be counter intuitive based on physics alone.

1.2 Some Basic Functional Analysis

In this section we will introduce some basic notions from fairly advanced mathematics that are essential to anything beyond a superficial understanding of the Navier–Stokes equations. Indeed, as we have already indicated, a not insignificant part of what we will present was developed by mathematicians in their studies of these equations. As might be expected, much of this material is difficult, even for students of mathematics, and as a consequence, we will usually dilute it considerably (but, hopefully, not fatally) in an effort to render it comprehensible to those not especially well trained in modern analysis.

There are numerous key definitions and basic ideas needed for analysis of the N.–S. equations. These include Fourier series, Hilbert (and Sobolev) spaces, the Galerkin procedure, weak and strong solutions, and various notions such as completeness, compactness and convergence. In some cases we will provide precise and detailed definitions, while in others we may present only the basic concept. As often as possible we will also discuss characterizations of the idea under consideration in terms of simpler (hopefully, well-known) mathematics—and/or physics—to clarify these principles and to make them seem more natural.

1.2.1 Fourier series and Hilbert spaces

In this subsection we will briefly introduce several topics required for the study of PDEs in general, and the Navier–Stokes equations in particular, from a modern viewpoint. These will include Fourier series, Hilbert spaces and some basic ideas associated with function spaces, generally.

Fourier Series

We begin by noting that there has long been a tendency, especially among engineers, to believe that only periodic functions can be represented by Fourier series. This misconception apparently arises from insufficient understanding of convergence of series (of functions), and associated with this, the fact that periodic functions are often used in constructing Fourier series. Thus, if we demand *uniform convergence* some sense could be made of the claim that only periodic functions can be represented. But, indeed, it is not possible to impose such a stringent requirement; if it were the case that only periodic functions could be represented (and only uniform convergence accepted), there would be no modern theory of PDEs.

Recall for a function, say $f(x)$, defined for $x \in [0, L]$, that formally its Fourier series is of the form

$$f(x) = \sum_{k}^{\infty} a_k \varphi_k(x) , \tag{1.2}$$

where $\{\varphi_k\}$ is a complete set of basis functions, and

$$a_k = \langle f, \varphi_k \rangle \equiv \int_0^L f(x)\varphi_k(x)\, dx . \tag{1.3}$$

The integral on the right-hand side is a linear functional termed the *inner product* when f and φ_k are in appropriate function spaces. We will later return to requirements on f for existence of such a representation. At this time we can at least discuss some terminology associated with the above expressions.

First, the notion of *basis functions* is analogous to that of basis vectors in linear algebra. For example, in the real N-dimensional Euclidean space $\mathbb{R}^N$ we can construct a basis $\{\boldsymbol{e}_i\}_{i=1}^N$ of the form

$$\boldsymbol{e}_i = (0, 0, \ldots , \overbrace{1}^{i}, 0, \ldots , 0)^T , \tag{1.4}$$

with all components of the vector being zero except the i^{th} one, which has a value of unity. We can then represent <u>every</u> vector $\boldsymbol{v} \in \mathbb{R}^N$ as

$$\boldsymbol{v} = \sum_{i=1}^{N} \langle \boldsymbol{v}, \boldsymbol{e}_i \rangle \boldsymbol{e}_i \,, \tag{1.5}$$

where

$$\boldsymbol{v} = (v_1, v_2, \ldots, v_N)^T \,,$$

and $\langle \boldsymbol{v}, \boldsymbol{e}_i \rangle$ is now just the usual "dot" product of two vectors. (Also note from our choice of the $\boldsymbol{e}_i$s that $\langle \boldsymbol{v}, \boldsymbol{e}_i \rangle = v_i$.) We see from this that we can view Eq. (1.2) as an infinite-dimensional version of (1.5) and, correspondingly, the set $\{\varphi_k\}$ is the infinite-dimensional analogue of the basis set with vectors defined as in Eq. (1.4). But care must be exercised in using such interpretations too literally because there is no question of convergence regarding the series in (1.5) for any $N < \infty$. But convergence (and, moreover, the specific type of convergence) is crucial in making sense of Eq. (1.2). Furthermore, in the finite-dimensional case of Eq. (1.5) it is easy to construct the basis set and to guarantee that it <u>is</u> a basis, using simple results from linear algebra. Treatment of the infinite-dimensional case is less straightforward. In this case we need the idea of *completeness*. Although we will not give a rigorous definition, we characterize this in the following way: if a *countable* basis set is <u>complete</u> (for a particular space of functions, *i.e.*, with respect to a specified norm), then any (and thus, <u>every</u>) function in the space can be constructed in terms of this basis.

We remark that completeness is especially important in the context of representing solutions to differential equations. In particular, in any situation in which the solution is not *a priori* known (and, *e.g.*, might have to be determined numerically), it is essential that the chosen solution representation include a complete basis set. Otherwise, a solution might not be obtained at all from the numerical procedure, or if one is obtained it could be grossly incorrect.

The question of where to find complete basis sets then arises. In the context of Fourier representations being considered at present we simply note without proof (see, *e.g.*, Gustafson [16] for details) that such functions can be obtained as the eigenfunctions of *Sturm–Liouville problems* of the form

$$-\frac{d}{dx}\left(p(x)\frac{d\varphi}{dx}\right) + q(x)\varphi = \lambda r(x)\varphi \,, \qquad x \in [a,b] \tag{1.6a}$$

$$B_a\varphi = B_b\varphi = 0 \,, \tag{1.6b}$$

where B_a and B_b are boundary operators at the respective endpoints a and b of the interval; p, q and r are given functions of x, and λ is an eigenvalue. It is shown in [16] that such problems have solutions consisting of a countable infinity of λ_ks, and associated with these is the countable and complete set of φ_ks.

At this point we see, at least in an heuristic sense, how to construct the series in Eq. (1.2). But we have not yet indicated for what kinds of functions f this might actually be done. To accomplish this we will need a few basic details regarding what are called Hilbert spaces.

Hilbert Spaces

We have already implied at least a loose relationship between the usual N-dimensional vector spaces of linear algebra and the Fourier representation Eq. (1.2). Here we will make this somewhat more precise. We will first introduce the concept of a function space, in general, and then that of a Hilbert space. We follow this with a discussion of the canonical Hilbert space, denoted L^2, and relate this directly to Fourier series. Finally, we extend these notions to Sobolev spaces which, as we will see, are Hilbert spaces with somewhat "nicer" properties than those of L^2 itself—*i.e.*, functions in a Sobolev space are "smoother" (more regular) than those in L^2.

First recall that vector spaces possess the linearity property: if $\boldsymbol{v}$ and $\boldsymbol{w}$ are in the vector space $\mathcal{S}$, for example, $\mathbb{R}^N$ and $a, b \in \mathbb{R}$ (or $a, b \in \mathbb{C}$), then

$$a\boldsymbol{v} + b\boldsymbol{w} \in \mathcal{S} \,.$$

This implies that $\mathcal{S}$ is *closed* with respect to finite linear combinations of its elements.

If we replace the finite-dimensional vectors $\boldsymbol{v}$ and $\boldsymbol{w}$ with functions f and g, both having Fourier representations of the form Eq. (1.2), closedness still holds (*modulo* a few technical details), and we have the beginnings of a function space. But as we have already hinted, because of the infinite basis set, convergence now becomes an issue, and we need tools with which to deal with it. If we consider a sequence of functions $\{f_n(x)\}_{n=0}^{\infty}$, and a possible limit function, say $f(x)$, then we need a way to precisely characterize the heuristic $f_n \to f$ as $n \to \infty$. Recall from elementary analysis that if instead of a sequence of functions we were to consider a sequence of numbers, say $\{a_n\}$, we would say that a_n converges to a, denoted $a_n \to a$, when $|a - a_n| < \epsilon \;\; \forall\, \epsilon > 0$, but depending on n. Furthermore, we remind the reader that it is often essential to be able to consider convergence in a slightly different sense, *viz.*, in the sense of Cauchy: $|a_m - a_n| < \epsilon \;\Rightarrow\; a_n \to a$. This property does not always hold, but the class of sequences for which it does is large, and any space in which convergence in the sense of Cauchy implies convergence (in the usual sense) is said to be *complete.*

The natural generalization of this to sequences of functions is to replace the absolute value $|\cdot|$ with a norm $\|\cdot\|$. We will not at the moment specify how $\|\cdot\|$ is to be calculated (*i.e.*, which norm is to be used), but however this might be done we expect that $\|f - f_n\| < \epsilon$ would imply $f_n \to f$ "in some sense." This provides sufficient machinery for construction of a function space: namely, a *function space* is a complete linear space of functions equipped with a norm. Such spaces are called *Banach spaces*, and they possess somewhat less structure than is needed for our analyses of the N.–S. equations. Independent of this observation, however, is the fact that because there are many different norms, we should expect that the choice of norm is a key part of identifying/characterizing the particular function space. Indeed, for an expression such as $\|f - f_n\| < \epsilon$ to make sense, it must first be true that $\|f - f_n\|$ exists; *i.e.*, it must be that $\|f - f_n\| < \infty$. This leads us to consider some specific examples of norms.

It is worthwhile to first consider a particular vector norm in a finite-dimensional real Euclidean space—the Euclidean length: let $\boldsymbol{v} = (v_1, v_2, \ldots, v_N)^T$; then the Euclidean norm is simply

$$\|\boldsymbol{v}\|_2 = \left(\sum_{i=1}^{N} v_i^2\right)^{1/2} ,$$

which is recognized as the length of $\boldsymbol{v}$. (We have used a "2" subscript notation to specifically distinguish this norm from the infinity of other finite-dimensional p norms defined in an analogous way.) It is also easy to see that the above expression is (the square root of) the vector dot product of $\boldsymbol{v}$ with itself:

$$\|\boldsymbol{v}\|_2^2 = \boldsymbol{v} \cdot \boldsymbol{v} = \langle \boldsymbol{v}, \boldsymbol{v} \rangle ,$$

the last equality representing the notation to be used herein, and which is often termed a *scalar product*—a generalization of inner product. We can see from these last two equations a close relationship between norm and inner (dot) product in Euclidean spaces, and a crucial point is that this same relationship holds for a whole class of (infinite-dimensional) function spaces known as Hilbert spaces. In particular, a *Hilbert space* is any complete normed linear space whose norm is induced by an inner product. Thus, one can think of Hilbert spaces as Banach spaces with more structure arising from the "geometry" of the inner product. (Recall that in finite-dimensional spaces the dot product can be used to define angles; this is true in infinite-dimensional inner product spaces as well.)

As we will see shortly there are many different Hilbert spaces, but we will begin by considering what is termed the "canonical" Hilbert space denoted L^2. L^2 is one member of the family of L^p spaces (all the rest of which are only Banach spaces) having norms defined by

$$\|f\|_{L^p} \equiv \left(\int_\Omega |f|^p \, d\mu\right)^{1/p} , \tag{1.7}$$

where Ω is a domain (bounded, or unbounded) in, say $\mathbb{R}^N$; $|\cdot|$ denotes, generally, the complex modulus (or, if f is real valued, absolute value), and μ is a "measure" on Ω. We will neither define, nor explicity use, the notion of measure to any extent in these lectures, despite its extreme importance in intermediate to advanced analysis, in general. For those readers not familiar with the concept, it will be sufficient for purposes herein to think of *measure* as being an interval (or subinterval, possibly small), and to associate $d\mu$ with the usual differential dx.

The integral in Eq. (1.7) is a *Lebesgue integral* (hence, the notation L^p) which provides a crucial generalization of the *Riemann integral* from elementary calculus. In light of our earlier discussions we see that a function f is in the function space $L^p(\Omega)$ if and only if $\|f\|_{L^p} < \infty$. In particular, $f \in L^2(\Omega)$ whenever $\|f\|_{L^2} < \infty$, where

$$\|f\|_{L^2} \equiv \left(\int_\Omega |f|^2 \, d\mu \right)^{1/2} . \tag{1.8}$$

In the sequel we will consider only real-valued functions f, obviating the need for the $|\cdot|$ notation when considering L^2 (or any other even-p L^p space). In addition, we will replace the Lebesgue measure μ with intervals contained in Ω and revert to the usual dx notation, even when the integral formally must be of Lebesgue type.

There is an important easily-proven inequality, applicable in any Hilbert space, known as the *Cauchy–Schwarz inequality*, and expressed in $L^2(\Omega)$ as

$$\langle f, g \rangle \le \|f\|_{L^2} \|g\|_{L^2} \qquad \forall \; f, g \in L^2(\Omega) . \tag{1.9}$$

Hence, the inner product of two functions in L^2 is bounded by the product of their L^2 norms, both of which are finite. This turns out to be very important in constructing estimates needed for proofs of existence of solutions to PDEs, and holds <u>only</u> in Hilbert spaces. We recognize from this that Hilbert spaces are closed under multiplication of their elements as well as under finite linear combination. A generalization to other L^p spaces, known as *Hölder's inequality*, is similarly important, but more difficult to employ for technical reasons, not the least of which is the fact that closure under multiplication no longer holds.

We now have in place the tools needed to return to the question of what classes of functions may be represented by Fourier series. The basic answer to this is: *any function in L^2 (or any other Hilbert space) can be expressed as a Fourier series.* (There are other functions that also can be so represented, but discussion of this would take us too far from the main topics of these lectures.) There are several related heuristic justifications one can provide for this result. First, we recall that for a function $f(x)$ the formal calculation of its Fourier coefficients is performed with the formula given in Eq. (1.3), repeated here as

$$a_k = \langle f, \varphi_k \rangle \equiv \int_\Omega f(x)\varphi_k(x) \, dx$$

for all k in a countably-infinite index set, and where in (1.3) $\Omega = [0, L]$ was used. We see by comparing this with the Cauchy–Schwarz inequality, Eq. (1.9), that if the φ_ks are only in $L^2(\Omega)$, and f is also in $L^2(\Omega)$, we can guarantee boundedness (and hence, <u>existence</u>) of the a_k.

There are two remarks that should be made at this point. First, in typical cases $\{\varphi_k\}$ lies in nicer spaces than simply $L^2(\Omega)$, and second the preceding observation is not sufficient to guarantee validity of the Fourier representation Eq. (1.2) for L^2 functions. In particular, existence of the inner products that produce the coefficients does not guarantee that they decay sufficiently fast with increasing k to imply convergence of the Fourier series. The *Riemann–Lebesgue lemma* (see any text on beginning graduate analysis), however, provides a suggestion that this must be true; namely, it states that $a_k \to 0$ as $k \to \infty$, even with quite weak hypotheses regarding f and $\{\varphi_k\}$. But the strongest results come from the *Riesz representation theorem* that can be used to deduce both existence of Fourier coefficients and convergence of the resulting series for $f \in L^2(\Omega)$. A yet more elementary way to see this is through the *Parseval identity* which, in the case that the φ_ks are orthonormal, gives

$$\|f\|_{L^2}^2 = \sum_k^\infty |a_k|^2 .$$

Thus, for $f \in L^2(\Omega)$, $\|f\|_{L^2} < \infty \Rightarrow \sum_k^\infty |a_k|^2 < \infty$, and by orthonormality

$$\left\| \sum_k^\infty a_k \varphi_k(x) \right\|_{L^2}^2 = \sum_k^\infty |a_k|^2 < \infty \,.$$

Consequently, the Fourier series converges in the L^2 norm. The texts by Stakgold [17] provide easily-understood discussions of these various concepts.

We emphasize that periodicity of f has never been used in any of these arguments, but we must also note that the above expressions do <u>not</u> imply uniform convergence in Ω. We remark here that by nature of the L^p spaces, convergence in L^2 does not even imply convergence at every point of Ω—in sharp contrast to uniform convergence. This arises because the Lebesgue integral is based on the concept of (Lebesgue) measure, and it is shown in beginning graduate analysis classes that integration over a set of zero measure produces a null result. In turn, this implies that we can ignore sets of measure zero when performing Lebesgue integration; hence, when calculating the L^2 norm, we are permitted to delete regions of "bad behavior" if these regions have zero measure. Furthermore, this leads to the widely-used concept "almost everywhere." When dealing with L^p spaces one often specifies that a given property (*e.g.*, convergence of a Fourier series) holds *almost everywhere*, denoted a.e. This means that the property in question holds everywhere in the domain under consideration except on a set of measure zero.

For later purposes it is useful to note that functions in L^2 are often characterized as having finite energy. This description comes from the fact that, up to scaling, $U^2 \equiv \boldsymbol{U} \cdot \boldsymbol{U}$ is kinetic energy of a velocity field having magnitude U. This notion extends naturally to functions in general, independent of whether they have any direct association with actual physics. Hence, Eq. (1.8) could be associated with (the square root of) the "energy" of the function f, *i.e.*, f^2, and this is clearly finite if $f \in L^2$.

Sobolev Spaces

In the analysis of PDEs, especially nonlinear ones such as the N.–S. equations, it is often necessary to estimate the magnitude of derivatives of the solution and, morever, to bound these derivatives. In order to work more conveniently with such estimates it is useful to define additional function spaces in terms of norms of derivatives. The *Sobolev spaces*, comprise one class of such spaces that can be defined for any L^p space but herein will be associated only with L^2.

To obtain a concise definition of these spaces we will introduce some notation. First, let $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_d)$ denote a vector of d non-negative integers, a *multi-index*, and define $|\alpha| \equiv \alpha_1 + \alpha_2 + \cdots + \alpha_d$. Then we denote the partial differential operator $\partial^{|\alpha|}/\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}$ by D^α. We can now define a general *Sobolev norm* (associated with $L^2(\Omega)$) as

$$\|f\|_{H^m(\Omega)} \equiv \left(\langle f, f \rangle_{H^m(\Omega)} \right)^{1/2} , \tag{1.10}$$

where

$$\langle f, g \rangle_{H^m(\Omega)} \equiv \sum_{|\alpha| \le m} \int_\Omega D^\alpha f D^\alpha g \, dx \,. \tag{1.11}$$

The *Sobolev space*, itself, is defined as

$$H^m(\Omega) = \left\{ f \in L^2(\Omega); D^\alpha f \in L^2(\Omega), |\alpha| \le m \right\} \,. \tag{1.12}$$

Clearly, when $m = 0$, $H^m = H^0 = L^2$; $m = 1$ and $m = 2$ will be the most important cases for the present lectures. The first of these implies that not only is $f \in L^2(\Omega)$, but so also is its first derivative, with analogous statements involving the second derivative when $m = 2$. We remark that these notions and additional ones (especially the Sobolev inequalities) are essential to the modern theory of PDEs, but we will not specifically use anything beyond the results listed above in the present lectures.

1.2.2 Classical, weak and strong solutions to PDEs

In this section we will provide considerable detail on some crucial ideas from modern PDE theory associated with so-called weak and strong solutions, and we will also note how these differ from the "classical" solutions familiar to engineers and physicists. To do this we will need the concept of a distribution (or "generalized" function), and we will also require an introduction to the Galerkin procedure. We begin by making precise what is meant by a "classical" solution to a differential equation to permit making a clear distinction between this and weak and strong solutions. We then provide an example of a weak solution to motivate the need for such a concept and follow this with an introduction to distribution theory, leading to the formal definition of weak solution. Next we show that such solutions can be constructed using the Galerkin procedure, and we conclude the section with a definition of strong solution.

Classical Solutions

It is probably useful at this point to recall some generally-known facts regarding partial differential equations and their (classical) solutions, both to motivate the need for and to better understand the main topics of this section, namely weak and strong solutions. We let $\mathcal{P}(\cdot)$ denote a general partial differential operator, common examples of which are the heat, wave and Laplace operators; $\mathcal{P}$ could be abstract notation for any of these, but also for much more general operators such as the N.–S. operator. To make the initial treatment more concrete we will take $\mathcal{P}$ to be the heat operator:

$$\mathcal{P}(\,\cdot\,) \equiv \left[\frac{\partial}{\partial t} - \kappa\frac{\partial^2}{\partial x^2}\right](\,\cdot\,)\,, \tag{1.13}$$

where $\kappa > 0$ is thermal diffusivity, here taken to be constant.

If we now specify a spatial domain and initial and boundary data we can construct a well-posed problem associated with this operator. For example, if the spatial domain is $\Omega = \mathbb{R}^1$, then no boundary conditions need be given, and the complete heat equation problem can be expressed as

$$\frac{\partial u}{\partial t} = \kappa\frac{\partial^2 u}{\partial x^2}\,, \qquad x \in \Omega \equiv (-\infty, \infty) \tag{1.14a}$$

$$u(x,0) = u_0(x), \qquad x \in \Omega\,, \tag{1.14b}$$

where $u_0(x)$ is <u>given</u> initial data.

A *classical solution* to this problem is one that satisfies Eq. (1.14a) $\forall\ t > 0$, and coincides with $u_0(x)$ at $t = 0\ \forall\ x \in \Omega$. For this to be true (assuming such a solution exists) it is necessary that $u \in C^1$ with respect to time, and $u \in C^2$ with respect to the spatial coordinate; *i.e.*, $u \in C^1(0,\infty)\times C^2(\Omega)$.

As we noted earlier, a modern view of the N.–S. equations is as a dynamical system, and it is worthwhile to introduce this viewpoint here in the well-understood context of the heat equation. From a non-rigorous viewpoint *dynamical system* is simply anything that undergoes <u>evolution in time</u>, and it is clear that Eqs. (1.14) fit this description. Because of the emphasis on temporal behavior it is common to express dynamical systems in the abstract form

$$\frac{du}{dt} = F(u)\,, \tag{1.15a}$$

$$u(0) = u_0\,, \tag{1.15b}$$

even though F may be a (spatial) partial differential operator, and u thus depends also on spatial coordinates.

This leads to a slightly different, but equivalent, view of the solution and corresponding notation associated with the function spaces of which it is a member. Namely, we can think of Eq. (1.15a) as abstractly providing a mapping from the space of functions corresponding to the right-hand side, that is, $C^2(\Omega)$ in the present case, to those of the left-hand side, $C^1(0,\infty)$, and denote this as

$$u(t) \in C^1(0,\infty; C^2(\Omega))\,. \tag{1.16}$$

This notation will be widely used in the sequel, so it is worthwhile to understand what it implies. In words, it says that $u(t)$ is a function that is once continuously differentiable in time on the positive real line, and is obtained as a mapping from the twice continuously differentiable functions in the (spatial) domain Ω.

The final piece of terminology that is readily demonstrated in the context of classical solutions (but which applies in general) is that of solution operator. Recall from elementary PDE theory that Eqs. (1.14) have the exact solution

$$u(x,t) = \frac{1}{\sqrt{4\pi\kappa t}} \int_{-\infty}^{\infty} u_0(\xi) e^{-\frac{(x-\xi)^2}{4\kappa t}}\, d\xi \qquad \forall\ (x,t) \in (-\infty,\infty)\times(0,\infty)\,. \tag{1.17}$$

It is obvious from the form of (1.17) that it is a linear transformation that maps the initial data u_0 to the solution of the PDE at any specified later time, and for all spatial locations of the problem domain Ω. Such a mapping is called a *solution operator*, and the notation

$$u(t) = S(t)u_0 \tag{1.18}$$

is widely used. This corresponds to choosing a time t, substituting it into the *kernel* of the solution operator, $e^{-(x-\xi)^2/4\kappa t}$, and evaluating the integral for all desired spatial locations x. Suppression of the x notation, as in (1.18), can to some extent be related to the dynamical systems viewpoint described above. We remark that derivation of (1.17) is elementary, but it is particularly important because its form permits explicit analysis of *regularity* (*i.e.*, smoothness) of solutions to the problem and, in particular, determination that indeed the solution is of sufficient smoothness to be classical in the sense indicated earlier. Such an analysis is provided in Berg and McGregor [18], among many other references.

A Non-Classical Solution

The concept of weak solution is one of the most important in the modern theory of partial differential equations—basically, the theory could not exist without this idea. It was first introduced by Leray in his studies of the Navier–Stokes equations, but we will present it here first in a more general abstract (and hence, simpler) setting and later apply it to the N.–S. equations.

The basic idea underlying the term *weak solution* is that of a solution to a differential equation that is not sufficiently regular to permit the differentiations required to substitute the solution into the equation. This lack of smoothness might occur on only small subsets of the domain Ω, or it could be present for essentially all of Ω. In any case, there would be parts of the solution domain on which the solution could not be differentiated enough times to satisfy the differential equation in the classical sense.

For those readers who have had no more than a first course in PDEs such ideas may seem rather counterintuitive, at best. Indeed, most solutions studied in a first course are classical; but not all are. Unfortunately, in a beginning treatment the tools needed for analysis of non-classical solutions (either weak or strong) are not available, so no attempts are usually made to address lack of regularity of such solutions. Here we will present a simple example to demonstrate that solutions to PDEs need not always be smooth. This is intended to provide motivation/justification for the discussions of weak and strong solutions that follow.

We consider Poisson's equation on the interior of the unit square Ω with Dirichlet conditions prescribed on the boundary of Ω, denoted $\partial\Omega$:

$$\Delta u = f\,, \qquad (x,y) \in (0,1)\times(0,1) \equiv \Omega\,, \tag{1.19a}$$

$$u(x,y) = g(x,y)\,, \qquad (x,y) \in \partial\Omega\,. \tag{1.19b}$$

The reader can easily check that if

$$f(x,y) = \frac{13}{4}\pi^2 \sin \pi x \sin \frac{3}{2}\pi y\,, \qquad \forall\ (x,y) \in \Omega\,, \tag{1.20}$$

then the function

$$u(x,y) = -\sin \pi x \sin \frac{3}{2}\pi y \tag{1.21}$$

is a solution to the differential equation (1.19a). Clearly this solution is a classical one if the boundary function g satisfies

$$g(x,y) = u(x,y)\Big|_{\partial\Omega}\,. \tag{1.22}$$

That is, $g \equiv 0$ except at $y = 1$, where $g = \sin \pi x$ holds.

But there is no *a priori* reason for such a boundary condition assignment. In fact, we will later work with a problem for the N.–S. equations in which $g \equiv 0$ on all of $\partial\Omega$ except at $y = 1$, where $g \equiv 1$. Figure 1.1 displays these two different boundary data at $y = 1$ to highlight the fact that what would have been a C^∞ (and thus, classical) solution with one set of boundary conditions becomes a much less regular solution near the boundary when a different set of boundary data is employed. The implication is that with the

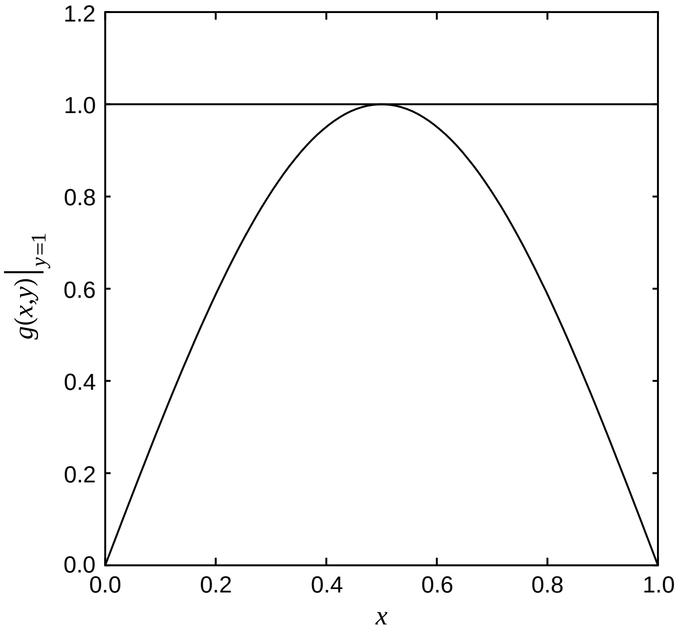

Figure 1.1: Comparison of two boundary condition assignments for the Poisson equation.

nonhomogeneous function f in Eq. (1.19a) given as (1.20), the solution on the interior of Ω should be that given in (1.21). But this solution does not coincide with the second set of boundary conditions considered above. This mismatch between the interior solution produced by an inhomogeneity and the boundary value $g \equiv 1$ on the top boundary of Ω is a form of *incompatibility*; but as already indicated, there is no reason to expect a relationship should exist between f and g of problem (1.19) to prevent this. In general, these two functions will be prescribed completely independently, so it is necessary to have a theory that is sufficiently general to treat the possibility of non-smooth solutions that could result.

There is an additional way to recognize that solutions to problem (1.19) need not be smooth. To consider this we first note that Eq. (1.19a) has a formal solution of the form

$$u(x,y) = \Delta^{-1} f(x,y) = \int_\Omega \tilde{f}(\xi,\eta) G(x,y|\xi,\eta)\, d\xi d\eta\,, \tag{1.23}$$

where $G(\cdot\,|\,\cdot)$ is the *Green's function* for the 2-D Laplacian with (homogeneous) Dirichlet boundary conditions (see, *e.g.*, Stakgold [17] for construction of Green's functions), and $\tilde{f}$ is the original right-hand side function f in Eq. (1.19a) modified to account for transformation of the nonhomogeneous boundary data (1.19b) to the inhomogeneity of the differential equation (see Berg and McGregor [18] for details).

First observe that Eq. (1.23) provides a second example of a solution operator, now acting only on inhomogeneities and boundary data. But the terminology is more widely used in the context of time-dependent problems discussed earlier; in this regard we also note that the kernel of the solution operator in Eq. (1.17) is often called the *causal* *Green's function* to emphasize the fact that it is associated with time-dependent behavior.

We next notice that the right-hand side of Eq. (1.23) is in the form of an inner product. Thus, if G is at least in $L^2(\Omega)$, the Cauchy–Schwarz inequality guarantees boundedness of this integral and, hence, existence of the solution u to problem (1.19) even if $\tilde{f}$ is no smoother than being in $L^2(\Omega)$ would imply. Indeed, this guarantees no smoothness at all—not even continuity in the usual sense. For example, it is easily checked that the function defined by

$$f(x) = \begin{cases} 1\,, & x \text{ irrational} \\ 0\,, & x \text{ rational}\,, \end{cases}$$

for $x \in [0,1]$ is in L^2.

Suppose now that $f \in L^2(\Omega)$, but not any smoother. Then we must ask what this implies regarding regularity of u, the solution to (1.19). From Eq. (1.19a) we see that $f \in L^2(\Omega)$ implies $\Delta u \in L^2(\Omega)$, and hence $u \in H^2(\Omega)$; *i.e.*, the second derivatives of u are in L^2. Thus, the second derivatives of u exist a.e., but may not be smooth—hence, it may be the case that no higher derivatives exist. (Recall that even continuity of the second derivative would not be sufficient to guarantee existence of the third derivative.) We remark, however, that $u \in H^2$ implies existence of Δu a.e., which characterizes a strong, but not classical, solution.

Introduction to Distribution Theory

We emphasize at this point that preceding discussion has been heuristic and lacking precision. In particular, it is possible that u is not even in H^2, implying existence only of what are termed distributional (or weak) derivatives. We will now summarize some key elements from the theory of distributions to allow a more accurate description. We first observe that what is termed a distribution by mathematicians is usually called a "generalized function" by physicists and engineers. Indeed, use of so-called "operational calculus" on such functions, especially by engineers in the early 20^{th} Century, preceded by many years the rigorous distribution theory of Schwartz [19] that ultimately provided justification for such techniques.

The main difficulty with generalized functions is that they are not functions. Hence, even pointwise evaluation cannot necessarily be performed, and differentiation is completely meaningless if viewed in the usual sense. We can see this by examining what is probably the best-known generalized function, the *Dirac* *δ-function* which arises repeatedly in quantum mechanics, but which can also be used to define the Green's function discussed earlier (see, *e.g.*, Stakgold [17] for details). The Dirac δ-function possesses the following properties:

i) it is zero on all of $\mathbb{R}^1$ except at zero, where its value is infinity;

ii) it has the normalization

$$\int_{-\infty}^{\infty} \delta(x)\,dx = 1\,;$$

iii) it exhibits the "sifting property,"

$$\int_{-\infty}^{\infty} f(x)\delta(x-x_0)\,dx = f(x_0)\,.$$

It is clear from the first of the above properties that $\delta(x)$ cannot be a function in any reasonable sense—it is zero everywhere except at a single point, where it is undefined! But the sifting property turns out to be valuable. Our main reason for presenting this function here, however, is to introduce a method of approximating generalized functions that suggests how they might be interpreted, and at the same time can sometimes be valuable in its own right.

When applying operational calculus it is often useful to obtain the δ-function by differentiating the *Heaviside function*, defined as

$$H(x) = \begin{cases} 0 & x < 0\,, \\ 1 & x \geq 0\,. \end{cases}$$

Clearly, this function is not continuous at $x = 0$ and so cannot possibly be differentiable there. So what sense can be made of the notion that the δ-function is the derivative of the Heaviside function? To answer this question we employ what is sometimes called a *δ-sequence* (see Stakgold [17]). The starting point (but not the one used in [17]) in the present case is the hyperbolic tangent function $\tanh kx$. This function is in $C^\infty(\mathbb{R}^1)$ $\forall\ k < \infty$, but it is easily seen (at least heuristically) that

$$\frac{1}{2}\Big[\tanh kx + 1\Big] \to H(x)$$

as $k \to \infty$. Since $\tanh kx$ is differentiable (in fact, C^∞) we can perform the differentiation and then let $k \to \infty$ to find that we obtain (after appropriate normalization) a function with precisely the properties of the δ-function. This suggests that a useful way to define distributions would be in terms of C^∞ functions. As we will now show, however, this is not quite as direct as utilizing δ-sequences, but can be made to be very general.

We begin with some additional notation and terminology. Let C_0^∞ denote the space of infinitely continuously differentiable functions having *compact support*, the latter meaning that such functions vanish outside a bounded interval. It is common in the theory of generalized functions to term functions of this type as *test functions* and denote the space of test functions by $\mathcal{D}$. Next, recall that when we introduced the inner product in Eq. (1.3) we noted that this is in the form of a so-called linear functional. In particular, if neither f nor φ_k of (1.3) is in L^2, then the expression cannot be viewed as an inner product, but instead is a more general *scalar product*, and in the context of Banach spaces this is (loosely—see below) referred to as a linear functional. It will be convenient to define distributions (generalized functions) in terms of linear functionals $\langle\,\cdot\,,\cdot\,\rangle$ such that one of the two entries comes from the space $\mathcal{D}$ of test functions. In particular, as in [17], we state the following.

Definition 1.1 *A linear functional on the space $\mathcal{D}$ of test functions is a mapping f that assigns to each test function $\varphi(x)$ a real number $\langle f, \varphi\rangle$ with the property that $\forall\ a, b \in \mathbb{R}^1$ and $\forall\ \varphi_1, \varphi_2 \in \mathcal{D}$,*

$$\langle f, a\varphi_1 + b\varphi_2\rangle = a\langle f, \varphi_1\rangle + b\langle f, \varphi_2\rangle\,. \tag{1.24}$$

In the present context f need not be a function in the usual sense, and we often apply the term linear functional to f itself, rather than to the number $\langle f, \varphi\rangle$ that it generates when integrated against a test function.

We next define a "weak form of continuity" for linear functionals in the following way. First observe that if $\varphi \equiv 0$, then $\langle f, \varphi\rangle = 0$ for any f that is finite a.e. on its domain of definition. Now let $\{\varphi_n\}$ denote a sequence of functions in $\mathcal{D}$. Then continuity of a linear functional f is defined as follows.

Definition 1.2 *A linear functional f on $\mathcal{D}$ is continuous if and only if $\langle f, \varphi_n\rangle \to 0$ whenever $\varphi_n \to 0$.*

Observe that in contrast to the pointwise definition for continuity of functions, continuity of a functional is defined in terms of an integral and is thus often termed *weak continuity.* Moreover, a sequence of (test) functions is required to check continuity of a functional. Also note, however, that for each n $\langle f, \varphi_n\rangle$ is a real number (from the definition of functional), so the test of continuity of a functional is equivalent to checking convergence (to zero) of a sequence of real numbers. We remark that, conceptually, this is not extremely different from the δ-ϵ definition of continuity in elementary analysis where we must permit $\delta > 0$ to be arbitrarily small. Here, we would simply use $\|\varphi\| < \delta$ with $\|\cdot\|$ being the uniform norm.

We have now accumulated sufficient information to be able to give a precise meaning to the term *distribution*; in particular, we have the following:

Definition 1.3 *A distribution is a continuous linear functional on the space $\mathcal{D}$ of test functions.*

Because of linearity we have not only (1.24), but also that $\forall\ \varphi \in \mathcal{D}$ and $a, b \in \mathbb{R}^1$ (or in $\mathbb{C}$) we can define new distributions as linear combinations of other distributions. For example, suppose f_1 and f_2 are distributions. Then $f \equiv af_1 + bf_2$ is also a distribution for we have

$$\langle f, \varphi\rangle = \langle af_1 + bf_2, \varphi\rangle = a\langle f_1, \varphi\rangle + b\langle f_2, \varphi\rangle\,. \tag{1.25}$$

Finally, we remark that not all distributions are defined via linear functionals. (Recall that no test function was used to define the Dirac δ-function.) More details regarding this can be found in [17], but we note that for our purposes Def. 1.3 will usually suffice.

From the standpoint of understanding weak solutions to differential equations, the most important operation associated with distributions clearly must be differentiation. (There are several other operations, and we refer the reader to [17] for discussions of these.) Indeed, it was specifically the possibility that solutions might exist that were not sufficiently differentiable to satisfy the differential equation that has motivated these discussions. But because distributions are defined in terms of integrals, we first need to introduce some terminology associated with these.

Definition 1.4 *A function is said to be integrable on $\mathbb{R}$ if*

$$\int_{-\infty}^{\infty} |f|\,dx < \infty\,. \tag{1.26}$$

From Eq. (1.7) it is clear that this implies $f \in L^1(\mathbb{R})$. In the sense of Riemann integration from elementary analysis we might say that f is "absolutely integrable." It is possible, of course, that the integral over $\mathbb{R}$ might not exist in the sense of Eq. (1.26), but that integrals over subsets of $\mathbb{R}$ might still be finite. Associated with this is the concept of local integrability. We say a function f is *locally integrable*, denoted here as L^1_{loc}, if for <u>every</u> finite subset $\Omega \subset \mathbb{R}$ we have

$$\int_{\Omega} |f|\,dx < \infty\,, \tag{1.27}$$

even though integrability in the sense of (1.26) might not hold. We can construct analogous definitions corresponding to any of the other L^p spaces (and the H^m spaces, as well), but this is not needed for our immediate purposes.

We can now state a lemma that is important in constructing derivatives of distributions.

Lemma 1.1 *<u>Every</u> locally-integrable function generates a distribution.*

This is easily proven since local integrability along with compact support of test functions is sufficient to guarantee (weak) continuity of the associated linear functional.

Now let f be a differentiable <u>function</u> whose derivative f' is in L^1_{loc}. In addition, let $\varphi \in \mathcal{D}$. From the preceding statement we see that f' generates the distribution

$$\begin{aligned}\langle f', \varphi\rangle &\equiv \int_{-\infty}^{\infty} f'(x)\varphi(x)\,dx\\ &= f\varphi\Big|_{-\infty}^{\infty} - \int_{-\infty}^{\infty} f(x)\varphi'(x)\,dx\,.\end{aligned}$$

But since $\varphi \in \mathcal{D}$ has compact support, we see that

$$f\varphi\Big|_{-\infty}^{\infty} = 0\,.$$

This suggests how we can construct the derivative of a distribution. Namely, now let f be any distribution associated with a linear functional, and write

$$\langle f', \varphi \rangle = -\langle f, \varphi' \rangle\,. \tag{1.28}$$

That is, we have used integration by parts to move (formal) differentiation off of a generally non-differentiable distribution and onto a C_0^∞ test function. Again, the compact support of φ removes the need to deal with boundary terms that arise during integration by parts. Furthermore, it is clear that the right-hand side of (1.28) is also a distribution since φ' is also a valid test function.

The derivative calculated in Eq. (1.28) in general possesses no pointwise values. It is termed a *distributional*, or *weak*, derivative because it is defined only in terms of integration over the support of the test function φ. We also note that higher distributional derivatives can be calculated via repeated integrations by parts and satisfy the general formula

$$\langle f^{(n)}, \varphi \rangle = (-1)^n \langle f, \varphi^{(n)} \rangle \tag{1.29}$$

for the n^{th} distributional derivative of f, as the reader may easily demonstrate..

An important—and in some sense surprising—observation is that distributions are infinitely differentiable in the weak sense, even though they may have no derivatives at all (in fact, might not even be continuous) in the pointwise sense. This follows from the fact that the test functions, by definition, are in C^∞, and it is only these that are actually differentiated when performing distributional differentiation.

Weak Solutions—in General

We now have the tools needed to consider weak solutions to differential equations. There is one additional piece of terminology required for a concise statement, and it furthermore has much wider application than we will utilize herein. It is the (formal) adjoint of a differential operator. We will explicitly consider only spatial operators, but the basic ideas can be extended to more general cases. Let

$$L(\cdot) \equiv \left[\sum_{k=0}^{n} a_k(x)\frac{d^k}{dx^k}\right](\cdot) \tag{1.30}$$

be a general variable-coefficient n^{th}-order linear operator. Then the *formal adjoint* of L, denoted L^*, is obtained by forming a linear functional—simply an inner product in Hilbert spaces, integrating by parts, and ignoring boundary terms. This leads to

$$L^*(\cdot) \equiv \sum_{k=0}^{n} (-1)^k \frac{d^k(a_k(x)\,\cdot)}{dx^k}\,., \tag{1.31}$$

as the reader is encouraged to demonstrate.

Because the adjoint is such an important concept, in general, it is worthwhile to consider constructing one in a specific case.

EXAMPLE 1.1: Consider the second-order linear operator

$$L = a(x)\frac{d^2}{dx^2} + b(x)\frac{d}{dx} + c(x)\,,$$

operating on C^2 functions that do not necessarily possess compact support. As noted above, we begin by forming the inner product

$$\begin{aligned}\langle Lu, v\rangle &= \int \big(a(x)u'' + b(x)u' + c(x)u\big)\, v\, dx \\ &= \int a(x)u''v\, dx + \int b(x)u'v\, dx + \int c(x)uv\, dx\,,\end{aligned}$$

for v also in C^2 (and not possessing compact support, in general). Application of integration by parts (twice) to the first integral on the right-hand side of the above produces

$$\int au''v = au'v\Big| - u(av)'\Big| + \int (av)''u\,.$$

Similarly, from the second integral we obtain

$$\int bu'v = buv\Big| - \int (bv)'u\,,$$

and the third integral remains unchanged.

Now disregard the boundary terms, and collect the above results to obtain

$$\begin{aligned}\langle Lu, v\rangle &= \langle u, (av)''\rangle - \langle u, (bv)'\rangle + \langle u, (cv)\rangle \\ &= \langle u, L^*v\rangle\,,\end{aligned}$$

with the last line following from the definition/notation of Eq. (1.31).

There are three things to note at this point. First, if $\langle u, v\rangle$ had been a distribution on $v \in \mathcal{D}$, the the boundary terms would have automatically vanished due to the compact support of v. Second, if the operator L were part of a boundary-value problem, then the boundary terms arising in the integrations by parts would be used to determine the *adjoint boundary conditions* for the corresponding adjoint boundary-value problem. In this situation, if it happens that $L^* = L$ and the adjoint boundary conditions are of the same form as those for L, the boundary-value problem is said to be *self adjoint*. In an abstract sense this is analogous to symmetry of a matrix, and indeed, the matrices arising from, *e.g.*, finite-difference, discretization of such problems are symmetric. Finally, we note that analogous constructions can be carried out in higher spatial dimensions.

With this notion of adjoint operators in hand, we can now give a precise definition of weak solution to a differential equation.

Definition 1.5 *Let $\mathcal{P}$ be a differential operator with formal adjoint $\mathcal{P}^*$, and let f be a distribution. Then a weak solution to the differential equation*

$$\mathcal{P}u = f$$

is any distribution u that satisfies

$$\langle u, \mathcal{P}^*\varphi\rangle = \langle f, \varphi\rangle \qquad \forall\ \varphi \in \mathcal{D}\,. \tag{1.32}$$

We remark that while such definitions are extremely important for theoretical purposes, they are of little practical (computational) use, in general, with the main exception being finite-element methods based on variational principles. In particular, this is a global (integral) rather than pointwise result, and it formally must be tested $\forall\ \varphi \in \mathcal{D}$. Thus, we need to investigate how weak solutions might be generated via numerical techniques.

The Galerkin Procedure

The best-known approach to obtaining weak solutions via numerical methods is the Galerkin procedure. This technique is also very important in theoretical analyses, especially for the N.–S. equations, as will become evident below. Here we will provide a somewhat general and abstract treatment, and later study the N.–S. equations as a specific example.

The *Galerkin procedure* can be formally constructed via the following sequence of steps.

i) Represent each dependent variable and forcing term of the differential equation (DE) as a Fourier series with basis functions depending on spatial coordinates. (Note: if the DE is not time dependent, the Fourier coefficients are constants; otherwise, they are functions of time.)

ii) Substitute these solution representations into the DE.

iii) Commute differentiation and series summation for each appropriate term, and differentiate the basis functions and/or Fourier coefficients, as appropriate.

iv) Form inner products with the DE and every member of the basis set. This will formally result in a countably-infinite system of equations (either algebraic or differential initial-value) for the Fourier coefficients.

v) Truncate this system to a finite number N, and solve (or analyze—this may be all that is possible in nonlinear cases) the result for the Fourier coefficients.

We emphasize that it is merely the Fourier coefficients that are to be determined from any such numerical procedure, and once these are found they can be substituted into the solution representation to obtain the solution itself. Furthermore, as we have emphasized earlier, all that is required for existence of such solutions is that they be in L^2, so in general they are weak in the sense indicated above; *i.e.*, they are not necessarily sufficiently smooth to satisfy the differential equation in the classical sense. Moreover, as will be clear, these will be solutions to integral forms of the original DE. On the other hand, in contrast with the general situation for weak solutions, pointwise evaluation will now be possible.

In the following examples we will demonstrate application of the Galerkin procedure for two specific cases, both of which have particular importance with respect to the N.–S. equations. The first will be linear and independent of time, corresponding to what we will later term a pressure Poisson equation in the N.–S. context. The second will be nonlinear and time dependent and thus, in a generic way, analogous to the N.–S. equations themselves.

EXAMPLE 1.2: Let the partial differential operator $\mathcal{P}$ be the Laplacian in two space dimensions, and consider

$$\Delta u = f(x,y)\,, \qquad (x,y) \in \Omega \subset \mathbb{R}^2\,,$$

with boundary conditions

$$u(x,y) = g(x,y)\,, \qquad (x,y) \in \partial\Omega\,.$$

We assume these nonhomogeneous conditions can be transformed to the differential equation (see, *e.g.*, [18]) so that we actually solve the related problem

$$\Delta v = \tilde{f}(x,y)\,, \qquad (x,y) \in \Omega\,,$$

with boundary conditions

$$v(x,y) = 0\,, \qquad (x,y) \in \partial\Omega\,.$$

We assume $\tilde{f} \in L^2(\overline{\Omega})$ and choose a complete (in $L^2(\Omega)$) orthonormal basis $\{\varphi_{\boldsymbol{k}}\}$ vanishing on $\partial\Omega$. Then we express v and $\tilde{f}$ as the Fourier series

$$\begin{aligned} v(x,y) &= \sum_{\boldsymbol{k}} a_{\boldsymbol{k}}\varphi_{\boldsymbol{k}}(x,y)\,, \\ \tilde{f}(x,y) &= \sum_{\boldsymbol{k}} b_{\boldsymbol{k}}\varphi_{\boldsymbol{k}}(x,y)\,, \end{aligned}$$

respectively, with $\boldsymbol{k} = (k_1, k_2)^T$.

Next, substitute these series into the differential equation:

$$\frac{\partial^2}{\partial x^2}\left(\sum a_{\boldsymbol{k}}\varphi_{\boldsymbol{k}}\right) + \frac{\partial^2}{\partial y^2}\left(\sum a_{\boldsymbol{k}}\varphi_{\boldsymbol{k}}\right) = \sum b_{\boldsymbol{k}}\varphi_{\boldsymbol{k}}\,.$$

Then commuting differentiation and summation leads to

$$\sum a_{\boldsymbol{k}}\left(\varphi_{\boldsymbol{k}}\right)_{xx} + \sum a_{\boldsymbol{k}}\left(\varphi_{\boldsymbol{k}}\right)_{yy} = \sum b_{\boldsymbol{k}}\varphi_{\boldsymbol{k}}\,,$$

and we note that subscript notation such as xx indicates partial differentiation. Since the $b_{\boldsymbol{k}}$s all are known the only unknowns are the $a_{\boldsymbol{k}}$s. To find these we first construct the inner products

$$\left\langle \sum a_{\boldsymbol{k}}\left[\left(\varphi_{\boldsymbol{k}}\right)_{xx} + \left(\varphi_{\boldsymbol{k}}\right)_{yy}\right], \varphi_{\boldsymbol{n}}\right\rangle = \left\langle \sum b_{\boldsymbol{k}}\varphi_{\boldsymbol{k}}, \varphi_{\boldsymbol{n}}\right\rangle\,,$$

$\forall\ \boldsymbol{n} = \boldsymbol{n}_0, \ldots, \infty$. We observe that the specific value of $\boldsymbol{n}_0$ must be the same as the minimum $\boldsymbol{k}$, which typically will be one of $-\infty$, 0, 1, depending on details of the particular problem and selected basis set.

We will now assume, for simplicity, that derivatives of the $\varphi_{\boldsymbol{k}}$s also exhibit orthogonality but not necessarily the same normality, and that they behave like complex exponentials with respect to differentiation. It should be noted that this usually is not the case, but that the widely-used complex exponentials and trigonometric functions obviously exhibit this behavior. This implies that in the above summations only the n^{th} term will be nonzero, and we obtain

$$\left[C_1^* n_1^2 \langle\varphi_{\boldsymbol{n}}, \varphi_{\boldsymbol{n}}\rangle + C_2^* n_2^2 \langle\varphi_{\boldsymbol{n}}, \varphi_{\boldsymbol{n}}\rangle\right] a_{\boldsymbol{n}} = b_{\boldsymbol{n}}\,, \qquad \forall\ \boldsymbol{n}\,,$$

with the C_i^*, $i = 1, 2$ depending on length scales associated with Ω, and in particular arising due to possible non-normality of derivatives of basis functions and defined to include any sign changes arising from differentiation of basis functions. We can now solve for the $a_{\boldsymbol{n}}$s and obtain the exact solution

$$a_{\boldsymbol{n}} = \frac{b_{\boldsymbol{n}}}{C_1^* n_1^2 + C_2^* n_2^2}\,,$$

for $\boldsymbol{n} = \boldsymbol{n}_0, \ldots, \infty$. Notice that in this simple linear case there was no need to truncate the representations prior to solving for the Fourier coefficients because these are completely decoupled and satisfy the given general formula.

The above result can now be substituted back into the Fourier series for v, which can then be transformed back to the desired result, u. We leave as an exercise to the reader construction of this transformation, but we comment that if g is not at least in $C^2(\Omega)$, the derivatives required for this construction will truly be only distributional. It should be recalled that this was, in fact, the case for a Poisson equation problem treated earlier.

As a further example of constructing a Galerkin procedure we consider a time-dependent nonlinear problem which is, in a sense, a generalization of typical problems associated with the N.–S. equations.

EXAMPLE 1.3: We now let

$$\mathcal{P} \equiv \frac{\partial}{\partial t} + \mathcal{N} + \mathcal{L}\,,$$

where $\mathcal{N}$ is a nonlinear operator, and $\mathcal{L}$ is linear, and consider the initial boundary value problem given as follows.

$$\mathcal{P}u = \frac{\partial u}{\partial t} + \mathcal{N}(u) + \mathcal{L}u = f\,, \qquad (x, y) \in \Omega \subset \mathbb{R}^2\,, \quad t \in (0, t_f]\,,$$

with initial conditions

$$u(x, y, 0) = u_0(x, y)\,, \qquad (x, y) \in \Omega\,,$$

and boundary conditions

$$u(x, y, t) = 0\,, \qquad (x, y) \in \partial\Omega \quad \forall\ t\,.$$

In this example we have assumed homogeneous Dirichlet conditions from the start. We again begin by expanding u and f in the Fourier series

$$\begin{aligned} u(x,y,t) &= \sum_{\boldsymbol{k}} a_{\boldsymbol{k}}(t)\varphi_{\boldsymbol{k}}(x,y)\,, \\ f(x,y,t) &= \sum_{\boldsymbol{k}} b_{\boldsymbol{k}}(t)\varphi_{\boldsymbol{k}}(x,y)\,. \end{aligned}$$

We note that if it happens that f is independent of time, then the $b_{\boldsymbol{k}}$s will be constants. In either case, since $f \in L^2(\Omega)$ is a given function we can calculate the $b_{\boldsymbol{k}}$s directly as

$$b_{\boldsymbol{k}}(t) = \int_\Omega f(x,y,t)\varphi_{\boldsymbol{k}}(x,y)\,dxdy\,,$$

in which t is formally just a parameter. We remark here that our treatment of f is not precise. In particular, it would be possible for f to be in L^2 spatially but exhibit less regularity in time. We are tacitly assuming this is not the case.

Substitution of these representations into the differential equation results in

$$\frac{\partial}{\partial t}\sum a_{\boldsymbol{k}}(t)\varphi_{\boldsymbol{k}}(x,y) + \mathcal{N}\left(\sum a_{\boldsymbol{k}}(t)\varphi_{\boldsymbol{k}}(x,y)\right) + \mathcal{L}\sum a_{\boldsymbol{k}}(t)\varphi_{\boldsymbol{k}}(x,y) = \sum b_{\boldsymbol{k}}(t)\varphi_{\boldsymbol{k}}(x,y)\,,$$

or, after commuting differentiation and summation,

$$\sum \dot{a}_{\boldsymbol{k}}\varphi_{\boldsymbol{k}} + \mathcal{N}\left(\sum a_{\boldsymbol{k}}\varphi_{\boldsymbol{k}}\right) + \sum a_{\boldsymbol{k}}\mathcal{L}\varphi_{\boldsymbol{k}} = \sum b_{\boldsymbol{k}}\varphi_{\boldsymbol{k}}\,.$$

Here, we should note that formal differentiation of the $a_{\boldsymbol{k}}$s with respect to time is done without proof that such temporal derivatives exist. Clearly, this will be influenced, at least, by the nature of f and that of any time-dependent boundary conditions.

We observe that the nonlinear operator $\mathcal{N}(u)$ can appear in many different forms, and clearly details of its treatment depend crucially on the specific form. We consider two examples here to demonstrate the range of possibilities. First, if $\mathcal{N}$ is quasilinear, as occurs in the N.–S. equations, then, *e.g.*,

$$\mathcal{N}(u,u) = u\frac{\partial u}{\partial x}\,,$$

and it follows that

$$\begin{aligned} \mathcal{N}(u,u) &= \left(\sum_{\boldsymbol{k}} a_{\boldsymbol{k}}\varphi_{\boldsymbol{k}}\right)\frac{\partial}{\partial x}\left(\sum_{\boldsymbol{\ell}} a_{\boldsymbol{\ell}}\varphi_{\boldsymbol{\ell}}\right) \\ &= \sum_{\boldsymbol{k},\boldsymbol{\ell}} a_{\boldsymbol{k}}a_{\boldsymbol{\ell}}\varphi_{\boldsymbol{k}}\varphi_{\boldsymbol{\ell},x}\,. \end{aligned}$$

The important feature of this case is that although the Fourier coefficients appear nonlinearly, it is nevertheless possible to express the complete equation in terms only of these (and, of course, the various constants that arise during subsequent construction of required inner products), as is readily seen by substituting this expression into the above ordinary differential equation (ODE), and computing the inner products. We leave this as an exercise for the reader.

By way of contrast, consider the nonlinearity that arises in Arrhenius-like reaction rate terms in the equations of combustion chemistry:

$$\mathcal{N}(u) = Au^\beta \exp\left(\frac{-C}{u}\right)\,,$$

where A, β and C are known constants for any particular chemical reaction, and the dependent variable u is now temperature. Substitution of the Fourier series yields

$$\mathcal{N}\left(\sum_{\boldsymbol{k}} a_{\boldsymbol{k}}\varphi_{\boldsymbol{k}}\right) = A\left(\sum_{\boldsymbol{k}} a_{\boldsymbol{k}}\varphi_{\boldsymbol{k}}\right)^{\beta} \exp\left(\frac{-C}{\sum_{\boldsymbol{k}} a_{\boldsymbol{k}}\varphi_{\boldsymbol{k}}}\right) .$$

It is often the case that β is not a rational number, and it can be either positive or negative. Thus, there is no means of separating $a_{\boldsymbol{k}}$ from $\varphi_{\boldsymbol{k}}$ in this factor; an even more complicated situation arises for the exp factor. This leads to two related consequences. First, inner products must be calculated via numerical methods (rather than analytically) and second, after each nonlinear iteration for finding the $a_{\boldsymbol{k}}$s, the solution representation for u must be constructed for use in evaluating the nonlinear terms. These difficulties are of sufficient significance to generally preclude use of Galerkin procedures for problems involving such nonlinearities. Nevertheless, we will retain the full nonlinear formalism in the present example to emphasize these difficulties.

If, as in the preceding example, we assume the $\varphi_{\boldsymbol{k}}$s are not only orthonormal but also are orthogonal to all their derivatives, then construction of Galerkin inner products leads to

$$\left\langle \sum_{\boldsymbol{k}} \dot{a}_{\boldsymbol{k}}, \varphi_{\boldsymbol{n}} \right\rangle + \left\langle \mathcal{N}\left(\sum_{\boldsymbol{k}} a_{\boldsymbol{k}}\varphi_{\boldsymbol{k}}\right), \varphi_{\boldsymbol{n}} \right\rangle + \left\langle \sum_{\boldsymbol{k}} a_{\boldsymbol{k}}\mathcal{L}\varphi_{\boldsymbol{k}}, \varphi_{\boldsymbol{n}} \right\rangle = \left\langle \sum_{\boldsymbol{k}} b_{\boldsymbol{k}}\varphi_{\boldsymbol{k}}, \varphi_{\boldsymbol{n}} \right\rangle ,$$

or, upon invoking orthogonality of the $\varphi_{\boldsymbol{k}}$s,

$$\dot{a}_{\boldsymbol{n}} + \left\langle \mathcal{N}\left(\sum_{\boldsymbol{k}} a_{\boldsymbol{k}}\varphi_{\boldsymbol{k}}\right), \varphi_{\boldsymbol{n}} \right\rangle + a_{\boldsymbol{n}}\left\langle \mathcal{L}\varphi_{\boldsymbol{n}}, \varphi_{\boldsymbol{n}} \right\rangle = b_{\boldsymbol{n}} , \qquad \forall\ \boldsymbol{n}_0, \ldots, \infty .$$

We again emphasize that the $\varphi_{\boldsymbol{n}}$ and $b_{\boldsymbol{n}}$ are known, so the above comprises an infinite system of nonlinear ODE initial-value problems often termed the *Galerkin ODEs*. It is thus necessary to truncate this after $N = N_1 \times N_2$ equations, where $\boldsymbol{N} = (N_1, N_2)^T$ is the highest wavevector considered (*i.e.*, N terms in the Fourier representation). For definiteness take $n_0 = 1$. Then we can express the problem for finding the $a_{\boldsymbol{n}}$s as

$$\dot{a}_{\boldsymbol{n}} + \left\langle \mathcal{N}\left(\sum_{\boldsymbol{k}} a_{\boldsymbol{k}}\varphi_{\boldsymbol{k}}\right), \varphi_{\boldsymbol{n}} \right\rangle + A_{\boldsymbol{n}}a_{\boldsymbol{n}} = b_{\boldsymbol{n}} , \qquad n = 1, \ldots, N,$$

with

$$a_{\boldsymbol{n}}(0) \equiv \langle u_0, \varphi_{\boldsymbol{n}} \rangle ,$$

and

$$A_{\boldsymbol{n}} \equiv \langle \mathcal{L}\varphi_{\boldsymbol{n}}, \varphi_{\boldsymbol{n}} \rangle ,$$

Clearly, it is straightforward, in principle, to solve this system if the nonlinear operator $\mathcal{N}$ is not extremely complicated, after which we obtain a N-term approximation for u:

$$u_N(x, y, t) = \sum_{\boldsymbol{k}}^{N} a_{\boldsymbol{k}}(t)\varphi_{\boldsymbol{k}}(x, y) .$$

But as we have already noted, if $\mathcal{N}$ is a general nonlinearity, evaluation of the second term on the left in the above ODEs, possibly multiple times per time step, can be prohibitively CPU time consuming.

In any case, just as was true in the preceding example, the fact that the solution is in the form of a Fourier series guarantees, *a priori*, that it might be no more regular than L^2. Hence, it may satisfy the original PDE only in the weak sense.

In closing this discussion of the Galerkin procedure as a means of generating weak solutions to differential equations, we first observe that although there are formal similarities between this approach and the Def. 1.5 of a weak solution, they are not identical. The general linear functional of the definition has been replaced with an inner product (a special case) in the Galerkin procedure, hence restricting its use to Hilbert space contexts. At the same time, these inner products are, themselves, constructed using the basis functions of the corresponding Fourier series solution representations, rather than from C_0^∞ test functions. As described briefly in Mitchell and Griffiths [20], it is possible to calculate the inner products using other than the basis set. In such a case the basis functions are termed *trial functions*, and the functions inserted into the second slot of the inner product are called *test functions* as usual. But, in any case, the Fourier coefficients that are the solution of the Galerkin equations are, by construction, solutions to a weak form—that is, an inner product (or linear functional) form of the original differential equation(s).

It is also important to recognize that typical Fourier series basis functions do not have compact support, although they are usually in C^∞. But details of construction of the Galerkin procedure render compact support less important than is true in the definition of weak solution—where, as observed earlier, it is crucial. Indeed, we should recall that in the Galerkin procedure, shifting of differentiation to the test functions is not necessary provided the basis functions are sufficiently smooth because differentiation is applied directly to the Fourier solution representation. In the foregoing we have not been careful to emphasize that proof of the ability to commute the two limit processes, differentiation and series summation, is not automatic. On the other hand, if the basis functions are sufficiently smooth this can always be carried out formally. But the question of convergence and, in particular, in what sense, must then be raised with regard to the differentiated series. There are a number of techniques needed to rigorously address such questions, and it is not our intent to introduce these in the present lectures. Our goal is simply to make the reader aware that validity of the Galerkin procedure requires proof—it is not automatic, in general.

Strong Solutions

In this subsection we will briefly introduce the notion of strong solution to a differential equation. It is somewhat difficult to find explicit definitions of this concept, and indeed, sometimes strong and classical solutions are equated. This is patently incorrect. The approach we follow in the present discussions is similar to that used in Foias *et al.* [15]; but we will provide a specific definition, in contrast to the treatment of [15].

There are several related ways to approach such a definition, and it is worthwhile to consider these to acquire a more complete understanding than can be obtained from a single viewpoint. We begin by recalling from our introductory discussions of weak solutions that a solution to a PDE might be weak only locally in space at any given time and, except on very small subsets of the problem domain Ω, might be classical. A somewhat similar notion can be associated with the fact, vaguely alluded to earlier, that distributional solutions to differential equations can be considered as comprising three specific types, described heuristically as:

i) classical solutions—these trivially satisfy the weak form of the differential equation;

ii) solutions that are differentiable, but not sufficiently so to permit formal substitution into the differential equation;

iii) solutions that are not differentiable at all, and may possibly not even be continuous.

Further details may be found in Stakgold [17].

To arrive at what is a widely-accepted definition of strong solution we will, in a sense, combine some of these ideas and cast the result in the notation introduced earlier in defining Sobolev spaces. We consider the following abstract initial boundary value problem. Let

$$\mathcal{P}u = f \qquad \text{in} \quad \Omega\times(t_0,t_f], \quad \Omega \subseteq \mathbb{R}^N\,, \tag{1.33a}$$

with initial conditions

$$u(\boldsymbol{x},t_0) = u_0(\boldsymbol{x})\,, \qquad \boldsymbol{x}\in\Omega\,, \tag{1.33b}$$

and boundary conditions

$$\mathcal{B}u(\boldsymbol{x},t) = g(\boldsymbol{x},t)\,, \qquad \boldsymbol{x}\in\partial\Omega\,, \quad t\in(t_0,t_f]\,. \tag{1.33c}$$

We assume $\mathcal{P}$ in (1.33a) is first order in time and of m^{th} order in space; $\mathcal{B}$ is a boundary operator consistent with well posedness for the overall problem (1.33). The operator $\mathcal{P}$ may be linear, or nonlinear, and we assume $f\in L^2(\Omega)$. As we have discussed earlier, this latter assumption has the consequence that $u\in H^m(\Omega)$, and hence that u can be represented by a Fourier series on the interior of Ω. If we require only that $u_0, g\in L^2(\Omega)$, then it is clear that u might not be in $H^m(\Omega)$ at early times and/or up to $\partial\Omega$ for all time. (We remark that there are theorems known as "trace" theorems dealing with such ideas; the reader is referred to, *e.g.*, Treves [21].) At the time we discuss behavior of N.–S. solutions we will provide some further details of some of these concepts, but here we consider only a case associated with strong solutions. Thus, we will suppose that both u_0 and g are sufficiently smooth that $u\in H^m(\Omega)$ holds. We then have the following definition.

Definition 1.6 *A* strong solution *to Prob.* (1.33) *is any function u that satisfies*

$$\|\mathcal{P}u - f\|_{L^2(\Omega)} = 0\,, \qquad \textit{and} \qquad \|\mathcal{B}u - g\|_{L^2(\partial\Omega)} = 0\,, \quad \forall\ t\in(t_0,t_f]\,, \tag{1.34}$$

and is in C^1 with respect to time. That is,

$$u(t)\in C^1(t_0,t_f;H^m(\Omega))\,. \tag{1.35}$$

From the theory of Lebesgue integration it follows that Eqs. (1.34) imply satisfaction of Eqs. (1.33) a.e. in Ω, and in particular, u is sufficiently smooth that its m^{th} derivatives exist except on sets of zero measure. This implies that over "most" of Ω, (spatial) differentiation can be performed in the usual sense through m^{th} order, but there could be some locations (that might move as the system evolves in time) at which this cannot be done. We also comment that our requirement of $u\in C^1$ with respect to time is slightly stronger than needed, and we will at times relax this in the sequel. Nevertheless, it is worthwhile to note that this requirement (invoked somewhat implicitly) and the norms of Eqs. (1.34) are widely used in the context of numerical solution of the N.–S. equations, as will be seen later. A final remark is that (1.34) can be written for any L^p space, but only L^2 will be needed herein.

1.2.3 Finite-difference/finite-volume approximations of non-classical solutions

Because essentially all of what we will later do in these lectures is finite-difference/finite-volume based, and we have already demonstrated the likelihood that N.–S. equations may exhibit non-classical solutions, it is important to consider how (or whether?) it is mathematically possible to produce such a solution with these approaches.

We first observe that any finite-difference (or finite-volume) solution is an approximation, supported on a finite number of points, of a true solution (if it exists) for a given problem—supported on an uncountable infinity of points. Moreover, for the low-order numerical methods we will utilize herein, it is clear that such numerical solutions are only piecewise C^k, with k fairly small when viewed as ordinary functions. For example, solutions obtained from second-order methods are piecewise linear with two different slopes meeting at each grid point (in 1D). This implies that at each such point the second derivatives are Dirac δ-functions, and thus are distributions. Hence, the numerical solution, itself, is clearly weak when viewed as a function on the continuum of the problem domain. The question that now arises is, “What happens as we refine the computational grid?” Clearly, we need to know the answer to this question in both the case of classical solutions and that of non-classical ones. In particular, are we able to obtain classical solutions when they exist, and is it possible to compute approximations to weak (or strong) solutions via finite-difference methods when classical solutions do not exist?

The former case is the subject of standard numerical analysis, and we shall merely recall a few simple facts here. We know for a grid function, $\{u_i\}_{i=1}^N$, computed on a grid of N points as an approximation to a function $u(x)$ that the relationship between u_i and $u(x_i)$ at any of the N points is given by

$$u_i = u(x_i) + \sum_{n=k}^{\infty} C_{n,i} h^n \,, \tag{1.36}$$

where the $C_{n,i}$ are constants (independent of h) that depend on derivatives of the <u>true</u> solution $u(x)$ at the point x_i, and h is the (possibly local) discrete step size. It is well known from basic analysis that if $u \in C^\infty$ in a neighborhood of x_i, then the above series converges, and the corresponding method is said to be of order k. In particular, each of the $C_{n,i}$s is bounded independent of h, so as $h \to 0$, the difference between the true and numerical solutions approaches zero. Hence, the numerical approximation converges to the true C^∞ solution, despite the fact that any individual solution with N finite is formally a distribution.

It is important to recognize, however, that if $u \in C^k(\Omega)$ with $k < \infty$, terms in the Taylor expansion of (1.36) are not necessarily bounded for $n > k+1$. Thus, even if the solution is classical, it is still possible for a numerical method to perform poorly in attempting to approximate it. For example, suppose the differential operators of the problem are of order no higher than two (our usual case), and suppose the exact solution to the problem can be shown to be in C^2, but not in C^3. Clearly, by our earlier definition, this is a classical solution, but even a low-order approximation will possess a truncation error whose dominant term will typically contain derivatives of at least fourth order—which do <u>not</u> exist. The situation becomes even more severe as we attempt use of higher-order methods with correspondingly higher-order derivatives as their leading truncation error terms, and this raises a serious question as to the value of such methods for anything but problems possessing C^k solutions with k quite large. We remark that C^∞ solutions are essentially always employed to test high-order methods, and in this context they are always shown to perform better than do low-order techniques—as should be expected. But this is not the correct approach for testing such methods. Instead, exact solutions employed should only be in C^k, with k less than the order of the method.

The foregoing provides a hint at the direction that must be taken when non-classical solutions are to be approximated using finite-difference or finite-volume methods. In both weak and strong cases, it is possible that <u>no</u> derivatives in the typical truncation error expansion exist in the usual sense, at least at some points in the solution domain Ω. In particular, in the case of weak solutions there may be no derivatives at all, while in the strong case derivatives may exist only in the sense of a global (over Ω) norm, and there may be at least a countable number of points in Ω where they do not exist. In order to understand the numerical treatment of such problems it is worthwhile to briefly look at what can be done analytically in such cases. We will need several specific results to carry this out.

The first of these is that any L^p function can be approximated arbitrarily closely in the L^p norm by step functions. This is a well-known result, the details of which can be found in any text on graduate-level analysis, *e.g.*, Royden [22]. Recall that step functions are piecewise constant, so they are precisely what would be produced by a first-order finite-difference method. We can analytically construct such functions very easily by first defining the *characteristic function* of a set A. Herein we will assume A is any connected subset of a solution domain $\Omega \subset \mathbb{R}^d$, $d = 1, 2, 3$, but we note that A could be a much more complicated set, and its characteristic function would still be well defined. In terms of this we have the following:

Definition 1.7 *For a subset A of a domain Ω, the <u>characteristic function</u> of A, denoted χ_A, is defined as*

$$\chi_A = \begin{cases} 1 & \forall\ x \in A\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{1.37}$$

Thus, if we let $\Omega = [a, b] \subset \mathbb{R}^1$, then for any ($L^p$) function f, the characteristic function can be used to define the step function $\chi_A f$ by dividing A into subintervals $A_i = [a_i, b_i]$, $i = 1, \ldots, N$, with $a_1 = a$, $b_N = b$ and $a_i = b_{i-1}$ $\forall$ $i > 1$ so that each A_i has length $h = b_i - a_i$, and evaluating f at the midpoint of each subinterval. We display these features in Fig. 1.2.

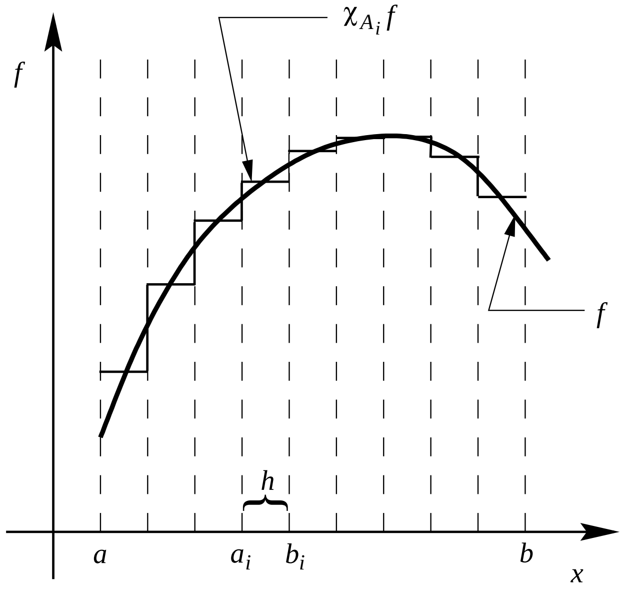

Figure 1.2: Step function constructed by applying the characteristic function to a function f.

Clearly, $\chi_A f$ is locally a scaled Heaviside function, and is thus a distribution with a (Dirac) δ-function as the distributional first derivative. Thus, although this type of approximation has advantages for analysis in general, it seems somewhat difficult to apply it in a straightforward way to approximate solutions to differential equations because of this inherent lack of differentiability—but this is exactly what we are doing in using (first-order) finite-difference methods. So we need to consider this further to justify use of such methods.

It should first be recalled that solutions to differential equations might be no smoother than the functions described above, so it is possible that such a general approximation may be needed in any case. Also, recall that in the context of distributions the remedy for nondifferentiability was introduction of C_0^∞ test functions and integration by parts, but we did not explain how such functions might be constructed in practice. Here we will provide one possible approach—but not the only one—and apply it in the context of solving PDEs. We will see that the details will be different than those presented earlier, and that the method now being considered has the advantage of producing functions that actually admit pointwise evaluation—which, as emphasized earlier, is not necessarily true of distributional solutions (defined in terms of linear functionals).

We begin by presenting a particular C_0^∞ function defined as

$$\delta_\epsilon(x) \equiv \begin{cases} c_\epsilon e^{-1/(\epsilon^2 - x^2)} & |x| < \epsilon\,, \\ 0 & |x| \geq \epsilon\,, \end{cases} \tag{1.38}$$

with normalization

$$c_\epsilon = \left[\int_{-\epsilon}^{\epsilon} e^{-1/(\epsilon^2 - x^2)}\, dx\right]^{-1}.$$

Now suppose we have a step function, say $\chi_A f$, approximating an arbitrary function $f \in L^2(a, b)$. It is shown in Gustafson [16] that

$$f_\epsilon(x) \equiv \int_a^b \chi_A f(y) \delta_\epsilon(x - y)\, dy \tag{1.39}$$

is in $C^\infty(a,b)$ provided ϵ is sufficiently small. Moreover, as already noted, this function can be evaluated pointwise a.e. in $[a,b]$. This process of converting a non-smooth function into a C^∞ one is known as *mollification*, and the function given in Eq. (1.38) is called a *mollifier* or, more precisely, the *kernel* of the mollifier given by (1.39). Figure 1.3 provides a plot of this function for several values of ϵ. Compact support of (1.38) is obvious, and the figure is suggestive of infinite smoothness. We leave formal proof of these properties to the reader.

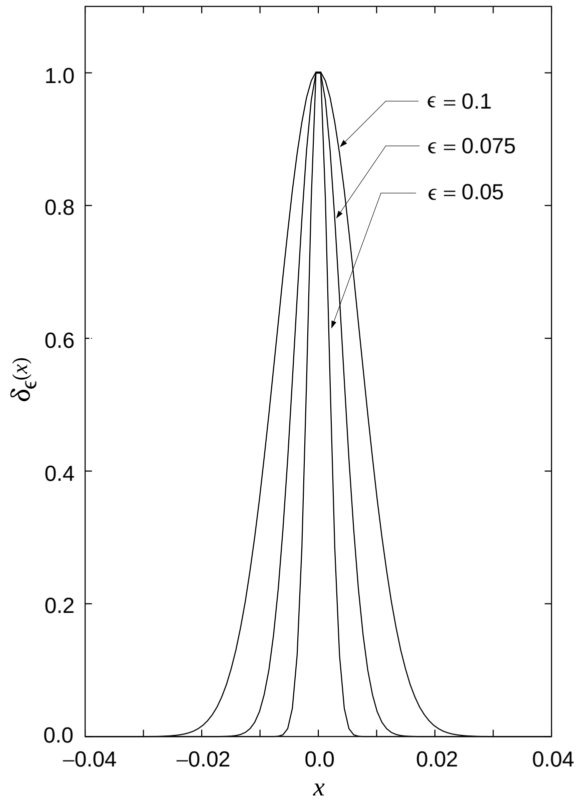

Figure 1.3: C_0^∞ δ_ϵ mollification functions for three values of ϵ.

Use of mollification as indicated by Eq. (1.39) is somewhat difficult in the context of any but linear PDEs, and even for these if coefficients are not constant it is necessary to estimate the error caused by commuting differentiation and mollification. In particular, suppose we consider the equation

$$\mathcal{P}u = f \qquad \text{on } \Omega$$

for which it is somehow known (for example, by analysis of boundary conditions and/or the specific nature of f) that u is not smooth. To treat this we apply mollification:

$$\int_\Omega \mathcal{P}u(y)\delta_\epsilon(x-y)\,dy = \int_\Omega f(y)\delta_\epsilon(x-y)\,dy\,. \tag{1.40}$$

(The reader familiar with large-eddy simulation of turbulence will recognize formal similarity of (1.40) to the basic construction of such methods.) Now recall that in the formal treatment of distributions δ_ϵ might have been a test function from the space $\mathcal{D}$, and in that case we would have integrated by parts to shift derivatives appearing in $\mathcal{P}$ onto δ_ϵ or, more precisely, for the variable-coefficient case we would carry this out for the adjoint operator $\mathcal{P}^*$.

But now the strategy is different—in a sense, more like that of the Galerkin procedure. If we could move $\mathcal{P}$ outside the integration we would be left with simply

$$\mathcal{P}u_\epsilon = f_\epsilon\,, \tag{1.41}$$

as is clear from (1.40). But since the limits of integration are constants for each fixed ϵ, this is possible whenever $\mathcal{P}$ is a constant-coefficient (and, in particular, linear) operator. Otherwise, there is an inherent commutation error. Up to estimating the damage done by such errors we see that in principle we can analytically treat the mollified equation for a nondifferentiable solution as if it were a C^∞ function a.e. From a practical, computational standpoint this corresponds to mollification od f, and probably also of u.

In the case of time-dependent problems such as will often be treated in the sequel, there is an alternative approach for utilizing mollification. Recall that in such cases we can, in principle, construct a solution operator such as Eq. (1.18), an example of which is provided by Eq. (1.17). (We can always construct discrete versions of solution operators numerically.) We have not previously discussed properties of solution operators, but a particularly important one for our purposes here is

$$S(t+s)u_0 = S(t)S(s)u_0 = u(t)\,. \tag{1.42}$$

This is one of several properties required for a family of operators to constitute a continuous *semigroup* (of operators). The interested reader is referred to texts on functional analysis such as Aubin [23] and Goldstein [24].

The notation of Eq. (1.42) implies that we can solve initial value problems over extended time intervals by subdividing these into smaller subintervals and solving a new initial value problem on each, using the final value attained on the previous interval as the initial data for the next succeeding one. It is easily seen that essentially all techniques for numerical integration of differential equations make use of this property, but we emphasize that it is not a trivial notion, as is clear from the cited references. We can formally express the above as

$$S(t)u_0 = S(t+s)u_0 = S(t)S(s)u_0 = S(t)u(s) = u(t)\,,$$

where $0 < s < t$. It is easily seen that typical discrete solution operators satisfy this property (*e.g.*, consider the forward-Euler approximation to the heat equation), but it is not trivial to demonstrate this in the continuous case.

It is clear from this (and more particularly, from extensions to larger numbers of subintervals) that we can actually apply mollification to the solution, itself, rather than to the PDE as done above, and this in turn permits us to treat even nonlinear problems in a natural way. The overall process then begins with mollification of the initial data (if necessary):

$$u_{0,\epsilon}(x) = \int_{-\epsilon}^{\epsilon} u_0(y)\delta_\epsilon(x-y)\,dy\,.$$

Then

$$S(s)u_{0,\epsilon} \simeq u(s)\,,$$

with approximate equality because $u_{0,\epsilon}(x)$ does not exactly equal $u_0(x)$ at all points of Ω.

Now, even though $u_{0,\epsilon} \in C^\infty(\Omega)$, it can easily be the case that $u(s) = S(s)u_{0,\epsilon} \notin C^\infty(\Omega)$. Thus, to compute $u(t)$ we should first apply mollification to $u(s)$ to obtain

$$u_\epsilon(x,s) = \int_{-\epsilon}^{\epsilon} u(y,s)\delta_\epsilon(x-y)\,dy\,,$$

where we have re-inserted explicit notation for spatial dependence of u for clarity. Then we obtain

$$S(t)u_\epsilon(s) \simeq u(t)\,.$$

We remark that it is often the case, especially for nonlinear problems such as those associated with the N.–S. equations, that existence of solutions can be proven only over finite (and sometimes rather short) time intervals. The preceding suggests a way by means of which time of existence might be extended. But it is important to recognize that this is not accomplished without a price. In particular, error must accumulate as the result of repeated mollification, so it is necessary to quantify and control such errors if the result at the final time is to be an accurate representation of the solution to the original problem. (In an analytical problem this error can be formally removed in the limit $\epsilon \to 0$, but in a computational setting this is not a useful limit because it recovers the non-smooth solution that was causing problems in the first place.)

We can now relate the preceding to the ability to approximate non-classical solutions using finite-difference and finite-volume procedures. The first point to note is that, as discussed earlier, discrete solutions by their basic nature are nonclassical, and we have already indicated how these are able to converge to true classical solutions—at least in cases where these are sufficiently smooth. At this point we need to reconsider what happens to a discrete solution approximating a non-classical one as discretization step sizes are decreased.

Suppose the true solution is sufficiently badly behaved that the formal dominant truncation error (obtained via Taylor expansion) is already a δ-function. Then as grid spacing is decreased by increasing the number of grid points, the number of points at which δ-function like errors occur will also increase, making convergence in a (uniform) pointwise sense impossible (as it must be for weak solutions, and often will be for strong ones at some points). Moreover, in the case of nonlinear problems such errors can easily be amplified, leading to numerical instability. So it might seem that attempts to accurately approximate non-classical solutions using difference approximations are doomed to failure. But just as was the case in the analytical treatment described above, the remedy is mollification. This has been used in many different forms beginning as early as the 1940s, and in a sense all are equivalent. Here, we emphasize a version of this that closely mimics the analytical procedure presented above.

For readers familiar with signal processing, or with any situation in which filters are employed (for example, large-eddy simulation, LES, of turbulent fluid flow mentioned earlier), it is readily apparent that Eq. (1.39) represents application of a filter with kernel δ_ϵ to the non-smooth function $\chi_A f$. This implies that if we can find a digital filter to apply to the numerical solution obtained after each step of a discrete solution operator, then we can produce a smooth solution that can, in principle, be made arbitrarily close to the exact non-classical solution. Figure 1.3 provides a hint as to the nature of a filter applicable to numerical mollification. In particular, it is apparent from the figure, that independent of the value of $\epsilon > 0$, the graph of δ_ϵ is fairly close to being triangular (but smooth); *i.e.*, it consists, to a fairly good approximation, of two straight line segments. In turn, this implies that the product of δ_ϵ and any step function can be integrated exactly via trapezoidal integration provided ϵ is sufficiently small that δ_ϵ spans only three function values of the (numerical) step function.

It turns out, as we will show in some detail in Chap. 2, that this heuristic argument can be made more precise to derive a specific form of filter introduced by Shuman [25] for the analysis of atmospheric data. For now we simply note that such procedures, as well as other similar approaches, are available for converting non-smooth numerical solutions to smooth ones. We will later provide details of some of these, and furthermore explain exactly how they accomplish the desired end. But we will do this more specifically in the context of the Navier–Stokes equations. In any case, at this point we expect that we can actually produce numerical approximations to non-classical solutions of PDEs in general, so it is now time to begin study of the nature of solutions to the N.–S. equations, which as we will see, are often nonclassical.

1.3 Existence, Uniqueness and Regularity of N.–S. Solutions

In this section we begin by introducing three physically-based ideas somewhat specific to the incompressible Navier–Stokes equations and use these to define function spaces needed to precisely state theorems associated with existence, uniqueness and regularity of N.–S. solutions. In particular, we will define spaces of functions based on finite energy and enstrophy, along with satisfaction of the divergence-free constraint. We will also present some specific mathematical techniques applied to the N.–S. equations that will be valuable both in discussing the theorems alluded to above and in understanding the foundations of some of the solution algorithms to be analyzed in the sequel. These will include Helmholtz–Leray projection, construction of weak and Galerkin forms of the N.–S. equations and energy (in)equalities. We conclude the section by stating and discussing some of the main well-known results from the mathematical theory of the Navier–Stokes equations.

1.3.1 Function spaces incorporating physics of incompressible flow

We begin this treatment by reminding the reader of the relationship between functions in L^2 and finite kinetic energy, and we describe a corresponding notion associated with the Sobolev space H^1 and a physical quantity known as enstrophy.

Finite Energy and the Function Space $H(\Omega)$

Recall from Eq. (1.8) that a function f is in $L^2(\Omega)$ if (and only if)

$$\int_\Omega |f|^2\,dx < \infty$$

where $|\cdot|$ denotes the modulus of f in the sense of complex numbers (and thus can be dropped for real-valued scalar functions). We can readily extend this to vector-valued real functions, say $\boldsymbol{u} = (u(\boldsymbol{x}), v(\boldsymbol{x}), w(\boldsymbol{x}))^T$ with $\boldsymbol{x} = (x, y, z)$, for example, by replacing the complex modulus with Euclidean length (and retaining the same notation). Thus, for $\Omega \subseteq \mathbb{R}^3$ we write

$$\int_\Omega |\boldsymbol{u}|^2\,d\boldsymbol{x} \equiv \int_\Omega u^2 + v^2 + w^2\,dxdydz = \int_\Omega \boldsymbol{u}\cdot\boldsymbol{u}\,dxdydz\,, \tag{1.43}$$

which up to a factor of $\frac{1}{2}\rho$, with ρ being fluid density, is the kinetic energy per unit volume. For incompressible flow, $\rho \equiv$ const., and in mathematics it is common to set this constant to unity. Hence, we define the (mathematical) energy of the flow field in a domain Ω as

$$e(\boldsymbol{u}) \equiv \frac{1}{2}\int_\Omega |\boldsymbol{u}|^2\,d\boldsymbol{x}\,. \tag{1.44}$$

Clearly, since $e = \frac{1}{2}\|\boldsymbol{u}\|^2_{L^2}$, if $\boldsymbol{u} \in L^2(\Omega)$ the flow has finite energy.

Now recall that the incompressible N.–S. equations comprise a system of initial boundary value problems (whose solutions must be divergence free—recall Eqs. (1.1) and so must have initial and boundary data prescribed to constitute a well-posed problem. From the perspective of mathematics, it is convenient to include at least the boundary data and the divergence-free requirement in definitions of function spaces in which solutions might be sought. We thus define the space of functions denoted H on a domain Ω as

$$H(\Omega) \equiv \left\{\boldsymbol{f} \in L^2(\Omega);\ \nabla\cdot\boldsymbol{f} = 0\,,\ \boldsymbol{B}_{\partial\Omega}\boldsymbol{f} \text{ prescribed}\right\}\,, \tag{1.45}$$

where $\boldsymbol{B}_{\partial\Omega}$ denotes the boundary operators on $\partial\Omega$ appearing in the boundary conditions of the given N.–S. problem. In words, we say a function $\boldsymbol{f}$ is in the space $H(\Omega)$ if it is divergence free, satisfies the prescribed boundary conditions of the problem and has finite energy in Ω in the sense of Eq. (1.44). This function space plays a key role in the theory of N.–S. solutions, as will become apparent. We also remark that for vector-valued functions (herein denoted by bold type) containing d components, as we are now treating, the corresponding function spaces are sometimes denoted as, *e.g.*, $L^2(\Omega)^d$ or $H(\Omega)^d$. We will not, however, follow this practice in these lectures as it is superfluous if the number of components of $\boldsymbol{f}$ is known. We further note that $H(\Omega)$ is <u>not</u> the same as the Sobolev space $H^1(\Omega)$.

Finite Enstrophy and the Function Space $V(\Omega)$

The term "enstrophy" seldom arises in elementary fluid dynamics—and probably almost never in mathematics; but it occurs frequently in studies of turbulence because, at least in classical analyses, much emphasis is placed on the role of vorticity in turbulence generation, and as shown below, enstrophy is essentially the L^2 norm of vorticity.

We first recall from elementary fluid dynamics that *vorticity* is defined as the curl of the velocity vector:

$$\boldsymbol{\omega} \equiv \nabla \times \boldsymbol{u} .$$

This is, of course, a vector if $\boldsymbol{u} \in \mathbb{R}^3$ (but only a scalar for $\boldsymbol{u} \in \mathbb{R}^2$); so we can write $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)^T$ and calculate a quantity analogous to energy (but without the factor of 1/2):

$$E(\boldsymbol{u}) \equiv \int_\Omega |\boldsymbol{\omega}|^2 \, d\boldsymbol{x} . \tag{1.46}$$

This equation defines <u>physical</u> *enstrophy*; clearly, enstrophy is simply the square of the L^2 norm of the vector vorticity.

The mathematical definition is slightly different, but equivalent in a sense we will indicate below. Recall that the components of vorticity are computed as differences of velocity derivatives (*i.e.*, fluid parcel angular rotation rates); for example,

$$\omega_3 = \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right) .$$

In light of (1.46) this suggests that enstrophy is associated with (first) derivatives of the velocity field being in $L^2(\Omega)$, and hence with the Sobolev space $H^1(\Omega)$. In turn, this implies we instead might consider the L^2 norm of the velocity gradient matrix (tensor):

$$\int_\Omega |\nabla \boldsymbol{u}|^2 \, dx ,$$

where now $|\cdot|$ denotes the finite-dimensional matrix 2-norm.

Now observe that if we add and subtract the transpose of $\nabla \boldsymbol{u}$, we obtain

$$\begin{aligned}
\nabla \boldsymbol{u} &= \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{pmatrix} \\
&= \frac{1}{2} \left[\begin{pmatrix} 2u_x & u_y + v_x & u_z + w_x \\ v_x + u_y & 2v_y & v_z + w_y \\ w_x + u_z & w_y + v_y & 2w_z \end{pmatrix} + \begin{pmatrix} 0 & u_y - v_x & u_z - w_x \\ v_x - u_y & 0 & v_z - w_y \\ w_x - u_z & w_y - v_z & 0 \end{pmatrix} \right] .
\end{aligned}$$

The first matrix on the right-hand side is the *strain rate* (tensor), while the second contains the components of the vorticity and is sometimes referred to as the *rotation tensor.* We leave as an exercise to the reader demonstration that

$$\frac{\sqrt{2}}{2} |\boldsymbol{\omega}| \le |\nabla \boldsymbol{u}| ,$$

and hence that using

$$E(\boldsymbol{u}) = \int_\Omega |\nabla \boldsymbol{u}|^2 \, d\boldsymbol{x} \tag{1.47}$$

is equivalent to using the physical definition of enstrophy in the sense that if $E(\boldsymbol{u})$ is finite by Eq. (1.47) then it will also be when calculated in terms of Eq. (1.46), but in general could be either somewhat smaller or larger. Equation (1.47) is the <u>mathematical</u> definition of *enstrophy.*

We can write this as

$$E(\boldsymbol{u}) = \sum_{i,j=1}^{d} \int_\Omega \left(\frac{\partial u_i}{\partial x_j}(\boldsymbol{x}) \right)^2 d\boldsymbol{x} \,,$$

where $d = 2, 3$, and we assume all partial derivatives take on only real values. This leads us to a definition of the function space usually denoted as V:

$$V(\Omega) \equiv \left\{ \boldsymbol{f} \in L^2(\Omega);\ \nabla\cdot \boldsymbol{f} = 0\,,\ B_{\partial\Omega}\boldsymbol{f} \text{ prescribed}\,,\ \nabla \boldsymbol{f} \in L^2(\Omega) \right\}\,. \tag{1.48}$$

In words, the function space $V(\Omega)$ consists of those vector-valued functions on Ω that are divergence free, satisfy boundary conditions for the given N.–S. problem and have finite enstrophy. It is clear that one could replace the finite enstrophy requirement by $\boldsymbol{f} \in H^1(\Omega)$. If we recall Eq. (1.11) and extend this in a natural way to the vector-valued functions we are now considering, we see that the H^1 norm of $\boldsymbol{u}$ will be greater than or equal to the square root of the enstrophy.

1.3.2 The Helmholtz–Leray decomposition and Leray projection

In this subsection we will present two pieces of mathematics that are crucial to modern analysis of the N.–S. equations. These are the Helmholtz–Leray decomposition of vector fields and the related Leray projector. It will be evident in Chap. 3 that the latter of these is widely used in modern numerical algorithms employed to solve the incompressible N.–S. equations.

Helmholtz–Leray Decomposition

The Helmholtz decomposition is fairly well known and widely used in specific situations in CFD, for example in simulation of geophysical flows by spectral methods. It is a special case of a more general mathematical result known as the *Hodge decomposition* [26] which can be loosely paraphrased as: any L^2 vector field can be decomposed as the sum of curl-free and divergence-free parts. The Helmholtz version of this actually applies to functions generally smoother than those simply in L^2 (but we could, in principle, use weak forms of the derivatives to circumvent this technicality). It should be further noted that such decompositions are not unique, and they do not explicitly account for boundary conditions associated with specific problems whose solutions might be the vector fields in question.

Leray provided a modification of the above description that addresses both of these two difficulties, and we present this here. We remark that this material is taken nearly *verbatim* from the monograph by Foias *et al.* [15].

In constructing a *Helmholtz–Leray decomposition* we consider a specified vector field $\boldsymbol{v}$ on a bounded domain $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, taking on specified conditions on $\partial\Omega$. We seek a representation of $\boldsymbol{v}$ in the form

$$\boldsymbol{v} = \nabla\phi + \boldsymbol{u}\,, \qquad \text{with} \quad \nabla\cdot\boldsymbol{u} = 0\,. \tag{1.49}$$

We will consider any required differentiation in the classical sense, but note that formal (as opposed to rigorous) extension to weak derivatives, if necessary, is straightforward, *e.g.*, via mollification. Now observe that in this context $\nabla\phi$ is clearly curl free (proof of this, using only elementary vector calculus, is left as an exercise for the reader), and $\boldsymbol{u}$ is divergence free by stipulation (but we will see below that this follows from the construction used to produce it). The question is whether for a given arbitrary vector field $\boldsymbol{v}$ on Ω we can actually find a scalar ϕ and a divergence-free vector field $\boldsymbol{u}$ such that the first of Eqs. (1.49) is satisfied.

Application of the divergence operator to (1.49) leaves

$$\Delta\phi = \nabla\cdot\boldsymbol{v}\,, \tag{1.50}$$

which is just a Poisson equation for ϕ and can be directly solved if boundary conditions for ϕ in terms of $\boldsymbol{v}$ can be obtained. If $\boldsymbol{v}$ satisfies no-slip and no-flux conditions, it follows that if we only require $\boldsymbol{u}\cdot\boldsymbol{n} = 0$

on $\partial\Omega$, we can derive an appropriate boundary condition. In particular, if we take the dot product of the first equation in (1.49) with the outward unit normal vector $\boldsymbol{n}$ to $\partial\Omega$, and use the condition just given for $\boldsymbol{u}$, we see that

$$\nabla\phi\cdot\boldsymbol{n} = \boldsymbol{v}\cdot\boldsymbol{n}\,,$$

or, equivalently, in more suggestive notation,

$$\frac{\partial\phi}{\partial\boldsymbol{n}} = \boldsymbol{v}\cdot\boldsymbol{n} \qquad \text{on} \quad \partial\Omega\,. \tag{1.51}$$

Together, Eqs. (1.50) and (1.51) constitute a Neumann problem for the Poisson equation for ϕ. To guarantee existence of a solution to such problems it is required that the right-hand side of the boundary condition satisfy a *consistency* or *compatibility* condition of the form (see Stakgold, Vol. II [17] or Garabedian [27])

$$\int_{\partial\Omega} \boldsymbol{v}\cdot\boldsymbol{n}\,dA = 0\,,$$

which holds automatically when no-slip/no-flux conditions are imposed on $\boldsymbol{v}\big|_{\partial\Omega}$. Thus in this case (corresponding, physically, to solid impenetrable boundaries), we are guaranteed that ϕ is well defined and unique (only) to within an additive constant. Then $\boldsymbol{u}$ can be computed directly from Eq. (1.49) rearranged as

$$\boldsymbol{u} = \boldsymbol{v} - \nabla\phi\,. \tag{1.52}$$

Furthermore, applying the divergence operator to both sides of this result, and using Eq. (1.50), shows that $\boldsymbol{u}$ is indeed divergence free, as required.

We comment that we will encounter an identical construction associated with a particular class of N.–S. solution algorithms (the projection methods) in Chap. 3 of these lectures, and that the formalism just presented provides a rigorous justification for use of these methods. We should also note that similar results can be obtained for periodic boundary conditions for $\boldsymbol{u}$ (see [15]). We leave proof of uniqueness (up to an additive constant) as an exercise for the mathematically-inclined reader. It requires only the most basic standard techniques used to prove uniqueness for PDE solutions.

The Leray Projector

Assuming the preceding operations have been carried out, we see that the mapping $\boldsymbol{v} \longmapsto \boldsymbol{u}(\boldsymbol{v})$ on Ω is well defined, and we denote this as

$$P_L\colon \boldsymbol{v} \longmapsto \boldsymbol{u}(\boldsymbol{v})\,. \tag{1.53}$$

This is formal symbolism for the *Leray projector*. Indeed, P_L is a projector, for if $\boldsymbol{v}$ is already divergence free it will remain unchanged upon application of Eq. (1.52), as follows from the fact that Eq. (1.50) becomes $\Delta\phi = 0$ which implies $\phi \equiv$ const for the boundary conditions considered here. Hence, $P_L^2 = P_L$, as required.

1.3.3 Derivation of "mathematical" forms of the Navier–Stokes equations

There are three forms of the N.–S. equations that have been widely studied by mathematicians, and indeed, the formal theorems to be presented below have been proven within the context of one or more of these forms—and not in the form of Eqs. (1.1). This fact tends to lead engineers to conclude that "the mathematicians don't know what they are doing." But we will show here that the mathematical forms are equivalent to Eqs. (1.1), and in some cases are (almost) directly used in modern simulation procedures. There are three such forms that we will discuss in these lectures: *i*) an evolution equation (dynamical system) for the velocity field obtained by Leray projection, *ii*) a weak form of the N.–S. equations and *iii*) the Galerkin form of the N.–S. equations (which is also a dynamical system, but now expressing time evolution of the Fourier coefficients—hence, also formally a weak representation).

For convenience we begin by recalling Eqs. (1.1) and expressing them in a slightly different and more useful form:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \nu \Delta \boldsymbol{u} + \boldsymbol{F}_B \tag{1.54a}$$

$$\nabla \cdot \boldsymbol{u} = 0\,, \tag{1.54b}$$

where we have divided by the constant density to arrive at kinematic viscosity ν as the coefficient of the Laplace operator; we have also absorbed the density into the pressure (resulting in what is sometimes referred to as "kinematic" pressure) and the body-force terms without changing their notation from that of Eqs. (1.1).

Leray Projection of the N.–S. Equations

The reader will recall that earlier (when the solution operator was introduced) we noted it is sometimes convenient to view the N.–S. equations as a dynamical system. Here we will provide the details of deriving this particular form. As already hinted, this can be done by Leray projection; the treatment presented here follows very closely that found in [15].

From Eq. (1.54b) we see that $\boldsymbol{u}$ must be divergence free (without concern for how this might actually occur—but we know it can be made divergence free via Leray projection). So application of the Leray projector, given symbolically in (1.53) and "computationally" in (1.52), results in

$$P_L \boldsymbol{u} = \boldsymbol{u}\,, \qquad P_L \frac{\partial \boldsymbol{u}}{\partial t} = \frac{\partial \boldsymbol{u}}{\partial t}\,, \qquad \text{and} \quad P_L \nabla p = 0\,.$$

Since $\boldsymbol{u}$ is divergence free, validity of the first two of these is obvious. We leave verification of the last result to the reader as an exercise in applying Leray projection.

There are three terms remaining in (1.54a) yet requiring projection. The simplest is the body force, and since we have not provided a specific form for this explicitly in terms of physical dependent variables, we will simply represent it symbolically as $\boldsymbol{f} = P_L \boldsymbol{F}_B$. (We note, however, that this is often required to be divergence free in mathematical analysis of the N.–S. equations.)

The next term we consider is the Laplacian of the velocity field, for which the following notation is standard in mathematical treatments:

$$A\boldsymbol{u} = -P_L \Delta \boldsymbol{u}\,.$$

The operator $A \equiv -P_L\Delta$ is called the *Stokes operator*. It is worth noting that for periodic boundary conditions $A = -\Delta$ holds, but the case of no-slip conditions is more complicated. In any case, it can be shown that the eigenfunctions of the Stokes operator provide a complete orthonormal basis for constructing Fourier representations of N.–S. solutions in the sense described earlier for Sturm–Liouville problems in general (but only in one space dimension). We emphasize that this is not a trivial result due to the nonlinearity of the N.–S. equations. Leray projection of these nonlinear terms is typically represented using the notation

$$B(\boldsymbol{u}) = B(\boldsymbol{u}, \boldsymbol{u}) \equiv P_L(\boldsymbol{u} \cdot \nabla \boldsymbol{u})\,.$$

We now substitute these notations back into Eq. (1.54a) to obtain

$$\frac{d\boldsymbol{u}}{dt} + \nu A \boldsymbol{u} + B(\boldsymbol{u}) = \boldsymbol{f}\,, \tag{1.55}$$

where we have replaced $\partial/\partial t$ with d/dt to emphasize the view that spatial dependence is completely absorbed in the operators A, B and in $\boldsymbol{f}$, and that $\boldsymbol{u} = \boldsymbol{u}(t)$ represents temporal evolution of the vector field $\boldsymbol{u}(\boldsymbol{x}, t)$. This suggests the usual notation employed for dynamical systems:

$$\boldsymbol{u}' = \boldsymbol{F}(\boldsymbol{u}, t)\,,$$

or if $\boldsymbol{F}_B$ is independent of time,

$$\boldsymbol{u}' = \boldsymbol{F}(\boldsymbol{u}) ,$$

with

$$\boldsymbol{F}(\boldsymbol{u}) \equiv \boldsymbol{f} - \nu A\boldsymbol{u} - B(\boldsymbol{u}) .$$

We note that $\boldsymbol{F}(\boldsymbol{u})$ contains no explicit information from the pressure. We will see from the Galerkin form of the N.–S. equations, below, how effects of pressure can be incorporated without explicitly including pressure-gradient terms, and we will later see that not having to explicitly deal with these is a significant advantage in practical N.–S. solution algorithms.

Weak Form of the N.–S. Equations

We begin by recalling that, in general, the weak form of a PDE is obtained by integrating the PDE against C_0^∞ test functions φ over the domain Ω. That is, we form the scalar product of terms in the PDE with test functions, and then integrate by parts to move derivatives from the nondifferentiable PDE solution variable to the test function, as in Eqs. (1.28), (1.29) and (1.32). The approach for the N.–S. equations is the same, in principle, but there are differences in details that will be evident as we proceed.

The first difference occurs with the choice of test functions. In obtaining the weak form of the N.–S. equations we employ test functions from the space $V(\Omega)$. Thus, instead of being infinitely continuously differentiable as are the functions $\varphi \in \mathcal{D}$ used previously, the functions $\boldsymbol{v} \in V(\Omega)$ are only in $H^1(\Omega)$; *i.e.*, they have only a single derivative in L^2. But we will see that this is all that is needed. Also recall that functions $\boldsymbol{v} \in V$ must be divergence free, and they must satisfy the same boundary conditions imposed on the N.–S. solutions, themselves, for any specific problem under consideration. With these basic ideas in mind, we now consider each of the terms in Eq. (1.54a) individually. We will carry out details only for the case of solid-wall boundaries on which are imposed the usual no-slip/no-flux boundary conditions, but most of what we present can be employed in cases of other standard boundary conditions, at least after some modification.

As is usual, the simplest term is the body-force term, and we write this in weak form as

$$\langle \boldsymbol{F}_B, \boldsymbol{v} \rangle = \int_\Omega \boldsymbol{F}_B \cdot \boldsymbol{v}\, d\boldsymbol{x} ,$$

which clearly is in the form of the L^2 (vector) inner product. For this to make sense we must require that $\boldsymbol{F}_B \in L^2(\Omega)$ hold. The next simplest term in Eq. (1.54a) is the time derivative. We have

$$\int_\Omega \frac{\partial}{\partial t}\boldsymbol{u}(\boldsymbol{x},t) \cdot \boldsymbol{v}(\boldsymbol{x})\, d\boldsymbol{x} = \frac{d}{dt}\int_\Omega \boldsymbol{u}(\boldsymbol{x},t) \cdot \boldsymbol{v}(\boldsymbol{x})\, d\boldsymbol{x} ,$$

which emphasizes the fact that the test functions $\boldsymbol{v}$ depend only on the spatial coordinates, and not on time. We also require Ω, itself, be independent of time. Then correctness of the above equality is obvious, and we express this as

$$\frac{d}{dt}\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \int_\Omega \frac{\partial \boldsymbol{u}}{\partial t} \cdot \boldsymbol{v}\, d\boldsymbol{x} .$$

We next consider the pressure gradient term of Eq. (1.54a). This leads to

$$\begin{aligned}
\langle \nabla p, \boldsymbol{v} \rangle &= \int_\Omega \nabla p \cdot \boldsymbol{v}\, dV \\
&= \int_{\partial\Omega} p\boldsymbol{v} \cdot \boldsymbol{n}\, dA - \int_\Omega p\nabla \cdot \boldsymbol{v}\, dV \\
&= 0 .
\end{aligned}$$

To see that this result holds, we first note that $\boldsymbol{v} \in V$ implies $\nabla \cdot \boldsymbol{v} = 0$ by the definition of the function space V; hence, the second term is zero. The first term is clearly zero as well in the case of no-flux boundary conditions considered here. We note that other boundary conditions are somewhat more difficult to analyze, but for conditions typical for the N.–S. equations the above result holds generally.

Now we treat the viscous (Laplacian) term. We have, by a direct calculation (left to the interested reader),

$$\langle \Delta \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{i,j=1}^{d} \int_{\partial\Omega} \frac{\partial u_i}{\partial x_j} v_i \, dA - \sum_{i,j=1}^{d} \int_{\Omega} \frac{\partial u_i}{\partial x_j} \frac{\partial v_i}{\partial x_j} \, dV \,, \qquad d = 2, 3 \,.$$

(The notation being used here is sometimes referred to as "Cartesian tensor" notation, and is widely used in classical turbulence formalisms. It corresponds to $\boldsymbol{u} = (u_1, u_2, u_3)$ and $\boldsymbol{x} = (x_1, x_2, x_3)$.) For no-slip/no-flux boundary conditions $v_i = 0$ on $\partial\Omega$, so the first integral on the right-hand side vanishes leaving

$$\langle \Delta \boldsymbol{u}, \boldsymbol{v} \rangle = - \sum_{i,j=1}^{d} \int_{\Omega} \frac{\partial u_i}{\partial x_j} \frac{\partial v_i}{\partial x_j} \, dV \,.$$

Similarity of the right-hand side of this equation to the H^1 inner product, Eq. (1.11), should be noted. Indeed, in (1.11) we need only start the indexing of the first summation at $k = 1$ to obtain the form given above. There are specific notations for this result, although none appear to be very standard. Since by definition $\nabla \boldsymbol{v} \in L^2(\Omega)$ must hold for $\boldsymbol{v}$ to be in V, we use the notation

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle_{V(\Omega)} \equiv \sum_{i,j=1}^{d} \int_{\Omega} \frac{\partial u_i}{\partial x_j} \frac{\partial v_i}{\partial x_j} \, dV \,. \tag{1.56}$$

Finally, we note that little can be done with the nonlinear term beyond introducing some notation for it. The scalar product in this case takes the form

$$\langle \boldsymbol{u} \cdot \nabla \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{i,j=1}^{d} \int_{\Omega} u_i \frac{\partial u_j}{\partial x_i} v_j \, dV \,,$$

and we employ the fairly standard short-hand notation

$$b(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{v}) \equiv \langle \boldsymbol{u} \cdot \nabla \boldsymbol{u}, \boldsymbol{v} \rangle \,. \tag{1.57}$$

We now combine the various preceding results and express them in a form that emphasizes temporal dependence, since now the spatial dependence will be integrated out in formally calculating the inner products. Thus, the *weak form* of the N.–S. equations can be written as

$$\frac{d}{dt} \langle \boldsymbol{u}(t), \boldsymbol{v} \rangle + \nu \langle \boldsymbol{u}(t), \boldsymbol{v} \rangle_{V(\Omega)} + b(\boldsymbol{u}(t), \boldsymbol{u}(t), \boldsymbol{v}) = \langle \boldsymbol{F}_B, \boldsymbol{v} \rangle \,, \tag{1.58}$$

which holds $\forall\, \boldsymbol{v} \in V(\Omega)$. Once initial data $\boldsymbol{u}(0) = \boldsymbol{u}_0(\boldsymbol{x})$ have been specified, this equation can be integrated in time to find $\langle \boldsymbol{u}(t), \boldsymbol{v} \rangle$. It should be clear that such a result must correspond to $\boldsymbol{u} \in H^1$, but $\boldsymbol{u} \in H^2$ is required to have even a strong (but not a classical) solution. Furthermore, the "algorithm" by means of which this might actually be carried out is not obvious. In particular, note that there is no straightforward procedure for computing $\partial u_i / \partial x_j$ knowing only $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$. Hence, it is not clear how $\langle \boldsymbol{u}, \boldsymbol{v} \rangle_{V(\Omega)}$ and $b(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{v})$ might be evaluated. On the other hand, these forms readily lend themselves to mathematical analysis.

One of the more practically useful things (from a mathematical standpoint) that can be obtained from the weak form of the N.–S. equations is an energy equation that describes the time evolution of (kinetic) energy of a velocity field, and hence whether the L^2 norm of the solution remains bounded. Two things are required to derive this. First, we choose a very special test function, *viz.*, $\boldsymbol{u}$ itself. Then we need to show that $b(\boldsymbol{u},\boldsymbol{u},\boldsymbol{u}) = 0$ in order to obtain the desired result. To show this we observe that using Eq. (1.57) with $\boldsymbol{v} = \boldsymbol{u}$ and the equation preceding this results in

$$\begin{aligned} b(\boldsymbol{u},\boldsymbol{u},\boldsymbol{u}) &= \sum_{i,j=1}^{d} \int_\Omega u_j \frac{\partial u_i}{\partial x_j} u_i \, dV \\ &= \frac{1}{2} \sum_{i,j=1}^{d} \int_\Omega u_j \frac{\partial}{\partial x_j} \left(u_i^2\right) \, dV \\ &= -\frac{1}{2} \sum_{i,j=1}^{d} \int_\Omega \frac{\partial u_j}{\partial x_j} u_i^2 \, dV \\ &= 0\,, \end{aligned}$$

since

$$\nabla \cdot \boldsymbol{u} = \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} + \frac{\partial u_3}{\partial x_3} = 0\,,$$

and boundary terms arising from integration by parts in going between the second and third steps drop out due to the no-slip/no-flux conditions.

We also have

$$\begin{aligned} \int_\Omega \frac{\partial \boldsymbol{u}}{\partial t} \cdot \boldsymbol{u} \, dV &= \frac{1}{2} \int_\Omega \frac{\partial}{\partial t}(U^2) \, dV = \frac{1}{2} \frac{d}{dt} \int_\Omega U^2 \, dV \\ &= \frac{d}{dt} e(\boldsymbol{u})\,, \end{aligned}$$

where we have used the notation $U^2 \equiv \boldsymbol{u} \cdot \boldsymbol{u}$. We can now identify Eq. (1.56) with (1.47) to arrive at

$$\frac{d}{dt} e(\boldsymbol{u}) = -\nu E(\boldsymbol{u}) + \langle \boldsymbol{F}_B, \boldsymbol{u} \rangle\,, \tag{1.59}$$

which shows that the growth (or decay) of energy of the flow field depends on a balance between dissipation due to viscosity times enstrophy and forcing from body forces (or boundary conditions). We remark that although the boundary conditions considered here arise frequently in physical problems, there are other possibilities, and these must be treated somewhat differently.

The Galerkin Procedure Applied to the N.–S. Equations

We have already presented a fairly detailed, general treatment of the Galerkin procedure in our earlier discussions of weak solutions. As noted at that time, this provides a weak form admitting pointwise evaluation and is thus potentially useful for practical computations. Here we apply this specifically to the N.–S. equations which for simplicity we treat in 2-D dimensionless form in the absence of body forces. We can express this case as

$$u_t + (u^2)_x + (uv)_y = -p_x + \frac{1}{Re}\Delta u\,, \tag{1.60a}$$

$$v_t + (uv)_x + (v^2)_y = -p_y + \frac{1}{Re}\Delta v\,, \tag{1.60b}$$

$$u_x + v_y = 0\,. \tag{1.60c}$$

Here subscripts denote partial differentiation, and Δ is the Laplacian; Re is the Reynolds number: $Re = UL/\nu$ with U and L being appropriate velocity and length scales, respectively. We observe that Eqs.

(1.60a) and (1.60b) are in so-called "conservation form," which we will discuss in more detail in Chap. 2. Here, we simply note that it is a convenient form of the equations for application of the Galerkin procedure. Moreover, we note that considerations of 3-D domains and/or including forcing terms does not change what we shall present in any essential way from the standpoint of the computational process, *per se*. On the other hand, as will be clear in the next subsection, these impose a significant change in what is guaranteed regarding nature of solutions.

In Examples 1.2 and 1.3 we earlier expressed the dependent variables in Fourier series; we follow the same approach here:

$$u(x,y,t) = \sum_{\boldsymbol{k}}^{\infty} a_{\boldsymbol{k}}(t)\varphi_{\boldsymbol{k}}(x,y)\,, \tag{1.61a}$$

$$v(x,y,t) = \sum_{\boldsymbol{k}}^{\infty} b_{\boldsymbol{k}}(t)\varphi_{\boldsymbol{k}}(x,y)\,, \tag{1.61b}$$

$$p(x,y,t) = \sum_{\boldsymbol{k}}^{\infty} c_{\boldsymbol{k}}(t)\varphi_{\boldsymbol{k}}(x,y)\,. \tag{1.61c}$$

As was true in earlier discussions of the Galerkin procedure, $\boldsymbol{k} \equiv (k_1, k_2)$, and the lower bound for components of this wavevector is typically one of $-\infty$, 0 or 1. Also, an analogous expansion would be given for the body force if one were present.

For convenience we will assume Ω is a rectangle and that the boundary conditions used with Eqs. (1.60) are periodicity conditions. Although this is a quite restrictive situation, our main goal here is to introduce some mathematical notions associated with the N.–S. equations when viewed in Fourier space, so the overall simplicity of this arrangement is an advantage. It is important to note, however, that it is difficult to lift these conditions and retain a numerically efficient procedure; as a consequence, most computer implementations of this method are quite similar to the version we will discuss here.

In light of the periodicity conditions we can employ complex exponentials as basis functions; that is, we set

$$\varphi_{\boldsymbol{k}}(x,y) = e^{i\boldsymbol{k}\cdot\boldsymbol{x}} = e^{i(k_1x+k_2y)} = e^{ik_1x}e^{ik_2y}\,. \tag{1.62}$$

The last form on the right is often termed a "tensor product" basis because its two factors are defined in an uncoupled way on two separate copies of subsets of $\mathbb{R}^1$.

To begin construction of the Galerkin form of Eqs. (1.60) we start with the simplest, Eq. (1.60c), the divergence-free condition, or continuity equation. We substitute Eqs. (1.61a) and (1.61b) into this equation and commute summation and differentiation to obtain

$$i\sum_{\boldsymbol{k}}^{\infty}\left(k_1a_{\boldsymbol{k}} + k_2b_{\boldsymbol{k}}\right)\varphi_{\boldsymbol{k}} = 0\,.$$

Since this must hold at all points of Ω (or, at least a.e. in Ω), and $\varphi_{\boldsymbol{k}} \not\equiv 0$, we must have

$$k_1a_{\boldsymbol{k}} + k_2b_{\boldsymbol{k}} = 0 \qquad \forall\ \boldsymbol{k}\,. \tag{1.63}$$

Similarly, if we substitute these expansions into the x-momentum equation (1.60a), we obtain

$$\frac{\partial}{\partial t}\sum_{\boldsymbol{\ell}} a_{\boldsymbol{\ell}}\varphi_{\boldsymbol{\ell}} + \frac{\partial}{\partial x}\sum_{\boldsymbol{\ell},\boldsymbol{m}} a_{\boldsymbol{\ell}}a_{\boldsymbol{m}}\varphi_{\boldsymbol{\ell}}\varphi_{\boldsymbol{m}} + \frac{\partial}{\partial y}\sum_{\boldsymbol{\ell},\boldsymbol{m}} a_{\boldsymbol{\ell}}b_{\boldsymbol{m}}\varphi_{\boldsymbol{\ell}}\varphi_{\boldsymbol{m}} =$$

$$-\frac{\partial}{\partial x}\sum_{\boldsymbol{\ell}} c_{\boldsymbol{\ell}}\varphi_{\boldsymbol{\ell}} + \frac{1}{Re}\left[\frac{\partial^2}{\partial x^2}\sum_{\boldsymbol{\ell}} a_{\boldsymbol{\ell}}\varphi_{\boldsymbol{\ell}} + \frac{\partial^2}{\partial y^2}\sum_{\boldsymbol{\ell}} a_{\boldsymbol{\ell}}\varphi_{\boldsymbol{\ell}}\right].$$

Again commuting summation and differentiation yields

$$\sum_{\boldsymbol{\ell}} \dot{a}_{\boldsymbol{\ell}}\varphi_{\boldsymbol{\ell}} + i\sum_{\boldsymbol{\ell},\boldsymbol{m}}(\ell_1+m_1)a_{\boldsymbol{\ell}}a_{\boldsymbol{m}}\varphi_{\boldsymbol{\ell}}\varphi_{\boldsymbol{m}} + i\sum_{\boldsymbol{\ell},\boldsymbol{m}}(\ell_2+m_2)a_{\boldsymbol{\ell}}b_{\boldsymbol{m}}\varphi_{\boldsymbol{\ell}}\varphi_{\boldsymbol{m}} = -i\sum_{\boldsymbol{\ell}}\ell_1 c_{\boldsymbol{\ell}}\varphi_{\boldsymbol{\ell}} - \frac{1}{Re}\sum_{\boldsymbol{\ell}}(\ell_1^2+\ell_2^2)\,a_{\boldsymbol{\ell}}\varphi_{\boldsymbol{\ell}}\,.$$

It can be seen that solutions (the $a_{\boldsymbol{\ell}}$s, $b_{\boldsymbol{\ell}}$s, $c_{\boldsymbol{\ell}}$s) to this equation can be complex, but we are only interested in those that are real if we are considering solutions corresponding to physical fluid flow. Indeed, if we had used sine and cosine as basis functions rather than the otherwise more convenient complex exponentials, this would not have been a concern. Furthermore, the reader will recall that sines and cosines may be expressed in terms of complex exponentials in any case (and conversely). This suggests that a means of avoiding the potentially complex solutions should be available. In fact, all that is required is setting $a_{-\boldsymbol{k}} = \overline{a}_{\boldsymbol{k}}$, where the overbar denotes complex conjugate, and the imaginary parts become zero (see, *e.g.*, [15]).

This permits us to consider the preceding equation without the imaginary factors i and proceed formally. (This is not exactly what is actually done in a computational setting, but it is sufficient for our analytical purposes.) We now form inner products of each basis function with the above equation and use orthonormality of $\{\varphi_{\boldsymbol{k}}\}$ to obtain

$$\dot{a}_{\boldsymbol{k}} + \sum_{\boldsymbol{\ell},\boldsymbol{m}} A^{(1)}_{\boldsymbol{k\ell m}}a_{\boldsymbol{\ell}}a_{\boldsymbol{m}} + \sum_{\boldsymbol{\ell},\boldsymbol{m}} B^{(1)}_{\boldsymbol{k\ell m}}a_{\boldsymbol{\ell}}b_{\boldsymbol{m}} = -k_1c_{\boldsymbol{k}} - \frac{|\boldsymbol{k}|^2}{Re}a_{\boldsymbol{k}}\,, \quad \forall\, \boldsymbol{k}\,. \tag{1.64}$$

In these equations the $A^{(1)}_{\boldsymbol{k\ell m}}$s and $B^{(1)}_{\boldsymbol{k\ell m}}$s, sometimes termed *Galerkin triple products*, are defined as, for example,

$$A^{(1)}_{\boldsymbol{k\ell m}} \equiv (\ell_1+m_1)\int_\Omega \varphi_{\boldsymbol{k}}\varphi_{\boldsymbol{\ell}}\varphi_{\boldsymbol{m}}\,dV\,, \tag{1.65}$$

where the superscript (1) denotes the x-momentum equation. Clearly, an analogous result holds for the y-momentum equation:

$$\dot{b}_{\boldsymbol{k}} + \sum_{\boldsymbol{\ell},\boldsymbol{m}} A^{(2)}_{\boldsymbol{k\ell m}}b_{\boldsymbol{\ell}}a_{\boldsymbol{m}} + \sum_{\boldsymbol{\ell},\boldsymbol{m}} B^{(2)}_{\boldsymbol{k\ell m}}b_{\boldsymbol{\ell}}b_{\boldsymbol{m}} = -k_2c_{\boldsymbol{k}} - \frac{|\boldsymbol{k}|^2}{Re}b_{\boldsymbol{k}}\,. \tag{1.66}$$

At this point we should recall that in both of the two forms of the N.–S. equations considered previously (Leray projected and weak) it was possible to eliminate pressure from the momentum equations, suggesting that this should be possible also in the present case. If we view Eqs. (1.64) and (1.66) as a vector equation for the vector $(a_{\boldsymbol{k}}, b_{\boldsymbol{k}})$ of Fourier coefficients, we can take the dot product of these equations with the wavevector $\boldsymbol{k} = (k_1, k_2)$ to obtain

$$\begin{aligned} k_1\dot{a}_{\boldsymbol{k}} + k_2\dot{b}_{\boldsymbol{k}} \;+ & \sum_{\boldsymbol{\ell},\boldsymbol{m}}\left[k_1\left(A^{(1)}_{\boldsymbol{k\ell m}}a_{\boldsymbol{\ell}}a_{\boldsymbol{m}} + B^{(1)}_{\boldsymbol{k\ell m}}a_{\boldsymbol{\ell}}b_{\boldsymbol{m}}\right) + k_2\left(A^{(2)}_{\boldsymbol{k\ell m}}b_{\boldsymbol{\ell}}a_{\boldsymbol{m}} + B^{(2)}_{\boldsymbol{k\ell m}}b_{\boldsymbol{\ell}}b_{\boldsymbol{m}}\right)\right] \\ = & \; -\left(k_1^2+k_2^2\right)c_{\boldsymbol{k}} - \frac{|\boldsymbol{k}|^2}{Re}\left(k_1a_{\boldsymbol{k}} + k_2b_{\boldsymbol{k}}\right) \qquad \forall\, \boldsymbol{k}\,. \end{aligned}$$

From Eq. (1.63) we see that the first term on the left-hand side and the second on the right-hand side are both zero, and moreover, we can solve what remains for $c_{\boldsymbol{k}}$ in terms of only the $a_{\boldsymbol{k}}$s and $b_{\boldsymbol{k}}$s:

$$c_{\boldsymbol{k}} = -\frac{1}{|\boldsymbol{k}|^2}\sum_{\boldsymbol{\ell},\boldsymbol{m}}\left[k_1\left(A^{(1)}_{\boldsymbol{k\ell m}}a_{\boldsymbol{\ell}}a_{\boldsymbol{m}} + B^{(1)}_{\boldsymbol{k\ell m}}a_{\boldsymbol{\ell}}b_{\boldsymbol{m}}\right) + k_2\left(A^{(2)}_{\boldsymbol{k\ell m}}b_{\boldsymbol{\ell}}a_{\boldsymbol{m}} + B^{(2)}_{\boldsymbol{k\ell m}}b_{\boldsymbol{\ell}}b_{\boldsymbol{m}}\right)\right]\,. \tag{1.67}$$

This implies, as we expected, that Eqs. (1.64) and (1.66) can be expressed in a form that is independent of pressure, just as was true in the previous cases. But beyond this it explicitly demonstrates the quantitative dependence of pressure on nonlinear aspects of the velocity field—in Fourier space. Furthermore, Eq. (1.67) can be substituted into these equations without altering their form.

Independent of this, it is worthwhile to examine some of the details of the Galerkin form of the N.–S. equations. We first make the obvious observation that this form comprises a system of equations for the time evolution of the Fourier coefficients of the velocity components, and as such is a dynamical system. Furthermore, at each instant in time for which $a_{\boldsymbol{k}}(t)$ and $b_{\boldsymbol{k}}(t)$ are known the Fourier coefficients for pressure can be calculated directly from Eq. (1.67), and these coefficients can be inserted into Eqs. (1.61) to obtain values of u, v and p at any point $\boldsymbol{x} \in \Omega$. Of course, in practice these Fourier representations can contain only a finite number N of terms, so the results are only approximate. But at least for solutions possessing a high degree of regularity, the Fourier representations converge very rapidly (in fact, exponentially fast), so not many terms are needed to obtain accurate approximations. For the reader interested in this feature, as well as numerous other details associated with these types of approximations, the monograph by Canuto *et al.* [28] is recommended.

It is also of interest to note that a Galerkin procedure might be applied to the Leray-projected form of the N.–S. equations given in Eq. (1.55). The natural basis set for this is the set of eigenfunctions of the Stokes operator, but other bases might be used as long as some provision is made regarding satisfaction of the divergence-free constraint.

Our main purpose in presenting the Galerkin form of the N.–S. equations is to demonstrate how this Fourier-space representation can be used to deduce qualitative mathematical (and physical) features of N.–S. flows. Here, we consider only the x-momentum equation (1.64); but the same treatment applies to y-momentum, and extension to 3-D flows occurs in a natural way. We begin by temporarily neglecting all nonlinear terms in Eq. (1.64). Then what remains is

$$\dot{a}_{\boldsymbol{k}} = -\frac{|\boldsymbol{k}|^2}{Re} a_{\boldsymbol{k}} \,, \tag{1.68}$$

the solution of which is

$$a_{\boldsymbol{k}}(t) = a_{\boldsymbol{k}}(0) e^{-\frac{|\boldsymbol{k}|^2}{Re} t} \,, \tag{1.69}$$

where

$$a_{\boldsymbol{k}}(0) \equiv \int_{\Omega} u_0(\boldsymbol{x}) e^{-i\boldsymbol{k}\cdot\boldsymbol{x}} \, dV \,. \tag{1.70}$$

Clearly, this solution decays in time and approaches zero as $t \to \infty$. Moreover, the rate at which this occurs is $|\boldsymbol{k}|^2/Re$. In particular, for fixed Re higher-wavenumber Fourier modes decay faster than do lower ones. From a mathematical perspective this suggests (but does not prove) convergence of the Fourier series representation and, hence, existence of solutions of the form Eqs. (1.61). On the other hand, if $|\boldsymbol{k}|$ is fixed, then the rate of decay of $a_{\boldsymbol{k}}$ <u>decreases</u> with increasing Re. Now if we recall that the right-hand side of Eq. (1.68) is precisely the Fourier-space representation of the viscous terms of Eq. (1.60a), we see that we can associate the rate of decay of $a_{\boldsymbol{k}}$ with viscous dissipation; in particular, increasing Re implies decreasing viscous dissipation, and conversely.

We now consider effects of the nonlinear terms. To do this we first drop the linear viscous dissipation term from Eq. (1.64), and we also drop the term containing $c_{\boldsymbol{k}}$ arising from the pressure gradient in Eq. (1.60a) since, as can be seen from Eq. (1.67), this is directly related to the nonlinear terms, so nothing is lost at the qualitative level by ignoring it. Then we are left with

$$\dot{a}_{\boldsymbol{k}} = -\sum_{\boldsymbol{\ell},\boldsymbol{m}} \left[A^{(1)}_{\boldsymbol{k}\boldsymbol{\ell}\boldsymbol{m}} a_{\boldsymbol{\ell}} a_{\boldsymbol{m}} + B^{(1)}_{\boldsymbol{k}\boldsymbol{\ell}\boldsymbol{m}} a_{\boldsymbol{\ell}} b_{\boldsymbol{m}} \right] . \tag{1.71}$$

We first observe that when $\boldsymbol{\ell} = \boldsymbol{m}$ a quadratic term appears in the equation. It is worthwhile to consider the effects of this alone since an analytical solution can be obtained, and this will provide at least some insight into the qualitative behavior of the nonlinear terms in general. Thus, we solve the initial-value problem

$$\dot{a}_{\boldsymbol{k}} = -A^{(1)} a_{\boldsymbol{k}}^2 \,, \tag{1.72}$$

with $a_{\boldsymbol{k}}(0)$ again given by Eq. (1.70) and subscript notation for $A^{(1)}_{\boldsymbol{k\ell m}}$ suppressed. We leave as a simple exercise to the reader demonstration that

$$a_{\boldsymbol{k}}(t) = \frac{1}{A^{(1)}t + 1/a_{\boldsymbol{k}}(0)} \,.$$

It is clear from this that if $A^{(1)}a_{\boldsymbol{k}}(0) > 0$, then $|a_{\boldsymbol{k}}| \to 0$ as $t \to \infty$, although only algebraically. But if $A^{(1)}$ and $a_{\boldsymbol{k}}(0)$ are of opposite signs, $a_{\boldsymbol{k}}(t) \to \infty$ can occur in finite time. This implies a potential for very ill behavior (including nonexistence after only a finite time) of N.–S. solutions. We will see in the next section that more sophisticated analyses of the 2-D case than considered here show this does <u>not</u> actually occur, but in 3D it is still considered a possibility.

There is a further aspect of the behavior of the terms on the right-hand side of Eq. (1.71) that deserves mention. It is that such nonlinearities can generate new Fourier modes not present in the original representation (or in initial data for the problem). Consideration of this will be important in Chap. 2 when we begin studies of discretization of the N.–S. equations. Here we consider only the first advective term of the x-momentum equation, $(u^2)_x$, and recall its Fourier representation:

$$(u^2)_x = \frac{\partial}{\partial x} \sum_{\boldsymbol{\ell},\boldsymbol{m}}^{N} a_{\boldsymbol{\ell}} a_{\boldsymbol{m}} \varphi_{\boldsymbol{\ell}} \varphi_{\boldsymbol{m}} \,,$$

written now for only a finite number N of Fourier modes, as would be required for computer implementation.

Now if the basis set $\{\varphi_{\boldsymbol{k}}\}$ is similar to complex exponentials or trigonometric functions we see that, *e.g.*,

$$\varphi_{\boldsymbol{\ell}}(\boldsymbol{x})\varphi_{\boldsymbol{m}}(\boldsymbol{x}) = e^{i\boldsymbol{\ell}\cdot\boldsymbol{x}} e^{i\boldsymbol{m}\cdot\boldsymbol{x}} = e^{i(\boldsymbol{\ell}+\boldsymbol{m})\cdot\boldsymbol{x}} \,.$$

Since each of $\boldsymbol{\ell}$ and $\boldsymbol{m}$ can be as large as N, their sum can be greater than N, and the corresponding nonlinear term $a_{\boldsymbol{\ell}} a_{\boldsymbol{m}}$ generates solution behaviors that cannot be resolved by the given N-mode representation. We will return to this in Chap. 2. We note here that this does <u>not</u> occur for the Galerkin procedure due to the values taken on by the Galerkin triple products and the intrinsically global nature of the Galerkin construction; but it <u>does</u> occur for the various Fourier collocation methods that are widely used as a more efficient alternative to the Galerkin procedure (see [28]). Moreover, it is easily argued that this must occur for finite-difference and finite-volume methods as well.

Finally, we emphasize that the actual time evolution of each of the $a_{\boldsymbol{k}}$s is effected by combinations of all the abovementioned behaviors, and as a consequence solutions can be very complicated—and very difficult to simulate. (The reader may wish to combine Eqs. (1.68) and (1.72), find an exact solution, and analyze its properties.)

1.3.4 The main well-known N.–S. solution results

In this section we will present the key results of this chapter, the main theorems on existence, uniqueness and regularity of solutions to the Navier–Stokes equations. While practitioners of CFD have a general tendency to pay little attention to such results we emphasize, as we have already in these lectures, that it is best to heed and, to the greatest extent possible, use the results from pure mathematics. At the same time it is recognized that these are often difficult to comprehend, and it has been the purpose of the foregoing sections to develop the material that should now make at least a superficial understanding possible.

The theorems we will state and discuss can be classified as pertaining to 2-D or to 3-D flows, and as to whether they relate to weak or strong solutions to the N.–S. equations. With regard to these classifications, it is usually said that essentially everything worth proving for 2-D problems has been proven. In particular, existence and uniqueness of solutions, both weak and strong, have been proven for all time beyond any specified initial time for quite reasonable (physically) problems. In 3D, long-time existence can be demonstrated for weak solutions, but uniqueness has not been proven for this case. On the other hand, only short-time existence has been proven for 3-D strong solutions, but it is known that these are unique. Especially with regard to existence of strong solutions, the constraints that must be imposed on the shape of the domain Ω (in particular, smoothness of $\partial\Omega$), the boundary and initial conditions, and particularly on body-force terms and Reynolds number ($\sim$viscosity), can be quite stringent if solutions are to be proven to exist for any but very short times.

Thus, the longstanding mathematical problem, first identified by Leray in the early 1930s, is proof of existence and uniqueness of strong (but not necessarily classical) solutions to the N.–S. equations for long times, and under reasonable physical conditions. It is of interest to mention that this problem is considered to be one of the most important problems in mathematics for the 21^{st} Century, and it carries a \$1 million "price on its head." Quite literally, the first person who succeeds in rigorously solving this problem will be awarded a prize of a million dollars!

In earlier subsections we constructed weak forms of the N.–S. equations, and we defined the strong form as studied by mathematicians: in the next two subsections we will provide a few of the important well-known results for solutions to the Navier–Stokes equations, first in 2D and then in 3D in this framework. (This treatment closely follows that of Foias *et al.* [15], and as noted in that work, the results were already known to Ladyzhenskaya [4].) Then, in a final subsection we will summarize some basic conclusions that impact the choice of numerical algorithms for solving the N.–S. equations, and at the same time provide an indication of what we should expect regarding the qualitative nature of solutions.

Solutions to the N.–S. Equations in 2D

As we noted above, the 2-D N.–S. problem is generally viewed as having been completely solved in the mathematical sense. Here we present two of the main results that have contributed to this understanding. We begin with the following theorem on existence $\underline{\text{and uniqueness}}$ for weak solutions, *i.e.*, solutionsto Eq. (1.58) which we repeat here for easy reference:

$$\frac{d}{dt}\langle \boldsymbol{u}(t), \boldsymbol{v}\rangle + \nu\langle \boldsymbol{u}(t), \boldsymbol{v}\rangle_{V(\Omega)} + b(\boldsymbol{u}(t), \boldsymbol{u}(t), \boldsymbol{v}) = \langle \boldsymbol{F}_B, \boldsymbol{v}\rangle\,.$$

Theorem 1.1 *Let $\boldsymbol{u}_0$, $\boldsymbol{F}_B$ and $t_f > 0$ be given such that*

$$\boldsymbol{u}_0(\boldsymbol{x}) \in H(\Omega)\,, \qquad \textit{and} \qquad \boldsymbol{F}_B(\boldsymbol{x},t) \in L^2(0,t_f;H)\,.$$

Then $\exists$ a $\underline{\text{unique}}$ solution $\boldsymbol{u} = (u_1,u_2)^T$ of Eq. (1.58) $\ni$

$$u_i\,,\ \frac{\partial u_i}{\partial x_j} \in L^2(\Omega\times(0,t_f))\,, \quad i,j = 1,2\,,$$

and $\boldsymbol{u}$ is continuous from $[0,t_f]$ into H. Moreover, the following energy equation holds for $t \in [0,t_f]$:

$$\frac{1}{2}\frac{d}{dt}\|u(t)\|^2_{L^2} + \nu\|u(t)\|^2_{H^1} = \langle \boldsymbol{F}_B(t), \boldsymbol{u}(t)\rangle\,.$$

There are a number of remarks to be made regarding this theorem. First, no restriction has been placed on the value of the final time, t_f, other than it be greater than zero—backward in time N.–S. problems are ill posed, and $t_f = 0$ would be trivial. Likewise, no restrictions are placed on the domain Ω. Second, from the definition of $H(\Omega)$ (Eq. (1.45)), it is clear that $\boldsymbol{u}_0$, the initial data, must be divergence free, and that it must satisfy whatever boundary conditions have been imposed on $\boldsymbol{u}$. Thus, initial and boundary data are *compatible* in the usual sense of elementary PDE theory, modulo the fact that this need only hold a.e. in a neighborhood of $\partial\Omega$. Furthermore, again from the definition of the function space H, we see that both $\boldsymbol{u}_0$ and $\boldsymbol{F}_B$ must be in $L^2(\Omega)$ with respect to spatial coordinates, and $\boldsymbol{F}_B$ must also be in L^2 with respect to the time interval $(0,t_f)$; moreover, F_B must be divergence free. We remark that this last requirement often cannot be realized in physical problems.

We next observe that the theorem statement conclusions indicate that the solution to the weak form of the N.–S. equations (1.58) is unique. Rather generally, weak solutions typically are <u>not</u> unique. Moreover, the fact that both u_i and $\partial u_i/\partial x_j$, $i,j = 1,2$ are in L^2 for all time implies that the solutions are actually in $H^1(\Omega)$. Thus, we see that the Navier–Stokes equations in this 2-D setting are providing smoothing; the solution operator is mapping L^2 initial data (and forcing function) into a solution in H^1. In addition, the fact that $\boldsymbol{u}$ is continuous for $[0,t_f]$ to H means that if only a small change in time is considered, then the change of the L^2 norm of $\boldsymbol{u}$ during this time will be small. Finally, satisfaction of the energy equality shows that the L^2 norm of $\boldsymbol{u}$ is bounded for all time, provided $\boldsymbol{F}_B \in L^2$, as required in the theorem statement.

We now state an analogous result concerning strong solutions. We remind the reader of Def. 1.6 for a strong solution, *viz.*, a solution possessing sufficient derivatives to permit evaluation of the corresponding differential equation a.e. in the domain Ω, leading to an ability to calculate a desired L^p norm of the equation "residual."

Theorem 1.2 *Suppose $\boldsymbol{u}_0$, $\boldsymbol{F}_B$ are given and are such that*

$$\boldsymbol{u}_0(\boldsymbol{x}) \in V(\Omega)\,, \qquad \textit{and} \qquad \boldsymbol{F}_B(\boldsymbol{x},t) \in L^2(0,t_f;H)\,,$$

for $t_f > 0$. Then $\exists$ a <u>unique</u> (strong) solution $\boldsymbol{u} = (u_1,u_2)^T$ with

$$u_i\,, \frac{\partial u_i}{\partial t}, \frac{\partial u_i}{\partial x_j}, \frac{\partial^2 u_i}{\partial x_j \partial x_k} \in L^2(\Omega\times(0,t_f))\,, \quad i,j,k = 1,2\,,$$

and $\boldsymbol{u}$ is continuous from $[0,t_f]$ into V.

As was true for the preceding theorem, this result requires considerable explanation for those not well versed in the mathematics involved. We should first note that the requirement on $\boldsymbol{u}_0$ is stronger in this case; in particular, $\boldsymbol{u}_0$ must now be in V rather than in H, implying that it must be once differentiable a.e. and thus in $H^1(\Omega)$ instead of only in L^2. On the other hand, requirements on the forcing function $\boldsymbol{F}_B$ remain the same as in the previous case.

As a result of this change in required smoothness of initial data, the first derivative with respect to time and the second with respect to space are now in L^2, implying that Eqs. (1.60) can be evaluated a.e. in Ω. Hence, the L^2 norm of the differential equation exists, and the solution $\boldsymbol{u}$ is strong in the sense of Def. 1.6. Here, we have relaxed the C^1 condition in time of that definition to continuity in time, but we still must require that $\partial u_i/\partial t$ exist a.e. in Ω for (almost) every $t \in (0,t_f]$. We should also again observe the smoothing properties of the N.–S. operator. Namely, in the present case initial data were in H^1, and the solution evolved so as to be in H^2; this should be expected, at least in an heuristic and qualitative way, due to the viscous dissipation of the diffusion terms of Eqs. (1.60).

There are two other important points to make at this time. The first is that because both weak and strong solutions can be proven to be unique, it follows that, in fact, there is <u>only one</u> solution—and it is strong. It is easily checked (and we leave this as an exercise) that strong (and even classical) solutions satisfy the weak form of the equations, as noted earlier in our initial discussions of strong solutions (but, of course, the converse is not true). Thus, if both weak and strong solutions exist and are unique, the unique strong solution must also be the unique weak one. We comment that it is often the practice in PDE theory to take the path of first proving weak results followed by proving strong results, and then attempting to show they are the same.

The final remark of this section is emphasis that although we have indicated existence of unique, strong solutions to the N.–S. equations (1.60), for all time in 2D, this does not imply that such solutions are any more regular than H^2 in space and H^1 in time. This has important implications concerning the choice of numerical methods to be used in implementing CFD algorithms (as was recognized already by Richtmyer and Morton [29], but seems to have been ignored in recent years) because it indicates that the Taylor series needed for truncation error analysis may not exist, and in turn, use of high-order methods is not likely to produce uniformly superior results. But in addition, this suggests that mollification (often termed *regularization*, especially in the numerical analytic context) will probably be needed, especially when coarse discretizations are used, to control growth of effects from unrepresented high modes arising from nonexistence of high-order derivatives and from nonlinear terms. Details of this will be provided in Chap. 2.

Solutions to the N.–S. Equations in 3D

The results we present in this subsection are in many respects quite similar to those given above for two space dimensions. But there are important differences as we will note in the following discussions. We again first provide results associated with weak solutions, and follow this with corresponding results for strong solutions.

Theorem 1.3 *Let $\boldsymbol{u}_0$, $\boldsymbol{F}_B$ and $t_f > 0$ be given and such that*

$$\boldsymbol{u}_0(\boldsymbol{x}) \in H(\Omega)\,, \qquad \textit{and} \qquad \boldsymbol{F}_B(\boldsymbol{x},t) \in L^2(0,t_f;H)\,.$$

Then $\exists$ at least one solution $\boldsymbol{u} = (u_1,u_2,u_3)^T$ to Eq. (1.58) $\ni$

$$u_i, \frac{\partial u_i}{\partial x_j} \in L^2(\Omega\times(0,t_f))\,, \qquad i,j = 1,2,3,$$

and $\boldsymbol{u}$ is $\underline{\text{weakly}}$ continuous from $[0,t_f]$ into H.

We first remind the reader that weak continuity was introduced in Def. 1.2 in terms of linear functionals based on C_0^∞ test functions. But here we will replace these test functions with functions $\boldsymbol{v}(\boldsymbol{x}) \in V(\Omega)\,(\subset H^1)$. Then weak continuity of $\boldsymbol{u}$ implies that $\forall\ \boldsymbol{v} \in V$ the mapping

$$t \longmapsto \langle \boldsymbol{u}(\boldsymbol{x},t), \boldsymbol{v}(\boldsymbol{x})\rangle = \int_\Omega \boldsymbol{u}(\boldsymbol{x},t)\cdot\boldsymbol{v}(\boldsymbol{x})\,dV$$

must be continuous (in the usual sense). We should note that this is different from the 2-D case in which continuity of $\boldsymbol{u}$ into H was strong. A second difference from the 2-D case is that weak solutions in 3D may not be unique (it is currently not known whether they are, or not). Finally, in place of the energy equality given in Theorem 1.1, only an energy inequality can be proven for the 3-D case:

$$\frac{1}{2}\frac{d}{dt}\|u(t)\|^2_{L^2} + \nu\|u(t)\|^2_{H^1} \le \langle \boldsymbol{F}_B(t), \boldsymbol{u}(t)\rangle\,,$$

in which all differentiations must be taken in the sense of distributions constructed with L^2 and H^1 inner products, as appropriate for individual terms.

Theorem 1.3 clearly is not a very favorable result; its one good feature is that there is no restriction on t_f; *i.e.*, we have global in time existence, but possibly nonuniqueness, of the weak solution(s) in 3D. We will now see that, in a sense, just the opposite holds for strong solutions.

Theorem 1.4 *Let $\boldsymbol{u}_0$, $\boldsymbol{F}_B$ and $t_f > 0$ be given and such that*

$$\boldsymbol{u}_0(\boldsymbol{x}) \in V(\Omega)\,, \qquad \textit{and} \qquad \boldsymbol{F}_B(\boldsymbol{x},t) \in L^2(0,t_f;H)\,.$$

Then $\exists$ $t_* \in (0,t_f]$ and depending on $\boldsymbol{u}_0$, $\boldsymbol{F}_B$, ν and $\Omega \ni$ there is a $\underline{\text{unique}}$ strong solution $\boldsymbol{u} = (u_1,u_2,u_3)^T$ to Eqs. (1.60) *with the properties*

$$u_i, \frac{\partial u_i}{\partial t}, \frac{\partial u_i}{\partial x_j}, \frac{\partial^2 u_i}{\partial x_j \partial x_k} \in L^2(\Omega\times(0,t_*))\,, \qquad i,j,k = 1,2,3\,,$$

and $\boldsymbol{u}$ is a (strongly) continuous function from $[0,t_*)$ into V.

It is of interest to note that by arguments analogous to those used in 2D, we see that uniqueness of this solution implies that there is no other solution—strong or weak—on the interval $[0, t_*)$. But we cannot prove existence of any strong solution beyond the time t_*, while time of existence for the possibly nonunique weak solution(s) is unlimited.

We should again observe that just as in 2D, we bought better solution behavior by requiring more regular initial data. But in the present case we can only prove such a solution exists for a finite time, a time which is influenced by essentially everything associated with the problem. Constantin and Foias [13] can be consulted for more details on this aspect of N.–S. solutions; here we simply note that, *e.g.*, as Re becomes large (and hence dissipation, *i.e.*, ν, becomes small), the provable time of existence of a strong solution becomes short—as one might expect.

Some Conclusions

We end this section with some basic conclusions that are easily deduced from the foregoing discussions. First, it is clear in both 2D and 3D that the nature of initial data has a significant impact on ultimate regularity of N.–S. solutions. In particular, for the theorems quoted here it is necessary that initial data be divergence free and be compatible with boundary conditions. Without these restrictions it is not clear that even existence can be proven; but at the very least, if these requirements are not met, smoothness of N.–S. solutions will be reduced. Thus, the first step in any computational algorithm for solving the incompressible N.–S. equations should be mollification to achieve boundary compatibility followed by projection of the mollified initial data to a divergence-free subspace via Leray projection.

Furthermore, there is no prescription for initial pressure in any of the theorem statements. In the case of weak solutions pressure does not enter the problem in any direct way, so this is to be expected. But for strong solutions this could be different, and in either case it leaves open the question of how to handle this in the context of numerical methods that generally do not employ divergence-free bases. In particular, it should be clear that regardless of any algorithmic details, pressure (initial, or otherwise) must be obtained from the velocity field (recall the Galerkin representation of the N.–S. equations, for example) so as to be consistent with the divergence-free condition.

The next thing to note is that the forcing function $\boldsymbol{F}_B$ has also been required to be divergence free; it is a mapping from H to L^2. In physical problems, as mentioned earlier, this might not be the case—and there may be nothing that can be done to improve the situation. Hence, solution behavior may be worse than indicated by the theorems of this section. On the other hand, with $\boldsymbol{F}_B \equiv 0$, as often occurs, the given results probably represent a "worst case." But even so, as we have already noted, especially in the most important case of 3-D flows at high Re, it is unlikely that solutions will be classical. As a consequence, much of Taylor series based numerical analysis is inapplicable without modification. We will treat some details of this in the next chapter.

1.4 Summary

In this introductory chapter we have provided some basic notions from functional analysis—definitions of various function spaces, linear functionals and distributions—that permit us to understand the concepts of weak and strong solutions to partial differential equations. In addition we have presented various mathematical techniques that are crucial for being able to analyze the N.–S. equations: mollification, Helmholtz–Leray decomposition, Leray projection and the Galerkin procedure. These tools permitted us to discuss weak and strong solutions to PDEs in general and to construct various forms of the N.–S. equations. We then stated and discussed some of the main theorems describing existence, uniqueness and regularity properties of N.–S. solutions. We have emphasized that in three space dimensions the theory of these equations is incomplete: existence of weak solutions can be proven for all time, but it is not known whether such solutions are unique; existence and uniqueness of strong solutions can be proven only for finite time. Finally, we have stressed that even strong solutions are considerably different from classical ones, and generally do not possess sufficient regularity for typical Taylor series based approaches to error analysis. This casts doubt on, among other things, the utility of attempting to employ higher-order methods when approximating solutions to the Navier–Stokes equations, and it implies that considerable care must be exercised in constructing solution algorithms and in attempting to validate computed results.

Chapter 2

Special Numerical Difficulties of the Navier–Stokes Equations

In this chapter we present several mathematical/numerical analytic topics that are quite specific to treatments of the Navier–Stokes equations. We begin by summarizing essentially all the different forms into which the N.–S. equations have been cast during attempts to solve them. We follow this with a description of the various grid structures one can consider and compare advantages and disadvantages of these as they relate to the problem of pressure-velocity coupling. Finally, we treat the problem associated with cell-Re restrictions on difference approximations and grid spacing, and we conclude the chapter with discussions pertaining to the related problem of aliasing.

2.1 Forms of the Navier–Stokes Equations

The N.–S. equations have, through the years, been cast in many different forms during attempts to solve them, especially in 2D. Here we will review essentially all of these. But we note that the main purpose of this is to allow the reader to recognize that in the current era of very powerful computing hardware it is seldom that anything but the so-called "primitive-variable" form of the equations is employed. Thus, we will begin this section with this basic form. We follow this with the stream function/vorticity formulation that once was widely used for 2-D calculations, and we then consider the velocity/vorticity formulation. Finally, we return to the primitive-variable equations and consider various modifications of these that sometimes can provide specific advantages.

2.1.1 Primitive-variable formulation

We previously presented the N.–S. equations in Chap. 1 as

$$\rho \frac{D\boldsymbol{u}}{Dt} = -\nabla p + \mu \Delta \boldsymbol{u} + \boldsymbol{F}_{\scriptscriptstyle B}\,, \tag{2.1a}$$

$$\nabla \cdot \boldsymbol{u} = 0\,. \tag{2.1b}$$

Here, ρ and μ are constant density and viscosity, and $\boldsymbol{F}_{\scriptscriptstyle B}$ is a body force. The dependent variables in this equation are the velocity vector $\boldsymbol{u} = (u, v, w)^T$ and the pressure p. This set of variables is termed "primitive" since in most typical laboratory experiments it includes the physical variables that can be directly measured. Moreover, other quantities of interest in a fluid flow (*e.g.*, vorticity or circulation) can be computed directly from these. It is important to recognize, however, that the body force $\boldsymbol{F}_{\scriptscriptstyle B}$ may contain further dependent variables, *e.g.*, temperature T, and when this is the case additional transport (or other) equations will be required. For the present we ignore such added complications beyond noting that typical body forces arise from buoyancy, rotation and electromagnetic fields.

A fundamental observation associated with Eqs. (2.1) is that the momentum equations (2.1a) should be viewed as the equations that describe the velocity field, while the divergence-free condition (2.1b) provides an equation for pressure. This should seem rather natural in light of our discussions in Chap. 1 where it was shown that the momentum equations can be solved for the velocity field in the absence of pressure gradient terms. This leaves only $\nabla \cdot \boldsymbol{u} = 0$ for finding the pressure; indeed, it is often said that "pressure enforces the divergence-free constraint in an incompressible flow." But this equation does not explicitly contain the pressure, and this has caused significant problems from a computational standpoint; we will consider several treatments of these difficulties in the sequel.

The first attempts to rectify this problem arose from recognition that a Poisson equation for pressure (the so-called PPE) can be derived by calculating the divergence of Eq. (2.1a): we have

$$\rho \nabla \cdot \frac{D\boldsymbol{u}}{Dt} = -\nabla \cdot (\nabla p) + \mu \nabla \cdot \Delta \boldsymbol{u} + \nabla \cdot \boldsymbol{F}_B \,,$$

or after commuting the divergence operator with other differential operators wherever this is convenient (and appropriate) we have (after expanding the substantial derivative)

$$\rho \frac{\partial (\nabla \cdot \boldsymbol{u})}{\partial t} + \rho \nabla \cdot (\boldsymbol{u} \cdot \nabla \boldsymbol{u}) = -\Delta p + \mu \Delta (\nabla \cdot \boldsymbol{u}) + \nabla \cdot \boldsymbol{F}_B \,,$$

or

$$\Delta p = \nabla \cdot \boldsymbol{F}_B + \mu \Delta (\nabla \cdot \boldsymbol{u}) - \rho \left[\frac{\partial (\nabla \cdot \boldsymbol{u})}{\partial t} + \nabla \cdot (\boldsymbol{u} \cdot \nabla \boldsymbol{u}) \right] . \tag{2.2}$$

This implies that pressure can be calculated provided the velocity field is known, a result we earlier deduced from the Galerkin form of the N.–S. equations. But Eq. (2.2) requires boundary conditions at all points on the boundary of the solution domain Ω. This fact led to considerable misunderstandings regarding solution of the N.–S. equations (see, *e.g.*, Gresho and Sani [30]) and was one of the motivations for solving various alternative forms of the equations of motion we will discuss in subsequent sections.

Before turning to these, however, there are two main points that should be made regarding Eq. (2.2). The first is that if we assume that (2.1b) is satisfied, Eq. (2.2) collapses to

$$\Delta p = \nabla \cdot \boldsymbol{F}_B - \rho \nabla \cdot (\boldsymbol{u} \cdot \nabla \boldsymbol{u}) \,. \tag{2.3}$$

Moreover, if $\boldsymbol{F}_B \in H(\Omega)$, as required by hypotheses of the theorems concerning existence, uniqueness and regularity of N.–S. equation solutions given in Chap. 1, this becomes

$$\Delta p = -\rho \nabla \cdot (\boldsymbol{u} \cdot \nabla \boldsymbol{u}) \,.$$

Thus, if we have somehow obtained a divergence-free velocity field we can determine the pressure from this equation supplemented with appropriate boundary conditions, which we will treat in more detail in a later section. But we note here that on solid surfaces the pressure usually is not known and is typically a major part of the desired information to be obtained by solving the problem.

The second thing to observe regarding use of (2.2) or (2.3) is that it does not accomplish the desired task of forcing the velocity field to be divergence free. In fact, within the confines of the description given here it is necessary to already have a divergence-free velocity in order to use Eq. (2.3) (or the equation following it). Moreover, solutions to the momentum equations (2.1a) are not necessarily divergence free (and this is all the more true of their discrete representations). Clearly, if they were, the constraint (2.1b) would be unnecessary. But we have already seen in Chap. 1 a procedure for obtaining a divergence-free velocity field—Leray projection, and we will later see that this provides a crucial computational tool in the present context.

2.1.2 Stream function/vorticity formulations

Difficulties with satisfying the divergence-free condition and prescribing boundary conditions for pressure led researchers to seek formulations that did not include the pressure, and which automatically enforced the divergence-free constraint. The first, and probably the most successful, of these is the stream function/vorticity formulation. The stream function was widely used in analytical treatments of the N.–S. equations (*e.g.*, in boundary-layer theory), so this was a natural approach. But it possesses two major shortcomings, the first of which is that there is no straightforward extension of the 2-D formulation to 3D.

We begin this section with the basic, well-known 2-D treatment, and follow this with a brief discussion of a possible way to handle 3-D cases.

2-D Stream Function/Vorticity

Consider a velocity field $\boldsymbol{u} = (u, v)^T$ defined on a domain Ω and the divergence-free constraint

$$u_x + v_y = 0\,. \tag{2.4}$$

We wish to find a way to automatically satisfy this condition. It is clear for any function $\psi \in C^2(\Omega)$ that its mixed partial derivatives satisfy

$$\psi_{xy} = \psi_{yx}\,.$$

It follows that we should define the *stream function* ψ by

$$u = \psi_y\,, \quad \text{and} \quad v = -\psi_x\,. \tag{2.5}$$

Then Eq. (2.4) is automatically satisfied. Thus, if we could find a PDE satisfied by ψ, and related to properties of the flow field, we would be able to obtain a divergence-free velocity field via Eqs. (2.5).

To do this we first recall that in 2D the vorticity vector has only a single component,

$$\omega = v_x - u_y\,. \tag{2.6}$$

Now substitution of Eqs. (2.5) into this yields

$$\Delta\psi = -\omega\,, \tag{2.7}$$

so if we can find an equation for ω we will have obtained a formulation that automatically produces divergence-free velocity fields. While the formal equivalence of this formulation with the N.–S. equations requires rigorous proof, the construction we will employ is suggestive, and we will not be further concerned with this approach in any case.

In 3D we would take the curl of the momentum equations to produce the desired result. But in 2D it is easier to simply "cross differentiate" the two momentum equations, and then subtract the x-momentum equation from the y-momentum equation. In this case the equations are (in the absence of body forces)

$$\rho\,(u_t + uu_x + vu_y) = -p_x + \mu\Delta u\,, \tag{2.8a}$$

$$\rho\,(v_t + uv_x + vv_y) = -p_y + \mu\Delta v\,, \tag{2.8b}$$

and cross differentiation leads to

$$\rho\,(u_{yt} + u_yu_x + uu_{xy} + v_yu_y + vu_{yy}) = -p_{xy} + \mu\Delta u_y\,,$$

$$\rho\,(v_{xt} + u_xv_x + uv_{xx} + v_xv_y + vv_{xy}) = -p_{xy} + \mu\Delta v_x\,,$$

where we have assumed sufficient smoothness to permit changing the order of differentiation whenever this is convenient.

Next substract the first of these from the second to obtain

$$\rho(v_x - u_y)_t + (v_x - u_y)u_x + (v_{xx} - u_{xy})u + (v_x - u_y)v_y + (v_{xy} - u_{yy})v = \mu\Delta(v_x - u_y)\,.$$

We observe that the second and fourth terms on the left-hand side can be combined to give

$$(v_x - u_y)(u_x + v_y) = \omega\nabla\cdot\boldsymbol{u} = 0\,,$$

and what remains can be expressed as

$$\rho(\omega_t + u\omega_x + v\omega_y) = \mu\Delta\omega\,, \tag{2.9}$$

or, after using Eqs. (2.5),

$$\omega_t + \psi_y\omega_x - \psi_x\omega_y = \nu\Delta\omega\,. \tag{2.10}$$

This is the basic transport equation for vorticity, and it completes the system of PDEs comprising the stream function/vorticity formulation of the 2-D incompressible N.–S. equations.

Clearly, boundary and initial conditions must be prescribed to complete a well-posed problem for Eqs. (2.7) and (2.10). Initial conditions are needed only for the latter of these, and they can be computed directly from the initial velocity field using Eqs. (2.5) and (2.6). Construction of boundary conditions is less straightforward and, as it turns out, rather counterintuitive—the second of the major shortcomings of this approach alluded to earlier. Much effort was devoted to this and other aspects of solving these equations during the 1970s and 80s, but because they are now seldom used, we will not here provide further details.

Extension to 3D

Extension of the stream function/vorticity formulation to 3D is not straightforward for two main reasons. First, there is no natural definition of the stream function in 3D; in place of the stream function one must employ a (vector) velocity potential $\boldsymbol{\phi}$ such that $\boldsymbol{u} = \nabla\times\boldsymbol{\phi}$. But now the vorticity equation must be replaced with a system of three equations for the components of the vorticity vector, each of which is more complicated than Eq. (2.9) due to the "vortex stretching" terms that occur in 3D. The problems associated with boundary condition formulation are now even more formidable, and since this 3-D version is almost never used we will not devote any further space to it here. The interested reader may wish to consult, *e.g.*, Fletcher [31] for a complete treatment.

A final remark on stream function/vorticity formulations in general is worthwhile. It is that although they do accomplish the desired goal of automatic satisfaction of the divergence-free condition, they merely shift problems associated with pressure boundary conditions to (even more difficult—see, *e.g.*, Gresho [32]) problems for vorticity boundary conditions.

2.1.3 Velocity/vorticity formulations

We will treat this formulation only in 2D. The corresponding 3-D treatment is similar but more involved (again, because vorticity is a vector in 3D but only a scalar in 2D). Moreover, this approach has not been widely used; our main purpose in presenting it is simply to make the reader aware that such efforts have previously been made, and that there is little reason to re-invent these (basically flawed) techniques.

The 2-D vorticity transport equation has already been derived in the previous section and given as Eq. (2.9); we repeat this here in its more often-used form

$$\omega_t + u\omega_x + v\omega_y = \nu\Delta\omega\,. \tag{2.11}$$

This equation contains three unknowns, vorticity and the two components of the velocity vector. Thus we need two more equations. These are supplied by the definition of vorticity,

$$\omega = v_x - u_y\,, \tag{2.12}$$

and the divergence-free constraint,

$$u_x + v_y = 0\,. \tag{2.13}$$

This formulation is sometimes popular with mathematicians because the last two equations comprise a Cauchy–Riemann system for which a great deal is known from complex analysis. Indeed, this has been employed by Kreiss and Lorenz [33] to prove existence of periodic C^∞ solutions to the N.–S. equations via what is formally a numerical algorithm, allowing discretization step sizes to approach zero, and examining the properties of the limits. Actual numerical simulations using Eqs. (2.11)–(2.13) are relatively rare (an example can be found due to Gatski *et al.* [34]). This is mainly because of problems with vorticity boundary conditions, as already briefly indicated.

As a consequence of these and various related problems, which are exacerbated in 3D, we will give no further consideration to this formulation except to observe that although finding vorticity boundary conditions to produce a well-posed problem for the vorticity equation (2.11) can be difficult, using this equation on an outflow boundary as a boundary condition in conjunction with Eqs. (2.12) and (2.13) may provide a useful augmentation to the boundary treatment of various primitive-variable formulations. See, *e.g.*, Ziaei *et al.* [35], and references therein for a 3-D treatment of this.

2.1.4 Further discussions of primitive variables

As we have already emphasized, the primitive-variable form of the N.–S. equations is now employed almost exclusively, both in new research codes and in commercial software. It will be the only form treated in the remainder of these lectures. It is important to recognize, however, that this form, itself, has many different subforms that are formally equivalent analytically in the context of classical solutions but which can lead to different algorithmic behaviors when replaced by discrete approximations, especially to non-classical solutions. The goal of this section is to provide an overview of these various forms. We begin by noting that the information presented here has appeared at numerous times in the archival literature, and the curren treatment relies heavily on the work of Gresho [32]. We will concentrate on alternative forms of the advective terms and note, in passing, that additional forms of the diffusion terms can also be used—and often are, mainly in the context of finite-element discretizations.

For purposes of comparisons, we first express the N.–S. equations in what might be viewed as a "standard" form (termed the "advective/convective" form in [32]) for the case of constant kinematic viscosity:

$$\boldsymbol{u}_t + \boldsymbol{u}\cdot\nabla\boldsymbol{u} = -\nabla p + \nu\Delta\boldsymbol{u}\,, \tag{2.14}$$

again with body-force terms omitted because they will not influence the present developments. It is also to be noted that pressure is now the "kinematic" pressure, *i.e.*, pressure divided by density ρ; but we will make no specific notational distinction.

Conservation Form

The *conservation* (or divergence) *form* of the momentum equations can be written as

$$\boldsymbol{u}_t + \nabla\cdot(\boldsymbol{u}\boldsymbol{u}) = -\nabla p + \nu\Delta\boldsymbol{u}\,, \tag{2.15}$$

and has already been encountered in applying the Galerkin procedure to the N.–S. equations in Chap. 1. Since

$$\boldsymbol{u}\boldsymbol{u} = \begin{pmatrix} u \\ v \\ w \end{pmatrix} (u\ \ v\ \ w) = \begin{pmatrix} u^2 & uv & uw \\ vu & v^2 & vw \\ wu & wv & w^2 \end{pmatrix}, \tag{2.16}$$

it follows that

$$\nabla\cdot(\boldsymbol{u}\boldsymbol{u}) = \boldsymbol{u}\cdot\nabla\boldsymbol{u} + \boldsymbol{u}(\nabla\cdot\boldsymbol{u})\,.$$

But from the divergence-free constraint the second term on the right-hand side is zero, showing that (2.15) is equivalent to (2.14) in the analytical sense. As noted in [32], Eq. (2.14) (the standard form) provides the simplest form for discretization and is widely used; but such discretizations generally do not "conserve" either momentum or kinetic energy. On the other hand, straightforward discretizations of Eq. (2.15) do conserve momentum. We remark, however, that as has already been shown in the mathematical analyses of Chap. 1, the N.–S. equations are not conservation laws, so the importance, *per se*, of preserving conservation is not entirely obvious. Nevertheless, in the absence of the viscous terms they would be; so, especially for high-*Re* flows, it can possibly be advantageous for discretizations of the advective terms to be conservative.

Rotation Form

The *rotation form* of the incompressible N.–S. equations is expressed as

$$\boldsymbol{u}_t + \boldsymbol{\omega}\times\boldsymbol{u} = -\nabla P_T + \nu\Delta\boldsymbol{u}\,, \tag{2.17}$$

where P_T is the (kinematic) total pressure given by

$$P_T = p + \frac{1}{2}U^2\,, \qquad \text{with} \quad U^2 = u^2 + v^2 + w^2\,. \tag{2.18}$$

To see that this is equivalent to the standard form of the momentum equations we first note that

$$\begin{aligned} \boldsymbol{\omega}\times\boldsymbol{u} &= \big[w(u_z - w_x) - v(v_x - u_y)\big]\boldsymbol{e}_1 \\ &\quad + \big[u(v_x - u_y) - w(w_y - v_z)\big]\boldsymbol{e}_2 \\ &\quad + \big[v(w_y - v_z) - u(u_z - w_x)\big]\boldsymbol{e}_3\,. \end{aligned} \tag{2.19}$$

If we consider only the first component we see that

$$\begin{aligned} (\boldsymbol{\omega}\times\boldsymbol{u})_1 &= vu_y + wu_z - vv_x - ww_x \\ &= uu_x + vu_y + wu_z - \frac{1}{2}\frac{\partial}{\partial x}(u^2 + v^2 + w^2) \\ &= \boldsymbol{u}\cdot\nabla u - \frac{1}{2}\frac{\partial}{\partial x}\left(U^2\right)\,, \end{aligned}$$

where uu_x has been added and subtracted in the second line. We leave complete verification of Eqs. (2.17) and (2.18) as an exercise for the reader.

The rotation form is widely used, especially in the context of pseudo-spectral approximations to the N.–S. equations where it, or the skew-symmetric form described below, is necessary for stability (see, *e.g.*, Canuto *et al.* [28]). It is clear that somewhat more arithmetic is required to evaluate discretizations of Eq. (2.17) in comparison with that needed for corresponding evaluations of either the standard or conserved forms. On the other hand, discrete conservation of both momentum and kinetic energy is easily achieved with this formulation, so it could potentially be of value in simulating very high-*Re* flows.

Skew-Symmetric Form

The *skew-symmetric form* of the N.–S. equations is constructed as the average of the conserved and unconserved (standard) forms, *i.e.*, the average of Eqs. (2.14) and (2.15); the form of the momentum equations in this case is

$$\boldsymbol{u}_t + \frac{1}{2}\left[\boldsymbol{u}\cdot\nabla\boldsymbol{u} + \nabla\cdot(\boldsymbol{u}\boldsymbol{u})\right] = -\nabla p + \nu\Delta\boldsymbol{u}\,. \tag{2.20}$$

This form leads readily to discrete conservation of kinetic energy and, like the rotation form, enhances stability of pseudo-spectral methods. Furthermore, it is claimed to be more accurate than the rotation form in this context (see [32] and references cited therein).

The preceding formulations have all been specifically associated with treatment of the advective terms. In problems for which transport properties are nonconstant with respect to spatial coordinates, it is necessary to include a modification of the diffusion terms. In the present case we need consider only nonconstant viscosity, say $\nu = \nu(x, y, z)$, for example. In this situation it is necessary to express the Laplace operator of the various preceding forms as

$$\Delta \boldsymbol{u} = \nabla \cdot (\nu \nabla \boldsymbol{u}) \ .$$

We also observe that the divergence form appearing on the right-hand side must also be employed if the problem being considered is posed in a generalized coordinate system resulting in a variable coefficient gradient operator.

This concludes our discussions of the forms of the Navier–Stokes equations. We once again emphasize that the primitive-variable formulation is, by far, the most widely used in modern CFD, and from this point on in the present lectures we will consider only this form.

2.2 Pressure-velocity coupling

The pressure-velocity coupling problem is a difficulty that is quite specific to the incompressible N.–S. equations; it does not occur for numerical approximations of the compressible equations, and it seldom if ever occurs in the numerical analysis of other PDEs. It is a problem whose symptoms are easily recognized, as we will demonstrate in more detail below, but one which is not easily understood in the context of finite-difference/finite-volume approximations. On the other hand, it is well understood in the finite-element analysis of the incompressible equations. In a superficial way one might associate pressure-velocity "uncoupling" with the fact that pressure does not appear explicitly in the equation that must be used to set the pressure (the divergence-free condition), as we have already noted, and even more, the pressure itself does not appear in the momentum equations. In particular, only its gradient is present, implying that at best pressure might only be determined to within an additive function of time.

The correct treatment of the pressure-velocity coupling problem can be prescribed very precisely for finite-element methods, as hinted above. In the case of finite-difference and finite-volume methods on which we will focus attention in these lectures, much of what is known is based on physical arguments and trial-and-error approaches to algorithm development, although it is possible to draw some analogies with the finite-element analyses. As we will describe in the following subsections, the correct treatment requires specific combinations of grid structure and discrete approximation of the equations of motion. We will begin by describing various grid structures that have been employed, and we follow this with two specific discretizations of the N.–S. equations. We then briefly discuss a necessary modification to one of these.

2.2.1 Types of grid configurations

There are three main types of grid structures (and some modifications of these) that have been employed in attempts to numerically solve the N.–S. equations. They are: *i*) unstaggered, *ii*) staggered and *iii*) "partially staggered." These, along with their common modifications, are shown in Fig. 2.1. In each case we display a single 2-D grid cell and indicate the locations within such a cell where each of the dependent variables is to be calculated. We have chosen to use 2D here for clarity; extension to 3D is straightforward. We now briefly discuss the features and advantages/disadvantages of each of these alternatives.

Unstaggered, Natural Grid

From the viewpoint of basic numerical analysis of partial differential equations the unstaggered grid, shown in part (a) of the figure, with <u>all</u> variables defined at grid points in their natural locations at the

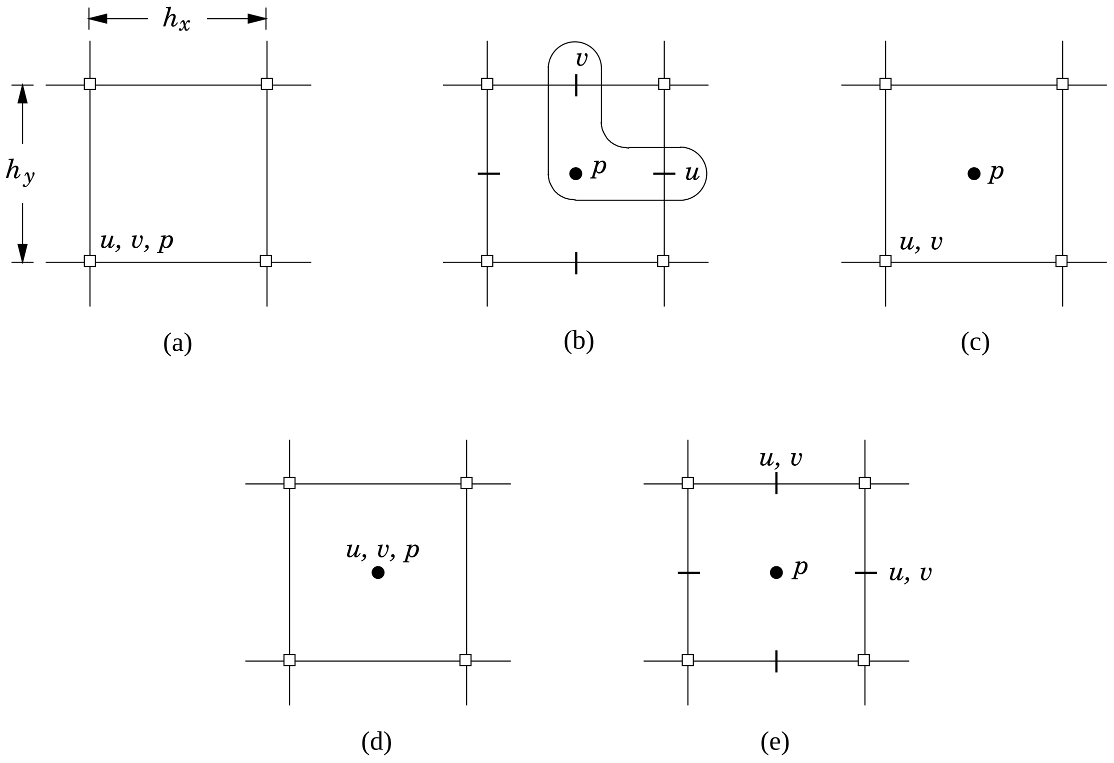

Figure 2.1: Alternative grid structures for solving the 2-D incompressible N.–S. equations; (a) natural unstaggered, (b) staggered, (c) partially staggered, (d) cell-centered unstaggered and (e) staggered w/ multiple momentum equations.

cell vertices would be the likely choice for discretization of any system of PDEs. With all discrete variables defined at each vertex, implementation of boundary conditions is straightforward. In particular, grid points will coincide with discrete boundary points, implying that Dirichlet conditions can be implemented without any approximation—a highly desirable situation. But in the context of the incompressible N.–S. equations there is a basic flaw with this grid configuration; namely, as we will see below, it is possible to discretely satisfy the divergence-free condition with velocity fields that are physically unrealistic, and in any given calculation there are essentially always some regions in the physical domain at which such velocities are produced. We will discuss this in more detail later.

Staggered Grid

The grid configuration presented in part (b) of the figure is termed "staggered" because the location at which each dependent variable is computed is different for all variables—the variables are staggered about the grid cell. The figure also displays a typical grouping of variables (clearly, there are other possibilities) needed for efficient indexing when coding computational algorithms. This type of grid was introduced by Harlow and Welch [36] in constructing the marker-and-cell (MAC) method to be described in Chap. 3, and it was the first configuration with which the N.–S. equations could be reliably solved in a primitive-variable formulation. In particular, the problem of pressure-velocity decoupling seen on the natural unstaggered grid does not occur. On the other hand, without the benefit of physical arguments such an arrangement of variables might seem counterintuitive, at best. But even worse is the fact that, as is easily seen, the no-slip condition cannot be exactly satisfied with this type of gridding. One might be inclined to simply shift dependent-variable locations down and to the left by a half grid spacing. This would completely remedy the problem with no-slip conditions, but it would at the same time introduce inability to exactly prescribe mass and momentum fluxes. These boundary condition difficulties at least in part motivated the search for other gridding arrangements.

Partially-Staggered (ALE) Grid

One such grid is the ALE (arbitrary Lagrangian-Eulerian) grid of Hirt *et al.* [37] which we loosely term "partially-staggered" because pressure is computed at cell centers as is the case for staggered grids, but both velocity components are calculated at the natural finite-difference grid points as in the unstaggered case. It is clear that such gridding provides a remedy to the problem of satisfying the no-slip and no-flux boundary conditions. But at any boundary point where pressure is to be specified, it is not possible to do this exactly (which is also true for the staggered grid). The major problems with this approach, however, are that the algorithm needed to implement the ALE method is quite complicated, and pressure-velocity decoupling can still occur, albeit, usually with less severity than is true for unstaggered grids. As a consequence of these difficulties ALE methods are seldom used, and we will provide no further consideration of them in these lectures.

Cell-Centered Unstaggered Grids

The cell-centered "unstaggered" grid (sometimes, and more appropriately, termed a co-located grid) has experienced much popularity following appearance of the paper by Rhie and Chow [38] in 1983. It is often simply called an unstaggered grid, but as can be seen by comparing parts (a) and (d) of Fig. 2.1, this is not the same configuration as the usual unstaggered grid. While all solution components are computed at the same location (hence, the terminology "co-located"), this location corresponds to the cell center rather than to a natural grid point at a cell vertex. Although one might imagine shifting all the cell centers to the lower left-hand vertex of each cell to formally obtain the structure of the natural unstaggered grid, this would not be consistent with details of constructing difference approximations as done in [38]. Indeed, the approximations employed are quite similar to those typically used for the compressible flow equations (see, *e.g.*, Hirsch [39]). Namely, all dependent variables are computed at cell centers, and the fluxes needed for these calculations are obtained at the cell walls. But for reasons to be discussed in more detail below, it is necessary to also alter the pressure Poisson equation for this approach to be successful in the incompressible case.

We also note that the original work of Rhie and Chow [38] was done in the context of 2-D steady-state calculations using a solution procedure that is basically a SIMPLE algorithm (see Patankar [40]) to be discussed in detail in Chap. 3 of these lectures. Zang *et al.* [41] later extended this unstaggered formulation to 3-D time-dependent problems employing a projection method (see Chap. 3) as the basic solution technique.

It is important to recognize that independent of whether the implementation is for steady or unsteady problems, co-located, cell-centered variables lead to inability to exactly satisfy all boundary conditions. Hence, results are generally less accurate than those computed on a staggered grid. Nevertheless, this form of gridding is widely used in present-day commercial CFD software.

Staggered Grids with Multiple Momentum Equations

The final combination of gridding and dependent variable placement we consider here consists of the usual staggered grid shown in Fig. 2.1(b) but with all velocity components computed on each cell wall, as indicated in Fig. 2.1(e). This approach is motivated by concerns regarding accuracy of interpolations of velocity components to locations where they have not been computed from the corresponding momentum equation(s) during construction of advective terms of the momentum equation associated with the particular cell wall location. (But note that very similar interpolations must be performed in the cell-centered unstaggered approaches in order to construct the required wall fluxes.) Moreover, computing all velocity components on all cell walls leads to exact satisfaction of no-slip and no-flux boundary conditions, which is not possible with the usual staggered gridding as we earlier noted. But the multiple-momentum equation approach leads to approximately twice the arithmetic required of the usual staggered method for momentum equation solutions in 2D, and three times the amount in 3D. As will be shown in more detail later, the averaging employed for the usual staggered-grid technique does not result in loss of formal accuracy, and there is thus little reason to accept the extra arithmetic required in the multiple-momentum equation methods unless accuracy of boundary conditions is a particularly serious issue in a specific problem.

We will not give further consideration to this approach in these lectures except to note that use of staggered grids has sometimes been criticized specifically because of the perceived extra required arithmetic (see Zang *et al.* [41]). But this involves a comparison between unstaggered methods and the multiple-momentum equation versions of the staggered-grid approach, rather than with the more widely-used one employing only a single momentum equation on each cell wall. Thus, such criticisms are inaccurate and misrepresent the actual situation. In fact, the arithmetic required for the typical staggered-grid formulation is actually slightly less than that required for co-located grid formulations due to the additional arithmetic required for the modified Poisson equation solves of the latter. In the sequel we will consider only the form of staggered grid depicted in Fig. 2.1(b).

2.2.2 Finite-difference/finite-volume discretization of the N.–S. equations

In this section we will treat the basic finite-difference spatial discretization of N.–S. equations. We will leave details of time integration procedures to Chap. 3 where we will consider specific solution algorithms, and here consider only the simple backward Euler integration scheme. In the present subsection, we will employ only centered-difference approximations, and in the next section of this chapter we will study some modifications to this that are sometimes used for the advective terms. We will consider three main cases: *i*) the natural, unstaggered grid, *ii*) the staggered grid and *iii*) the cell-centered (co-located) unstaggered grid. We will consider each of these in the context of a straightforward model problem, the so-called “lid-driven cavity” problem, depicted in Fig. 2.2 along with qualitative solution features.

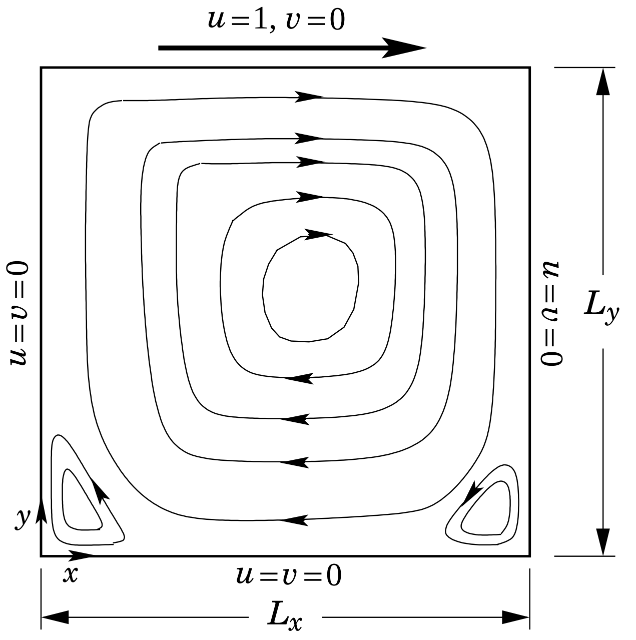

Figure 2.2: Lid-driven cavity geometry, boundary conditions and streamlines for the viscous, incompressible N.–S. equations.

Lid-Driven Cavity Problem Statement

We begin by providing a complete statement of the mathematically well-posed problem. We define the domain $\Omega \equiv (0, L_x)\times(0, L_y)$ and let $L_x = L_y = 1$. Then we solve

$$\boldsymbol{u}_t + \boldsymbol{u}\cdot\nabla\boldsymbol{u} = -\nabla p + \nu\Delta\boldsymbol{u}\,, \qquad (x,y)\in\Omega\,, \tag{2.21}$$

with initial data

$$\boldsymbol{u} \equiv 0 \quad \text{in } \overline{\Omega}\setminus\{(x,1)\}\,;$$

that is,

$$u = 1 \quad \text{for } y = 1\,.$$

The boundary conditions are as shown in Fig. 2.2 and correspond to no-slip and no mass flow on and through all solid boundaries with the top boundary moving fromleft to right at a uniform, constant speed of unity.

To obtain an equation explicitly in terms of pressure, we use the PPE given in Eq. (2.3) in place of the usual continuity equation. Thus, in the present case we have

$$\Delta p = -\nabla\cdot(\boldsymbol{u}\cdot\nabla\boldsymbol{u}) \tag{2.22}$$

with the boundary conditions

$$\frac{\partial p}{\partial\boldsymbol{n}} = 0 \quad \text{on } \partial\Omega\,.$$

It is known that if ν is not too small, this problem possesses a steady solution. We shall assume that is a possibility, but we will nevertheless construct a solution procedure capable of computing time-dependent solutions as well.

Before starting discretization of this problem it is important to notice that the PDE boundary value problem for pressure is of Neumann type, and as such does not possess a unique solution. Indeed, there does not exist any solution for Poisson/Neumann problems in general unless

$$\int_\Omega f\,d\boldsymbol{x} = \int_{\partial\Omega} g\,d\ell\,,$$

where f is the right-hand side forcing of the Poisson equation (2.22), and g is the right-hand side of the Neumann boundary condition. This follows directly from the divergence theorem (demonstration of which we leave as an exercise to the reader), and in the present case leads to the solvability condition

$$\int_\Omega u_y v_x - u_x v_y\,d\boldsymbol{x} = 0$$

via direct calculation that invokes the divergence-free requirement. We emphasize that this solvability condition is not guaranteed to hold, *a priori* , and we will later see how to construct Neumann conditions that guarantee its satisfaction.

Discretization of the N.–S. Equations on an Unstaggered Grid

We begin this treatment with discretization of the momentum equations for which we use the formal notation

$$\begin{aligned} D_{0,x} &\equiv \frac{u_{i+1,j} - u_{i-1,j}}{2h} \\ D^2_{0,x} &\equiv \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}\,, \end{aligned}$$

with analogous definitions for the y-direction centered differences $D_{0,y}$ and $D_{0,y}^2$. Here, h is the (uniform) grid spacing. We follow this with a similar discretization for the PPE, and we then describe the nature of the computed results arising from this treatment, in particular, discussing what has gone wrong with this proposed formulation.

Momentum Equation Treatment. For the grid covering the problem domain displayed in Fig. 2.3, we see that $h_x = h_y = h$, and we can apply very standard, straightforward difference techniques to approximate Eqs. (2.21). We first express these in component form as

$$u_t + uu_x + vu_y = -p_x + \nu\Delta u\,, \tag{2.23a}$$

$$v_t + uv_x + vv_y = -p_y + \nu\Delta v\,. \tag{2.23b}$$

These are (coupled) quasilinear, parabolic PDEs, and if we use backward Euler time integration with time

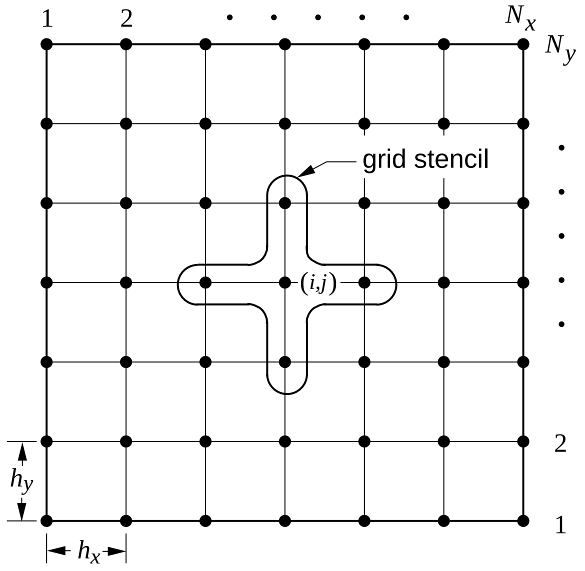

Figure 2.3: Basic unstaggered gridding of problem domain.

step k $(= \Delta t)$ we can express these as

$$\left\{I + k\left[u_{i,j}^{(m)}D_{0,x} + v_{i,j}^{(m)}D_{0,y} - \nu\left(D_{0,x}^2 + D_{0,y}^2\right)\right]\right\}u_{i,j}^{(m+1)} = u_{i,j}^n - D_{0,x}\,p_{i,j}^{(m)}\,, \tag{2.24a}$$

$$\left\{I + k\left[u_{i,j}^{(m)}D_{0,x} + v_{i,j}^{(m)}D_{0,y} - \nu\left(D_{0,x}^2 + D_{0,y}^2\right)\right]\right\}v_{i,j}^{(m+1)} = v_{i,j}^n - D_{0,y}\,p_{i,j}^{(m)}\,. \tag{2.24b}$$

Here, parenthesized superscripts denote iteration counters associated with any chosen nonlinear iterative procedure, and such that, *e.g.*, $u^{(m)} \to u^{n+1}$ as $m \to \infty$. The grid stencil for this discretization shown in Fig. 2.3 is the standard one. It is clear from this, and the fact that only Dirichlet conditions are being imposed on Eqs. (2.23), that discrete boundary condition implementation is trivial, and we will not give further consideration to this here. (The reader unfamiliar with the necessary procedure is referred to McDonough [42].)

It is also easily seen from the grid stencil that the coefficient matrices arising from collection of the left-hand sides of the individual Eqs. (2.24) must be of the form of 2-D discrete Laplacians, that is, noncompactly banded five-band matrices. (We remark that if the discrete system is left in fully-coupled form, the single matrix will be a 2×2 block coupled five-band one. But this is seldom done when solving the incompressible equations.) This implies that in the context of generating time-accurate solutions, the most efficient approach should be some form of time splitting. It is assumed readers are familiar with such techniques; but for those who may not be, any of the following references may be consulted: Mitchell and Griffiths [20], Douglas and Gunn [43], or McDonough [44].

Here, we will employ the Douglas and Gunn procedure [43] which is equivalent to the Douglas and Rachford scheme (see [20]) in the present case, due to our use of backward Euler time integration. It is easily seen that once $u_{i,j}^{(m)}$ and $v_{i,j}^{(m)}$ have been specified Eqs. (2.24) are linear, and the systems arising from the collection of these can be expressed as, *e.g.*, for x-momentum,

$$(\boldsymbol{I} + k\boldsymbol{A})u^{n+1} + \boldsymbol{B}u^n = \boldsymbol{s}^n \,, \tag{2.25}$$

where the n time level superscript on the right-hand side is formal and merely denotes <u>known</u> quantities, independent of their specific time level—even an advanced one. Also, un-subscripted quantities denote vectors or matrices, the latter if the notation is bold face.

It is clear from the form of Eqs. (2.24) that in either of these the matrix $\boldsymbol{A}$ can be decomposed as

$$\boldsymbol{A} = \boldsymbol{A}_x + \boldsymbol{A}_y \,, \tag{2.26}$$

where each term on the right-hand side is a $N \times N$ matrix ($N = N_x N_y$) containing the coefficients arising from discretizations of the form indicated in (2.24). In particular, each of $\boldsymbol{A}_x$ and $\boldsymbol{A}_y$ is a tridiagonal matrix. As already noted, we will employ the Douglas and Gunn formalism and, further, use the δ form of this; details can be found in [43] and [44]. Here we mainly present the results and leave details of obtaining them as an exercise for the reader.

If for the x-momentum equation we define

$$\delta u^{(\ell)} \equiv u^{(\ell)} - u^n \,, \qquad \ell = 1, 2 \,, \tag{2.27}$$

then we can compute the next iterate $u^{(m+1)}$ of u during calculations at time level $n+1$ via the following sequence of steps:

$$\left\{ I + k\left[u_{i,j}^{(m)} D_{0,x} - \nu D_{0,x}^2 \right] \right\} \delta u_{i,j}^{(1)} = -D_{0,x} p_{i,j}^{(m)} - k\left[u_{i,j}^{(m)} D_{0,x} + v_{i,j}^{(m)} D_{0,y} - \nu\left(D_{0,x}^2 + D_{0,y}^2 \right) \right] u_{i,j}^n \,, \tag{2.28a}$$

$$\left\{ I + k\left[v_{i,j}^{(m)} D_{0,y} - \nu D_{0,y}^2 \right] \right\} \delta u_{i,j}^{(2)} = \delta u_{i,j}^{(1)} \,, \tag{2.28b}$$

together with rearrangement of (2.27) and using $u^{(2)} = u^{(m+1)}$ to obtain

$$u_{i,j}^{(m+1)} = u_{i,j}^n + \delta u_{i,j}^{(2)} \,, \qquad \forall \;\; (i,j) = (1,\ldots,N_x; 1,\ldots,N_y) \,. \tag{2.29}$$

Clearly an analogous sequence of steps can be used to compute $v^{(m+1)}$. We remark that $u^{(m+1)}$ would probably be used for all entries of $u^{(m)}$ in the corresponding y-momentum equation as this would produce a block Gauss–Seidel iteration, compared with the block Jacobi scheme arising from use of $u^{(m)}$.

It should be noted that quasilinearization (Newton–Kantorovich) iteration, δ form or otherwise, might be used in place of the simple Picard iteration we have described here (see [44] for details). In general, this would be preferred, and we will present this later in the context of a different method.

Pressure Poisson Equation Treatment. We next provide discretization of the PPE; this is straightforward and uses the same mesh star shown for the momentum equations in Fig. 2.3. Boundary condition implementation requires discretization of Eq. (2.22) <u>on the boundaries</u> of Ω and use of the Neumann boundary condition to eliminate image points.

To begin, write the PPE as

$$\Delta p = -2(u_y v_x - u_x v_y) \,, \tag{2.30}$$

and consider a point on the bottom boundary of Fig. 2.3, say $(i,1)$, for definiteness. At this point, the left-hand side of Eq. (2.30) is approximated as

$$\left(D_{0,x}^2 + D_{0,y}^2 \right) p_{i,1} = \frac{1}{h^2}\left(p_{i-1,1} - 2p_{i,1} + p_{i+1,1} + p_{i,0} - 2p_{i,1} + p_{i,2} \right) \,, \tag{2.31}$$

which contains a value of p at the image point $(i,0)$. On this boundary the homogeneous normal derivative boundary condition is

$$\frac{\partial p}{\partial \boldsymbol{n}} = \frac{\partial p}{\partial y} = 0\,,$$

which upon discretization with a centered difference implies that $p_{i,0} = p_{i,2}$. Then substitution of this into the right-hand side of (2.31) yields

$$\frac{1}{h^2}\left(p_{i-1,1} - 4p_{i,1} + p_{i+1,1} + 2p_{i,2}\right)\,,$$

thus eliminating the image point.

Similar results can be obtained for the remaining three boundaries, including corner points. We leave demonstration of this as an exercise for the reader. We note at this point that the system matrix arising from this discretization is nearly singular—theoretically, it is singular; but in machine arithmetic it misses being singular by the level of representation and rounding errors. Iterative solution methods perform very poorly in this nearly-singular context.

We now consider treatment of the right-hand side of Eq. (2.30) on the boundaries of Ω. It should be clear that for the present problem, we have $u_x = v_x = 0$ at all horizontal boundary points, and similarly $u_y = v_y = 0$ at vertical boundary points. At the same time, the right-hand side of Eq. (2.30) contains one x derivative and one y derivative in each term; hence, it is zero around the entire boundary. So no further attention need be given to the boundary treatment for this equation, and the basic problem discretization is complete.

There is a final detail to consider before presenting a pseudo-language algorithm for solving the current problem. Recall that, as already noted, solutions to Neumann problems are not unique. Two main approaches have been used to address this numerically. The first, and less often used, is to set $\langle p, 1\rangle = 0$. That is, at the end of the iterations associated with solving the PPE, we average the result and subtract the average value from the solution. The main difficulty with this approach is that it does nothing to improve the performance of the iterative technique being used to solve the PPE. It is a satisfactory analytical approach that is of little value in practical computational procedures.

The second alternative entails assigning a value of pressure at one boundary point. Formally, this is equivalent to adding a constant value at all points, but since we do not yet know the pressure at any point it does not matter whether we actually make the assignment anywhere except at the chosen boundary point. Once this has been done, the PPE problem becomes a mixed Dirichlet–Neumann one, possessing a unique solution and no longer exhibiting a singular coefficient matrix, although the condition number may still be quite large. Thus, numerical solution procedures perform somewhat better. Moreover, adding a constant in this way is of no other consequence since solutions to the original Neumann problem are unique only to within an additive function of time in any case.

This completes the basic treatment required to formulate a pseudo-language algorithm; we present this as follows.

Algorithm 2.1 *(Basic unstaggered-grid treatment of N.–S. equations) Suppose n time steps (or pseudo-time steps, if a steady solution is being sought) have been completed. To advance the solution to time level $n+1$, perform the following steps.*

```
1. Do $m_1 = 1, maxpitr$
     Do $m_2 = 1, maxitr$
       solve discrete x-momentum equations (2.28) with fixed $p^{(m_1)}$
       solve discrete y-momentum equation analogous to (2.28) with fixed $p^{(m_1)}$
       test convergence of velocity iterations:
         If $\max\left(\left\|u^{(m_2+1)} - u^{(m_2)}\right\|, \left\|v^{(m_2+1)} - v^{(m_2)}\right\|\right) < \epsilon$, then go to 2
     Repeat $m_2$
     Write error message
```

```
2. Solve PPE
     set $p_{N_x/2,1} = 0$
     iteratively solve discretization of Eq. (2.30)
     test convergence of pressure iterations:
       If $\left(\left\|p^{(m_1+1)} - p^{(m_1)}\right\|\right) < \epsilon$, then print results
                                                   stop
                                              else Repeat $m_1$

3. Write error message
```

A few remarks are in order regarding details of the algorithm, *per se*, aside from those concerning the problem formulation. First, the choice of solving the PPE in the outer loop is arbitrary, but precisely where this is done has little effect on the ultimate outcome. On the other hand, from a pure mathematics standpoint, it leads to difficulty in solving the momentum equations at the first time step since we are not permitted to prescribe initial pressure independent of the velocity field. A simple remedy is to "create" initial data for pressure using initial velocity data to evaluate the right-hand side of Eq. (2.20). But this implies that the velocity field should be divergence free. An obvious alternative is to interchange steps 1 and 2.

Second, the value of p chosen for the boundary assignment needed to produce a Dirichlet condition is completely arbitrary in the present problem, as should be clear from replacing p with $p+C$ for any constant C in the PPE and Neumann conditions. Finally, the location on the grid at which this assignment is made is also arbitrary in principle, but not completely so in practice. In particular, for centered approximations of the Laplacian, as employed herein, it is necessary to avoid the corners of a domain represented in Cartesian coordinates because Dirichlet conditions at such points are never included in the discrete equations. It is common to make the Dirichlet condition assignment at a the midpoint of one of the sides of the domain, as done in the algorithm presented above.

Shortcomings of Basic Approach

As intuitively appealing as this algorithm might seem, it embodies many fundamental flaws rendering it incapable of producing correct numerical solutions to the N.–S. equations. For the most part these are related to the PPE and/or the divergence-free condition. We first list these, and then provide some further details.

i) The PPE is constructed under the assumption of satisfaction of the divergence-free constraint, but there is no mechanism within the algorithm to guarantee this.

ii) Boundary conditions for the PPE are not generally correct—neither physically, nor mathematically.

iii) Centered finite-difference approximations on unstaggered grids permit discrete satisfaction of the divergence-free constraint by non-physical velocity fields.

iv) Similarly, centered approximations to the pressure gradient terms (on unstaggered grids) in the momentum equations allow non-physical pressure fields to go "undetected"—and thus, uncorrected.

v) There exists a basic theorem from finite-element analysis of the N.–S. equations showing that use of centered approximations of both velocity and pressure, together, on an unstaggered grid will not converge to a physically correct divergence-free solution.

We now present further details associated with each of these.

Failure to Explicitly Satisfy $\nabla\cdot\boldsymbol{u} = 0$. With regard to the first item we note that although the divergence-free constraint has been employed to simplify the PPE, it (the constraint equation) has not been "solved." Moreover, solutions to the momentum equations are not necessarily divergence free. (It they were, we would not need to separately impose the divergence-free constraint in the first place.) It is worth noting, however, that if initial data are divergence free, pressure is calculated correctly, and truncation and rounding errors are negligible, then time evolution of such initial data preserves this property. It is well known that this can be accomplished for short times using spectral methods and high-precision machine arithmetic, but eventually the divergence-free property is lost.

Incorrect PPE Boundary Conditions. We next consider the boundary conditions required to solve the PPE, Eq. (2.30). It is often argued that at least in flow situations not too different from unseparated boundary layers the condition

$$\frac{\partial p}{\partial \boldsymbol{n}} = 0$$

might be adequate. (Indeed, it is often viewed as an "assumption"—but it is actually an outcome—in boundary-layer theory.) Here, we will show more precisely under what circumstances this is true, and that it cannot be used in general. In the course of doing this we provide the correct boundary conditions.

To begin, we recall that to guarantee existence of solutions to Poisson/Neumann problems of the form

$$\Delta p = f \qquad \text{in } \overline{\Omega} \tag{2.32}$$

with boundary conditions

$$\frac{\partial p}{\partial \boldsymbol{n}} = g \qquad \text{on } \partial\Omega\,, \tag{2.33}$$

the following *consistency condition* (or solvability condition), derived from the divergence theorem, must be satisfied (see, *e.g.*, [27]):

$$\int_\Omega f\,d\boldsymbol{x} = \int_{\partial\Omega} g\,d\ell\,, \tag{2.34}$$

as noted earlier.

Now recall that in constructing the PPE we started by taking the divergence of the momentum equations, and then simplified the result using the divergence-free condition. We begin in the same way here, but it is not useful to apply the divergence-free condition in this case because this arises as a vector operator, and here we will want to identify scalar boundary conditions. Thus, we write

$$-\nabla\cdot\nabla p = \nabla\cdot(\boldsymbol{u}_t + \boldsymbol{u}\cdot\nabla\boldsymbol{u} - \nu\Delta\boldsymbol{u}) \equiv f\,,$$

but we do not carry out details of constructing the divergence. Instead, we nowapply the divergence theorem to see that

$$\int_\Omega \nabla\cdot\nabla p\,d\boldsymbol{x} = \int_{\partial\Omega} \boldsymbol{n}\cdot\nabla p\,d\ell\,,$$

and at the same time we have (by using the N.–S. equations on the right-hand side)

$$\begin{aligned}\int_\Omega \nabla\cdot\nabla p\,d\boldsymbol{x} &= -\int_\Omega \nabla\cdot(\boldsymbol{u}_t + \boldsymbol{u}\cdot\nabla\boldsymbol{u} - \nu\Delta\boldsymbol{u})\,d\boldsymbol{x} = \int_\Omega f\,d\boldsymbol{x}\\ &= -\int_{\partial\Omega} \boldsymbol{n}\cdot(\boldsymbol{u}_t + \boldsymbol{u}\cdot\nabla\boldsymbol{u} - \nu\Delta\boldsymbol{u})\,d\ell\left(\equiv \int_{\partial\Omega} g\,d\ell\right)\,,\end{aligned}$$

with the divergence theorem used to obtain the equality in the second line. Thus, we see that the consistency requirement is automatically satisfied if on vertical boundaries we set

$$\frac{\partial p}{\partial x} = -\left(u_t + uu_x + vu_y - \nu\Delta u\right)\,, \tag{2.35}$$

and on horizontal boundaries we use

$$\frac{\partial p}{\partial y} = -\left(v_t + uv_x + vv_y - \nu\Delta v\right)\,, \tag{2.36}$$

as proposed by Gresho and Sani [30]. In this way we have prescribed functions g that automatically satisfy (2.34).

It is of interest at this point to see under what conditions the above essentially collapse to the homogeneous Neumann conditions that are often used. First, it is clear that the boundary values of $\boldsymbol{u}$ must be independent of time. Furthermore, if u and v are both zero on all boundaries, then the only remaining terms are $\nu\Delta u$ on vertical boundaries and $\nu\Delta v$ on horizontal ones. (We remark that such a combination of velocity boundary conditions will require that body forces be acting to drive the flow unless there is the additional complication of inflow and outflow.) In any case, under these assumed conditions we see that as $\nu \to 0$ (or, equivalently, as $Re \to \infty$), the above general conditions collapse to

$$\frac{\partial p}{\partial x} \simeq 0\,, \qquad \text{and} \qquad \frac{\partial p}{\partial y} \simeq 0\,.$$

Effects of Centered Differencing of Continuity Equation. We next consider the effects of centered differencing with respect to satisfaction of the continuity equation. In particular, we will show that completely non-physical velocity fields are able to satisfy the discrete divergence-free condition obtained by centered differencing. We begin by considering the simple 1-D case for which the divergence-free condition collapses to $u_x = 0$. This, of course, implies $u \equiv \text{const}$. But in Fig. 2.4 we display a <u>non-constant</u> grid function $\{u_i\}$ that satisfies $D_{0,x}u_i = 0$.

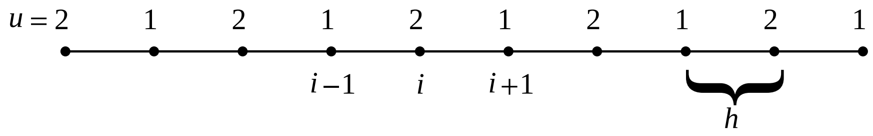

Figure 2.4: Non-constant grid function satisfying discrete 1-D continuity equation.

The same situation occurs also in 2D and 3D. We illustrate this here for the former case, and leave a similar construction in 3D for the reader. Now the discrete divergence-free constraint is

$$D_{0,x}u_{i,j} + D_{0,y}v_{i,j} = 0\,,$$

or with $h_x = h_y = h$,

$$u_{i+1,j} - u_{i-1,j} + v_{i,j+1} - v_{i,j-1} = 0\,. \tag{2.37}$$

Figure 2.5 presents a velocity field that exactly satisfies Eq. (2.37), but which makes no physical sense. The reader can easily check that Eq. (2.37) is satisfied at all interior grid points. Clearly, somewhat unrealistic boundary conditions would be needed to produce this uniform distribution of values, and in practice the effect is not so perfectly obtained. Nevertheless, it is observed in significant regions of a domain for actual calculations carried out with centered-difference approximations on unstaggered and partially-staggered grids.

For reasons that should be obvious from the figure, this phenomenon is termed "checkerboarding." It is interesting to note that in the extreme case depicted in Fig. 2.5 the right-hand side of the usual discrete PPE would be zero. Together with homogeneous Neumann boundary conditions this would result in a constant pressure field, so its use in the discrete momentum equations would have no effect on the computed velocity field. We see from this that once it occurs, the checkerboard solution is very stable and cannot easily be corrected.

It is clear from Fig. 2.5 that if we were to employ first-order forward differencing to approximate the continuity equation, the checkerboard solution shown here would not be discretely divergence free. On the other hand, if we were to use a combination of forward and backward differences such as

$$D_{+,x}u_{i,j} + D_{-,y}v_{i,j} = 0\,,$$

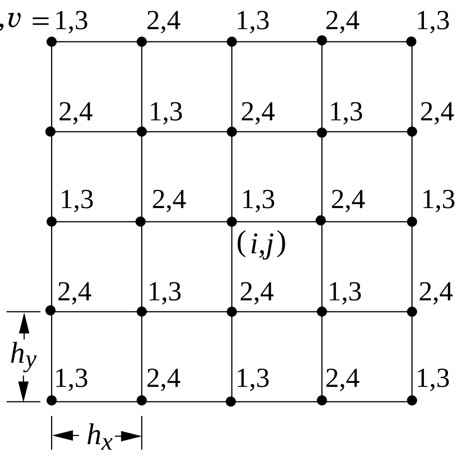

Figure 2.5: Non-physical grid function satisfying discrete 2-D continuity equation.

we would recover satisfaction of the discrete divergence-free constraint. This suggests that use of low-order, one-sided difference approximations, as has sometimes been proposed, will not necessarily correct the problem.

Centered Approximation of Pressure Gradient in Momentum Equations. It can also be seen that centered approximations to the pressure-gradient terms of the momentum equations are unable to correct a checkerboard phenomenon. For example, consider the pressure field exhibited in Fig. 2.6. Clearly, once

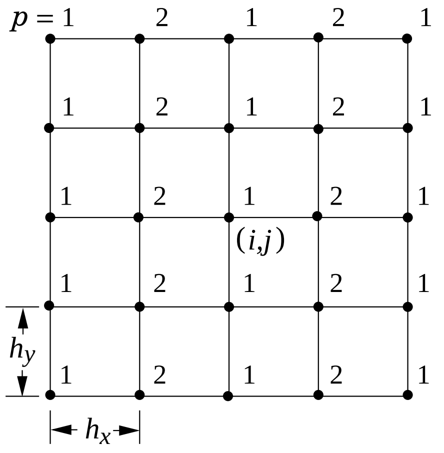

Figure 2.6: Non-physical checkerboard pressure distribution.

this pressure field exists, it can no longer act to modify the velocity field that may have been responsible for creating it in the first place because the centered discretizations representing the pressure gradient, $D_{0,x}p_{i,j}$ and $D_{0,y}p_{i,j}$, produce identically zero results for both components, and the momentum equations can no longer effect changes to this velocity distribution.

The Div-Stability Condition. The last main issue with the current algorithm to be considered here is in many ways the most important, and at the same time the least well known and understood. In finite-element analysis of the Navier–Stokes equations there is a condition known as the *Ladyzhenskaya–Babuska–Brezzi (LBB) condition*, more recently termed the *div-stability* condition (see Gunzburger [45] and references therein), whose satisfaction guarantees that associated finite-element spaces for velocity and pressure are such that divergence-free finite-element approximations converge to divergence-free exact solutions as the element size approaches zero. Details of this condition are technically difficult; moreover, the condition is not easily checked for general practical situations. As a consequence we will not present the mathematical representation of the condition, itself, but rather list some well-known and important consequences. These are as follows:

i) the combination of centered finite-difference approximation and unstaggered gridding does not satisfy the div-stability condition;

ii) in general, for finite-element methods it is necessary either to use different degree polynomial approximations for velocity and pressure (so-called *mixed* finite elements) on the same mesh, or to use the same degree and different meshes (staggered grid) in order to satisfy the div-stability condition;

iii) in the finite-element context, satisfaction of the div-stability requirement can be circumvented via modifications to the discrete continuity equation.

We note that *ii*) can be taken to imply that staggered grids should be used in the finite-difference context if the same order of accuracy is to be used in approximating both velocity and pressure. In addition, within the confines of finite-difference approximations *iii*) implies that modifications to the PPE (or other equations for pressure—see the artificial compressibilty method in Chap. 3) might be made to permit calculations on unstaggered grids, or from *ii*), a lower order of accuracy might be employed for pressure calculations.

Finite-Volume Discretization of the N.–S. Equations on Staggered Grids

In this subsection we present a detailed analysis of finite-volume discretization of the N.–S. equations on a staggered grid. We note that, in principle, this is equivalent to a finite-difference approximation; indeed, the final results will be identical. But the finite-volume derivation is useful in its own right, and in addition it actually provides the motivation for specific point location, and dependent variable location, assignments on the staggered grid. Hence, we will emphasize it here. We begin with discretization of the continuity equation which requires dependent variable assignments at grid locations different from those of the natural finite-difference grid points; so this permits us to set up the staggered-grid formalism in the simplest possible situation. We follow this with a corresponding treatment of the x-momentum equation and an abbreviated analysis of the y-momentum equation, and we conclude by briefly describing boundary condition implementations.

Continuity Equation. Consider the continuity equation (divergence-free constraint) in 2D,

$$\nabla \cdot \boldsymbol{u} = u_x + v_y = 0\,, \tag{2.38}$$

on a single grid cell, denoted $\Omega_{i,j}$ and displayed in Fig. 2.7. As the figure shows, this cell is defined as the region bounded by the $i-1$ and i grid lines in the x-direction, and by the $j-1$ and j lines in the y-direction. These are separated by the grid spacings h_x and h_y, respectively. On each such cell we must have

$$\int_{\Omega_{i,j}} \nabla \cdot \boldsymbol{u}\; dV = 0$$

(since Eq. (2.38) holds at every point in the continuum), and from Gauss's theorem we obtain

$$\int_{\partial\Omega_{i,j}} \boldsymbol{u} \cdot \boldsymbol{n}\; dA = 0\,,$$

or

$$\int_{\partial\Omega_{i,j}} u\,n_x + v\,n_y\; dA = 0\,. \tag{2.39}$$

Here, the differential elements dV and dA correspond to the 3-D case, but we will retain this notation here in 2D where they actually imply area and arclength, respectively.

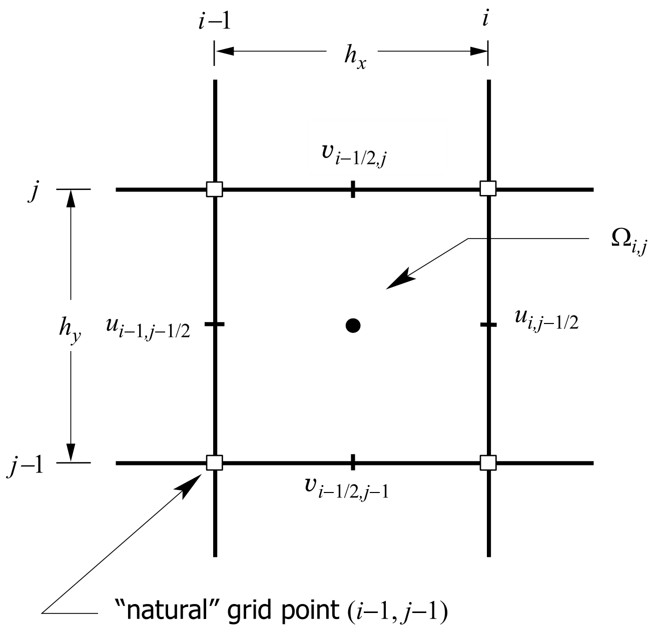

Figure 2.7: Staggered grid cell, $\Omega_{i,j}$.

If we let $\widetilde{u}$ denote the average velocity through a vertical segment of $\partial\Omega_{i,j}$ and let $\overline{v}$ be the average through horizontal segments we can evaluate (2.39) as

$$(\widetilde{u}_i - \widetilde{u}_{i-1})h_y + (\overline{v}_j - \overline{v}_{j-1})h_x = 0\,, \tag{2.40}$$

with single indices denoting the associated face of $\Omega_{i,j}$. But it is convenient to view these averages as simply being values at the center of the corresponding cell faces (which they are as the discretization step sizes approach zero). So, by using the notation of Fig. 2.7, we can write Eq. (2.40) as

$$\frac{1}{h_x}\left(u_{i,j-1/2} - u_{i-1,j-1/2}\right) + \frac{1}{h_y}\left(v_{i-1/2,j} - v_{i-1/2,j-1}\right) = 0\,. \tag{2.41}$$

It should be noted that this is a centered approximation to the divergence-free constraint, but one using only a single grid cell rather than the usual four cells that would be needed in the context of unstaggered gridding. We leave demonstration of second-order accuracy of this approximation as an exercise for the reader.

Now with this representation of the continuity equation it is natural to locate the associated pressure at the cell center, as indicated in Fig. 2.7. In particular, recall that pressure must be determined so as to satisfy the divergence-free constraint expressed here as Eq. (2.41), and the grid location $(i-1/2, j-1/2)$ is closest to (in fact, equidistant from, if $h_x = h_y$) all of the grid locations used in (2.41). Indeed, it is the location at which the divergence is being calculated, as is clear from Eq. (2.41); thus, it should be the preferred location for p.

Momentum Equations. Having established a rationale for grid function locations within a grid cell, we can now proceed to the more complicated derivation of the finite-volume form of the momentum equations. We will carry this out in detail for the x-momentum equation and merely outline steps and give the result for y-momentum, leaving the details of obtaining this as an exercise for the reader.

We start with the vector form of the equations (in the absence of body forces),

$$\frac{D\boldsymbol{u}}{Dt} = -\nabla p + \nu\nabla\cdot\nabla\boldsymbol{u}\,,$$

which we integrate over the grid cell $\Omega_{i,j}$:

$$\int_{\Omega_{i,j}} \frac{D\boldsymbol{u}}{Dt}\, dV = \int_{\Omega_{i,j}} -\nabla p + \nu \nabla \cdot \nabla \boldsymbol{u}\, dV \,.$$

Application of the divergence theorem to the right-hand side results in

$$\int_{\Omega_{i,j}} \frac{D\boldsymbol{u}}{Dt}\, dV = \int_{\partial\Omega_{i,j}} -p\boldsymbol{n} + \nu \nabla \boldsymbol{u} \cdot \boldsymbol{n}\, dA \,,$$

and we write this in component form as

$$\int_{\Omega_{i,j}} \frac{D}{Dt} \begin{pmatrix} u \\ v \end{pmatrix} dV = \int_{\partial\Omega_{i,j}} - \begin{pmatrix} pn_x \\ pn_y \end{pmatrix} + \nu \begin{pmatrix} u_x n_x + u_y n_y \\ v_x n_x + v_y n_y \end{pmatrix} dA \,. \tag{2.42}$$

We now consider the x-component of this,

$$\int_{\Omega_{i,j}} \frac{\partial u}{\partial t} + \boldsymbol{u} \cdot \nabla u\, dV = \int_{\partial\Omega_{i,j}} -pn_x + \nu(u_x n_x + u_y n_y)\, dA \,, \tag{2.43}$$

in detail. By assuming the problem geometry and grid configuration are independent of time we can commute differentiation and integration in the first term on the left-hand side, and after applying Gauss's theorem to the second term on the left, we obtain

$$\frac{\partial}{\partial t} \int_{\Omega_{i,j}} u\, dV + \int_{\partial\Omega_{i,j}} u^2 n_x + vun_y\, dA = \int_{\partial\Omega_{i,j}} -pn_x + \nu(u_x n_x + u_y n_y)\, dA \,. \tag{2.44}$$

Here, we have used $\boldsymbol{u} \cdot \nabla u = \nabla \cdot (\boldsymbol{u} u)$ (which follows from the divergence-free constraint) in order to obtain a form to which Gauss's theorem applies. (Observe that this step automatically produces the conservation form of the momentum equations.)

The next step is to evaluate each of the terms in Eq. (2.44) on the grid cell shown in Fig. 2.8. We emphasize that this is not the same grid cell utilized in the analysis of the continuity equation. Indeed, when working with a staggered grid it is necessary to define separate grid cells for each of the momentum equations, and for the continuity equation; pressure, and all other scalar quantities, use the same grid cell as that employed for the continuity equation. We observe that the grid cell displayed in Fig. 2.8 is centered on the location of the x-component of velocity at the i^{th} grid line. The vertical cell walls correspond to locations $i - 1/2$ and $i + 1/2$, and horizontal cell walls coincide with the $j - 1$ and j grid lines. There are two main factors motivating this choice of cell boundaries. First, accurate evaluation of the first term on the left-hand side of Eq. (2.44) is simpler if the velocity component is at the cell center. Second, because pressure is often viewed, physically, as the main driving force in the momentum equations (in the absence of body forces), it is natural to define pressure on cell walls—pressure produces a surface force, so the natural place to represent it is on a cell boundary. We remark that this is completely analogous to the control-volume analyses learned in elementary fluid dynamics courses, but there the control volume would usually be of macroscopic size, rather than the grid cell used here that in principle will be allowed to go to zero area or volume.

In terms of the grid cell shown in Fig. 2.8 it is clear that the first term in Eq. (2.44) can be evaluated as

$$\frac{\partial}{\partial t} \int_{\Omega_{i,j}} u\, dV = \frac{\partial}{\partial t} u_{i,j-1/2} h_x h_y \,, \tag{2.45}$$

under the assumption that $u_{i,j-1/2}$ is the average x-component velocity within the cell. We note that since $u_{i,j-1/2}$ is cell centered, spatial integration over the grid cell presented in Fig. 2.8 corresponds to midpoint quadrature, and thus is third-order accurate, locally.

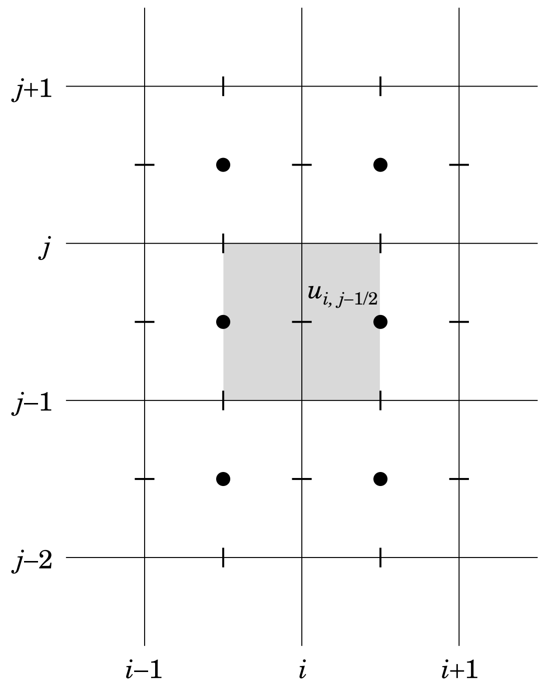

Figure 2.8: Grid cell (shaded region) for x-momentum equation.

The second term on the left-hand side is somewhat more difficult to evaluate because the x-component of velocity is not defined at any of the cell walls. Thus, averages (or, possibly, more general interpolations) must be used. On the $i + 1/2$ face it is natural to define

$$\overline{u}_{i+1/2,j-1/2} \equiv \frac{1}{2}(u_{i,j-1/2} + u_{i+1,j-1/2})\,, \tag{2.46}$$

with an analogous definition for the $i - 1/2$ face. Now on the j and $j - 1$ cell faces we must define average values of both the x- and y-components of velocity. For the x-component we can readily calculate

$$\widetilde{u}_{i,j} \equiv \frac{1}{2}(u_{i,j+1/2} + u_{i,j-1/2})\,, \tag{2.47}$$

and for the y-component

$$\overline{v}_{i,j} \equiv \frac{1}{2}(v_{i-1/2,j} + v_{i+1/2,j})\,. \tag{2.48}$$

It is of interest to note, as implied by the indexing, that these locations coincide with natural finite-difference grid points.

Clearly, analogous definitions hold on the $j - 1$ face. Also, observe that the convention being used here for notation is a "$-$" to denote averages in the x direction and a "$\sim$" for y-direction averaging. We can now approximate the second term on the left-hand side of (2.44) as

$$\int_{\partial\Omega_{i,j}} u^2 n_x + vun_y\, dA \simeq \left(\overline{u}^2_{i+1/2,j-1/2} - \overline{u}^2_{i-1/2,j-1/2}\right) h_y + \left(\overline{v}_{i,j}\widetilde{u}_{i,j} - \overline{v}_{i,j-1}\widetilde{u}_{i,j-1}\right) h_x\,. \tag{2.49}$$

On the right-hand side of Eq. (2.44) we evaluate the first term as

$$\int_{\partial\Omega_{i,j}} pn_x\, dA = \left(p_{i+1/2,j-1/2} - p_{i-1/2,j-1/2}\right) h_y\,, \tag{2.50}$$

with no formal averaging required. The second term takes the form

$$\begin{aligned}\nu\int_{\partial\Omega_{i,j}} u_x n_x + u_y n_y\,dA &= \nu\left[\left(u_x\Big|_{i+1/2,j-1/2} - u_x\Big|_{i-1/2,j-1/2}\right)h_y + \left(u_y\Big|_{i,j} - u_y\Big|_{i,j-1}\right)h_x\right] \\ &= \nu\Big[\left(u_{i+1,j-1/2} - 2u_{i,j-1/2} + u_{i-1,j-1/2}\right)\frac{h_y}{h_x} \\ &\qquad + \left(u_{i,j+1/2} - 2u_{i,j-1/2} + u_{i,j-3/2}\right)\frac{h_x}{h_y}\Big]\,. \end{aligned} \tag{2.51}$$

Again, no averaging is needed to construct the discrete form of this term.

We now combine the right-hand sides of Eqs. (2.45), (2.49), (2.50) and (2.51) and divide by $h_x h_y$ to obtain

$$\begin{aligned}\frac{\partial}{\partial t}u_{i,j-1/2} &= \nu\left[\frac{1}{h_x^2}\left(u_{i+1,j-1/2} - 2u_{i,j-1/2} + u_{i-1,j-1/2}\right) + \frac{1}{h_y^2}\left(u_{i,j+1/2} - 2u_{i,j-1/2} + u_{i,j-3/2}\right)\right] \\ &\quad - \frac{1}{h_x}\left(\overline{u}^2_{i+1/2,j-1/2} - \overline{u}^2_{i-1/2,j-1/2}\right) - \frac{1}{h_y}\left(\overline{v}_{i,j}\widetilde{u}_{i,j} - \overline{v}_{i,j-1}\widetilde{u}_{i,j-1}\right) \\ &\quad - \frac{1}{h_x}\left(p_{i+1/2,j-1/2} - p_{i-1/2,j-1/2}\right)\,. \end{aligned} \tag{2.52}$$

This is the *semi-discrete form* of the finite-volume x-momentum equation on a staggered grid, and except for the 1/2-index notation it looks very much like a typical finite-difference approximation; moreover, as should be clear from the preceding derivation, it is completely centered. We observe that (discrete) time integration can now be performed with any appropriate method to complete the discretization.

We next outline an analogous treatment for the y-momentum equation. The control-volume form of this equation is

$$\frac{\partial}{\partial t}\int_{\Omega_{i,j}} v\,dV + \int_{\partial\Omega_{i,j}} uv\,n_x + v^2 n_y\,dA = \int_{\partial\Omega_{i,j}} -pn_y + \nu(v_x n_x + v_y n_y)\,dA\,, \tag{2.53}$$

analogous to Eq. (2.44) for the x-momentum equation. The required control volume is shown in Fig. 2.9. We will need some additional definitions of averaged quantities in order to construct the required finite-volume approximations. Namely, on the top and bottom faces of the control volume the vertical velocity component must be defined as, for example on the top face,

$$\widetilde{v}_{i-1/2,j+1/2} \equiv \frac{1}{2}\left(v_{i-1/2,j} + v_{i-1/2,j+1}\right)\,. \tag{2.54}$$

On the vertical faces the horizontal component of velocity is defined as given earlier in Eq. (2.47):

$$\widetilde{u}_{i,j} \equiv \frac{1}{2}(u_{i,j+1/2} + u_{i,j-1/2})\,, \tag{2.55}$$

and similarly we have

$$\overline{v}_{i,j} \equiv \frac{1}{2}(v_{i-1/2,j} + v_{i+1/2,j})\,, \tag{2.56}$$

as already given in Eq. (2.48).

Application of the same type of control-volume analysis employed for the x-momentum equation results in

$$\begin{aligned}\frac{\partial}{\partial t}v_{i-1/2,j} &= \nu\left[\frac{1}{h_x^2}\left(v_{i+1/2,j} - 2v_{i-1/2,j} + v_{i-3/2,j}\right) + \frac{1}{h_y^2}\left(v_{i-1/2,j+1} - 2v_{i-1/2,j} + v_{i-1/2,j-1}\right)\right] \\ &\quad - \frac{1}{h_x}\left(\widetilde{u}_{i,j}\overline{v}_{i,j} - \widetilde{u}_{i-1,j}\overline{v}_{i-1,j}\right) - \frac{1}{h_y}\left(\widetilde{v}^2_{i-1/2,j+1/2} - \widetilde{v}^2_{i-1/2,j-1/2}\right) \\ &\quad - \frac{1}{h_y}\left(p_{i-1/2,j+1/2} - p_{i-1/2,j-1/2}\right)\,, \end{aligned} \tag{2.57}$$

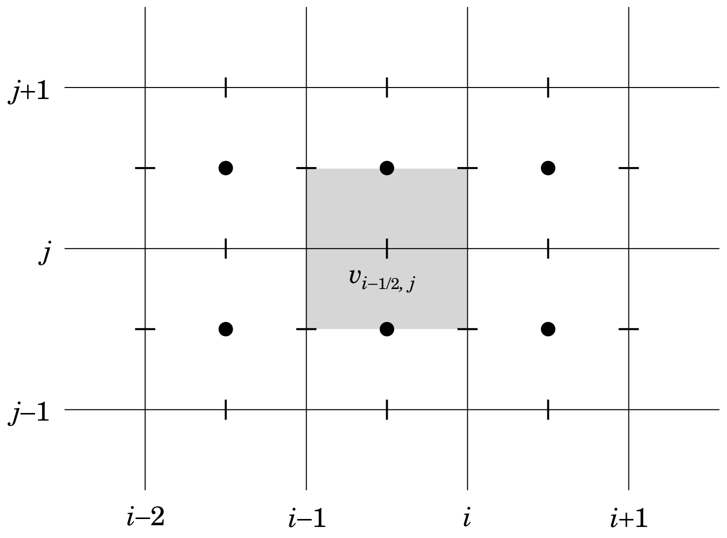

Figure 2.9: Grid cell (shaded region) for y-momentum equation.

for the semi-discrete y-momentum equation.

There are a number of observations we need to make regarding the semi-discrete system of equations (2.41), (2.52) and (2.57). The first is that we have not included a pressure Poisson equation, but we have noted that the same control volume would be used for this as is used for the continuity equation. We will introduce PPEs later when we consider specific solution methods. Second, we again emphasize that a different control volume is used to construct the discrete equations for each variable, and in 3D yet a fourth control volume is needed for the z-momentum equation. Next, we should note that the spatial discretizations are all formally centered, so they are expected to be second-order accurate in space; also, there is now no violation of the div-stability condition because of the staggered grid. Not only are the pressure gradient and advective terms approximated with centered differences, but this is done over a single grid cell of spacing h, rather than the $2h$ distances used with unstaggered grids. Thus, all things being equal, we would expect the finite-volume discretization to be more accurate than a corresponding finite-difference one. But, of course, "all things are not equal" because averaging has been used during the construction of the finite-volume approximations. This has been a concern for some time and has motivated, among other things, use of the multi-momentum equation versions of staggered gridding. It is thus worthwhile to see how much error is introduced by this use of averaging.

Truncation Error Due to Averaging. We first observe, as emphasized above, that no averaging is used in approximating the diffusive or pressure-gradient terms, so our only concern here will be the advective terms. We specifically analyze the nonlinear term from the x-momentum equation,

$$\frac{1}{h_x}\left(\overline{u}^2_{i+1/2,j-1/2} - \overline{u}^2_{i-1/2,j-1/2}\right) , \tag{2.58}$$

in Eq. (2.52), which is intended to approximate $\left(u^2\right)_x\big|_{i,j-1/2}$. To simplify notation we suppress the $j-1/2$ index since it is the same in all terms to be treated subsequently. We then recall from Eq. (2.46) that

$$\overline{u}_{i+1/2} = \frac{1}{2}(u_i + u_{i+1})\,,$$

and

$$\overline{u}_{i-1/2} = \frac{1}{2}(u_i + u_{i-1})\,.$$

We next note that for uniform grid spacing, $h_x = h$,

$$\begin{aligned} u_i &= u_{i+1/2} - \frac{h}{2}u'_{i+1/2} + \frac{h^2}{8}u''_{i+1/2} + \cdots\,, \\ u_{i+1} &= u_{i+1/2} + \frac{h}{2}u'_{i+1/2} + \frac{h^2}{8}u''_{i+1/2} + \cdots\,, \end{aligned}$$

which implies that

$$\overline{u}_{i+1/2} = u_{i+1/2} + \frac{h^2}{8}u''_{i+1/2} + \cdots\,,$$

where " $'$ " denotes differentiation (with respect to x in this case). Similarly, we obtain

$$\overline{u}_{i-1/2} = u_{i-1/2} + \frac{h^2}{8}u''_{i-1/2} + \cdots\,.$$

Substitution of the expansions for $\overline{u}_{i+1/2}$ and $\overline{u}_{i-1/2}$ into (2.58) results in

$$\begin{aligned} \frac{1}{h_x}\left(\overline{u}^2_{i+1/2} - \overline{u}^2_{i-1/2}\right) &= \frac{1}{h_x}\left(u^2_{i+1/2} - u^2_{i-1/2}\right) + \frac{h_x^2}{4}\left[\frac{1}{h_x}\left(u_{i+1/2}u''_{i+1/2} - u_{i-1/2}u''_{i-1/2}\right)\right] \\ &\qquad + \frac{h_x^4}{64}\left[\frac{1}{h_x}\left(u''_{i+1/2} - u''_{i-1/2}\right)\right] + \cdots\,, \\ &= \left(u^2\right)_x\Big|_i + \frac{h_x^2}{4}\left(uu_{xx}\right)_x\Big|_i + \mathcal{O}(h_x^4)\,. \end{aligned} \tag{2.59}$$

It is easily checked that the $\mathcal{O}(h^2)$ term on the right-hand side of this expression is of exactly the same form as the dominant truncation error term of a conserved-form centered-difference approximation using the natural finite-difference grid points $i-1$ and $i+1$, but it is actually <u>smaller</u>. In particular, the latter is $\frac{h^2}{3}uu_{xxx} + h^2u_xu_{xx}$, so each term of this is larger than the corresponding term in the staggered-grid finite-volume approximation, the second term on the right-hand side of Eq. (2.59) after expansion by differentiation. Thus, we see that the combination of averaging and then differencing over an interval half the usual size actually <u>reduces the error</u>, and concerns that have been raised regarding effects of averaging are probably unwarranted—at least in many situations. Moreover, we observe that the truncation error expansions contain only even powers of h; hence, Richardson extrapolation will be effective for improving accuracy. We leave as an exercise for the reader the task of comparing the corresponding errors in bilinear terms such as $(uv)_y$ and the nonlinear term $(v^2)_y$ from the y-momentum equation.

Notation and Implementation Considerations. As we have already noted, the staggered-grid formulation requires use of three separate control volumes in 2D (four in 3D), and in addition to this the one-half indexing is very inconvenient from the standpoint of coding and implementation. There is little doubt that these problems played at least some role in the use of the "NESW indexing" that has been widely used in CFD at least since introduction of the SIMPLE algorithm by Patankar, which we will study in detail in the next chapter. But this notation does not extend in any natural way to 3D, and it is difficult (but, of course, not impossible) to use for any but the simplest discretizations. Here we intoduce an approach that removes the problem of one-half indices without requiring storage of every point of the staggered grid. In particular, we will demonstrate how to employ "cell" notation and storage instead of the usual grid-point notation (and storage). In light of the fact that our discretizations have been constructed on grid cells via control-volume analysis on each such cell, this will be a natural approach. We note that it had first been introduced (using different notation) at least by the time of publication of Patankar's book [40].

Figure 2.10 provides a schematic of the notation, and a motivation for this approach in terms of the three overlapping staggered-grid control volumes. In this figure we display with dashed lines the three control volumes treated earlier, and we also indicate a mesh star associated with the collection of staggered

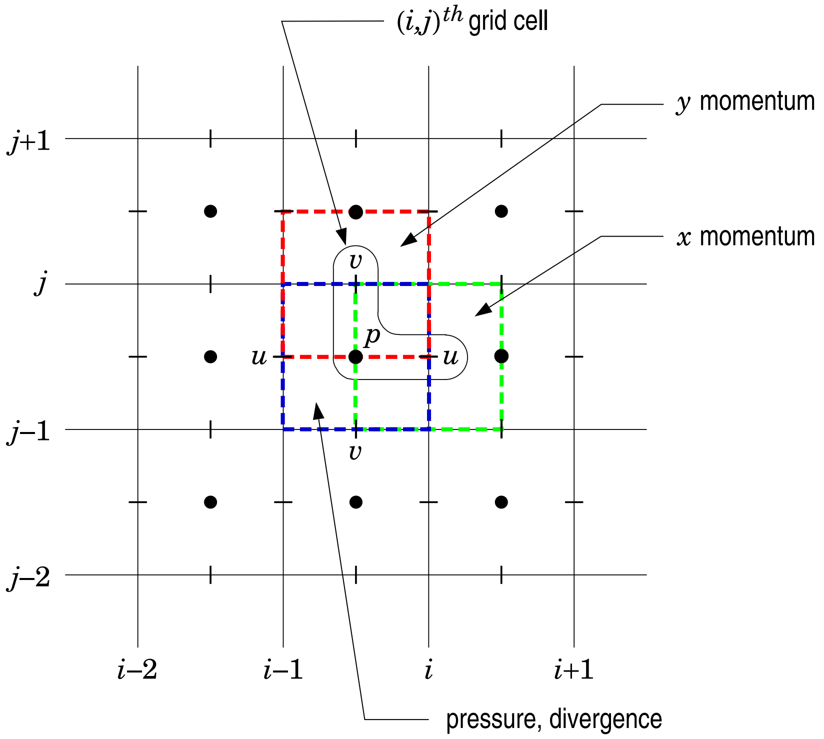

Figure 2.10: Grid-cell indexing.

points, one from the center of each of the three control volumes, corresponding to the dependent variables associated with the grid cell (i,j), *i.e.*, $\Omega_{i,j}$. (Observe that the cell multi index corresponds to that of the natural finite-difference grid point in the upper right-hand corner of the cell.) We remark that this configuration is not the only possible one, as the reader will quickly recognize, but in general all other alternatives are of no more (or less) utility than the one shown; and this one is widely used.

The important point to observe is that with this change of indexing, $u_{i,j-1/2} \longmapsto u_{i,j}$, $v_{i-1/2,j} \longmapsto v_{i,j}$, and $p_{i-1/2,j-1/2} \longmapsto p_{i,j}$, corresponding to the indicated mesh star, all cell values of dependent variables are taken into account as i and j are ranged over the total set of grid cells. Moreover, there is no extra storage needed in simple implementations beyond the one storage location per dependent variable, per grid cell—precisely the same as for an unstaggered grid. Finally, we can express the difference approximations obtained earlier as Eqs. (2.41), (2.52) and (2.57) in the non-fractional index notation as follows.

Continuity (divergence-free constraint):

$$\frac{1}{h_x}(u_{i,j} - u_{i-1,j}) + \frac{1}{h_y}(v_{i,j} - v_{i,j-1}) = 0 \qquad (2.60)$$

Momentum equations (Newton's second law of motion):

$$\frac{\partial}{\partial t}u_{i,j} = \nu\left[\frac{1}{h_x^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) + \frac{1}{h_y^2}(u_{i,j+1} - 2u_{i,j} + u_{i,j-1})\right]$$
$$- \frac{1}{h_x}(p_{i+1,j} - p_{i,j}) - \frac{1}{h_x}\left(\overline{u}^2_{i+1,j} - \overline{u}^2_{i,j}\right) - \frac{1}{h_y}\left(\overline{v}_{i,j}\widetilde{u}_{i,j} - \overline{v}_{i,j-1}\widetilde{u}_{i,j-1}\right)\,, \qquad (2.61a)$$

$$\frac{\partial}{\partial t}v_{i,j} = \nu\left[\frac{1}{h_x^2}(v_{i+1,j} - 2v_{i,j} + v_{i-1,j}) + \frac{1}{h_y^2}(v_{i,j+1} - 2v_{i,j} + v_{i,j-1})\right]$$
$$- \frac{1}{h_y}(p_{i,j+1} - p_{i,j}) - \frac{1}{h_x}\left(\widetilde{u}_{i,j}\overline{v}_{i,j} - \widetilde{u}_{i-1,j}\overline{v}_{i-1,j}\right) - \frac{1}{h_y}\left(\widetilde{v}^2_{i,j+1} - \widetilde{v}^2_{i,j}\right)\,. \qquad (2.61b)$$

The various averaged quantities used in these equations are now defined in terms of integer indices as

$$\overline{u}_{i,j} = \frac{1}{2}(u_{i,j} + u_{i-1,j})\,, \qquad (2.62a)$$

$$\overline{v}_{i,j} = \frac{1}{2}(v_{i,j} + v_{i+1,j})\,, \qquad (2.62b)$$

$$\widetilde{u}_{i,j} = \frac{1}{2}(u_{i,j} + u_{i,j+1})\,, \qquad (2.62c)$$

$$\widetilde{v}_{i,j} = \frac{1}{2}(v_{i,j} + v_{i,j-1})\,. \qquad (2.62d)$$

We comment that, at face value, the various approximations of first derivative terms appear to correspond to first-order accurate forward and backward differences, so it is important to recognize that this is only notation, and not the actual form (centered with step size h) of the difference approximations.

A final note is needed here with regard to implementation. Namely, most CFD post-processing tools (*e.g.*, *FieldView* and *TecPlot*) construct their displays based on unstaggered, natural grid-point locations. Thus, computed results from staggered (and also from cell-centered unstaggered) grids must be interpolated to these cell-vertex locations. Clearly, there are many possible approaches for doing this, and we leave selection of any specific one to the reader. We observe, however, that in practice very simple multi-linear interpolations are often employed, for good or for ill—even in commercial software.

Implementation of Boundary Conditions on Staggered Grids. Now that we have a practical indexing system at our disposal we can consider the final topic of this section, implementation of boundary conditions in the context of staggered-grid discretizations. Figure 2.11 displays a neighborhood of the bottom horizontal boundary of a computational domain. We will consider boundary conditions for x- and y-momentum equations and the pressure Poisson equation at the grid cell $\Omega_{i,2}$. Analogous treatments can be constructed on all other boundaries, and for other equations associated with any specific physical problem (*e.g.*, thermal energy or species concentration).

Since the grid line $j = 1$ coincides with the actual physical boundary, we observe that the no-slip condition for the x-momentum equation cannot be exactly satisfied, as we have noted earlier. In particular, this condition must be applied at the natural finite-difference grid point indicated with a solid square in the figure, *i.e.*, on the physical boundary. It is patently incorrect to attempt to impose the no-slip condition for the x-momentum equation at the point corresponding to the u-component of velocity in the $(i, 2)$ grid cell.

But it is clear that required discretizations in the y direction at this point will involve the u-component image point $(i, 1)$ whose value is unknown. Hence, in addition to attempting to satisfy a boundary condition at a point that is not even a part of the calculation, we will in any case need a grid-point value at a point outside the solution (and physical) domain for which we have no governing equation. Fortunately, the remedy for both of these difficulties is the same—and it is well known.

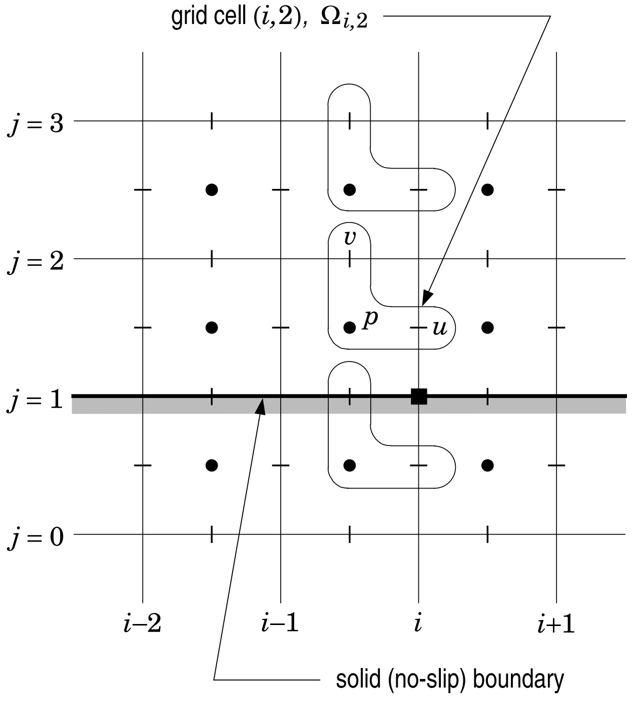

Figure 2.11: Staggered-grid boundary condition implementation schematic.

We construct an interpolation formula between the $(i,1)$ and $(i,2)$ grid cell u-velocity locations and evaluate this at the grid line $j=1$ where the no-slip boundary condition is to be imposed. In particular, for definiteness, suppose $j=1$ corresponds to $y=0$, and we have the boundary condition

$$u(x,0) = U_0(x)\,, \tag{2.63}$$

with $U_0(x)$ a prescribed function of x. Then straightforward linear interpolation yields

$$u|_{i,j=1} = U_0(x_i) = \frac{1}{2}(u_{i,1} + u_{i,2}) + \mathcal{O}(h_y^2)\,, \tag{2.64}$$

and it follows that the image-point value needed for discretization of y derivatives in the x-momentum equation in the grid cell $(i,2)$ is given by

$$u_{i,1} = 2U_0(x_i) - u_{i,2}\,. \tag{2.65}$$

This can be substituted into the discrete x-momentum equation and used in precisely the same way to eliminate the associated image point value as are Neumann conditions on natural, unstaggered grids. It is important to note that $u_{i,1} \neq 0$ in general, contrary to what a simple physical argument might suggest. At this point the problem is a mathematical one—equations representing the desired physics have already been presribed, and as emphasized throughout these lectures, it is dangerous to attempt solutions of such problems via physical reasoning alone.

We next consider boundary conditions for the y-momentum equation at the $(i,2)$ grid cell. We first treat a Dirichlet condition corresponding to specified "mass flux" (recall from elementary fluid dynamics that $\dot{m} = \rho VA$) across the physical $j=1$ boundary, and note that a staggered-grid location exists where this can be applied exactly. Thus, if the boundary condition is posed as

$$v(x,0) = V_0(x)\,,$$

then the associated numerical boundary condition is

$$v|_{i,j=1} = V_0(x_i)\,.$$

But because the numerical boundary point in this case coincides with the physical boundary, these values can be assigned in the usual way, with no approximation. Moreover, the difference approximation to the y-momentum equation is first employed in the $(i,2)$ cell because the boundary condition is of Dirichlet type. The y-direction discretization will include the point at $(i,1)$, and the value of $V_0(x_i)$ can be directly substituted for the corresponding grid-function value.

Just as the Dirichlet conditions are handled in the usual way for the y-momentum/horizontal boundary case, so also are Neumann conditions. In this situation it is clear that the boundary at $j=1$ cannot be solid. Furthermore, as is always the case with Neumann boundary conditions, the differential equation must be solved on the boundary. It is easily seen that the corresponding discretization will include a grid-point value on the $j=0$ line, *i.e.*, an image-point value. But the Neumann condition, when discretized in the usual way, will contain this same value; so it can be eliminated in a manner that is identical to that used in the same circumstance on a natural, unstaggered grid.

Finally, we consider the PPE. In the case of solid boundaries we have already seen that the correct boundary condition for the present case is Eq. (2.36), which we repeat here:

$$\frac{\partial p}{\partial y} = -(v_t + uv_x + vv_y - \nu\Delta v)\,. \tag{2.66}$$

For definiteness we suppose $v \equiv 0$ (*i.e.*, a homogeneous Dirichlet condition) holds on the $j=1$ boundary, and we leave as an exercise for the interested reader the task of identifying and treating other cases.

We first consider the left-hand side of this equation. The PPE, itself, will have been discretized in the grid cell $(i,2)$, but not in general in the image cell $(i,1)$. Approximation of $\partial^2 p/\partial y^2$ in the usual way then leads to

$$\frac{1}{h_y^2}(p_{i,1} - 2p_{i,2} + p_{i,3})\,,$$

and $p_{i,1}$ must be determined from the Neumann boundary condition.

Now just as was true for the x-momentum equation, the grid points for pressure do not occur on the horizontal boundary (in fact, for pressure, they do not coincide with any boundaries), but the Neumann condition must be applied on the physical boundary, *i.e.*, at $j=1$. This implies that $\partial p/\partial y|_{i,j=1}$ should be approximated as

$$\left.\frac{\partial p}{\partial y}\right|_{i,j=1} = \frac{1}{h_y}(p_{i,2} - p_{i,1})\,, \tag{2.67}$$

showing that $p_{i,1}$ can be expressed in terms of $p_{i,2}$ in order to eliminate it from the difference equation for the latter.

What remains is to treat the right-hand side of Eq. (2.66). In the specific case of $v \equiv 0$ on the boundary, as being considered here, it is clear that $v_t \equiv 0$, and $v_x \equiv 0$. Thus, the only nonzero term in the right-hand side of (2.66) is $\nu\partial^2 v/\partial y^2$. Because of the Dirichlet condition applied to the y-momentum equation in this case, there is no image-point value available with which to construct a centered approximation on the boundary in any direct way, except to first perform an extrapolation. A typically-better alternative is to employ a one-sided difference approximation. This can be done to maintain second-order accuracy with the following discretization:

$$\frac{\partial^2 v}{\partial y^2} = \frac{1}{h_y^2}(2v_{i,1} - 5v_{i,2} + 4v_{i,3} - v_{i,4}) + \mathcal{O}\left(h_y^2\right) \tag{2.68}$$

with $v_{i,1} \equiv 0$ in the present case. Although this appears somewhat complicated, it does not pose any serious difficulties because $\{v_{i,j}\}$ will already have been determined prior to its use in a boundary condition for pressure of the form Eq. (2.66).

The final case of boundary conditions to be considered is that of Dirichlet conditions for the pressure Poisson equation. These often arise as part of outflow boundary specification for incompressible flows, and they are sometimes used at inflows although this is generally ill posed for the N.–S. equations. The treatment is analogous to that employed to implement the no-slip condition for the x-momentum equation. In particular, the line $j = 1$ is the location at which pressure, $p|_{i,j=1}$ is to be assigned a value. Discretization of the PPE in the y direction on the grid cell $(i, 2)$ produces the need for a value of $p_{i,1}$. But this can be obtained as

$$p_{i,1} = 2p|_{i,j=1} - p_{i,2}\,,$$

with $p|_{i,j=1}$ given, just as in the x-momentum equation case.

We close this section by noting that more complicated Robin conditions, when appropriate, can be constructed as linear combinations of Dirichlet and Neumann conditions, so the treatment provided here covers essentially all cases of physical linear boundary conditions except more elaborate outflow conditions for the momentum equations.

Finite-Difference Approximation of N.–S. Equations on Cell-Centered Unstaggered Grids

In this subsection we will briefly treat an approach that in recent years has become quite popular and, indeed, is now often used in commercial CFD codes. It is use of cell-centered unstaggered, or co-located, grid systems in which all dependent variables are computed at the same locations. Such an approach is conceptually simpler than the staggered gridding treated in detail in the preceding section provided simple procedures can be found to circumvent the div-stability condition requirement described earlier.

As might be inferred from those discussions, a remedy must involve applying different orders of accuracy in approximating pressure and velocity on the same grid. In the finite-element context there are many natural ways in which this can be done because finite-element methods are defined (and constructed) in terms of discrete function spaces used to approximate the required dependent variables. For finite-difference/finite-volume methods the approach to satisfying the div-stability condition is less clear cut. Nevertheless, there have been several successful algoithms. We briefly describe the fundamentals of these here, leaving details to the specific cited references.

We will consider two different implementations of the co-located grid system. The first is due to Rhie and Chow [38]. This approach, while clearly only *ad hoc*, has had a major impact on recent incompressible flow code implementations, probably in large part because as noted earlier it is based on the highly-popular SIMPLE scheme of Patankar [40]. The modification required to satisfy the div-stability condition is in this case made in discretization of the pressure-gradient terms. The second approach we will treat is that of Zang *et al.* [41] which is based on the same projection method studied earlier by Kim and Moin [46], and termed a "fractional-step" method. In this case the modifications needed for satisfaction of the div-stability condition are introduced in approximations to momentum fluxes on cell boundaries.

Before proceeding to the main details of these two methods we provide a few general remarks specifically associated with the cell-centered grid depicted in Fig. 2.12. We first observe that all dependent variables are computed at cell centers, and no solution components are calculated at the natural finite-difference grid points. If a control-volume analysis is employed to construct the required approximations, the same control volume can be used for all variables. As a consequence of this, boundary condition treatment is the same for all variables; in particular, it is carried out in the same way as is done for pressure (and other scalars) in the staggered-grid arrangement. This, in turn, implies that no boundary conditions can be implemented exactly. Recall that in the staggered-grid case, while the no-slip condition could not be represented exactly, at least exact representation of mass fluxes across a boundary was possible; this is no longer the case with a co-located grid arrangement.

Second, we observe that the standard treatment of co-located grids is finite-volume based, and as a consequence fluxes must be computed on all cell walls via interpolation, as is true for staggered grids. Thus, the amount of required arithmetic, and the overall numerical accuracy, are generally quite similar.

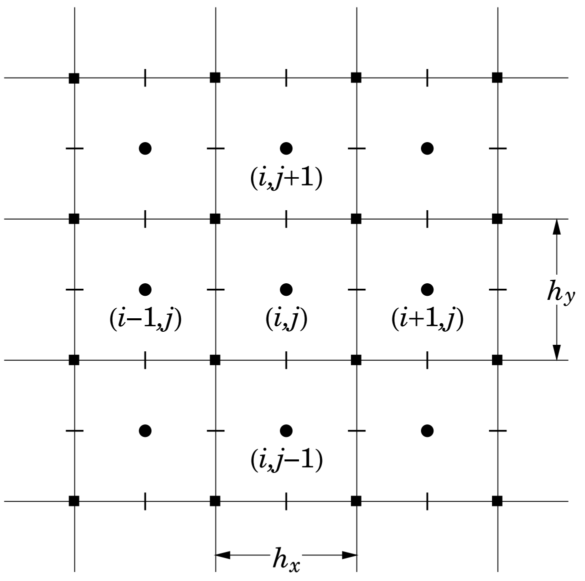

Figure 2.12: Cell-centered, unstaggered grid.

It should be noted, however, that because no solution components are directly computed on any cell wall, and in addition because of the need to circumvent the div-stability condition, the averaging (interpolation) needed is usually more elaborate than that required on a staggered grid for the same level of accuracy. Hence, the simplicity of a single control volume is for the most part offset by the additional effort needed to construct the approximations required because of the single control volume.

Finally, we again emphasize that calculation of all dependent variables at cell centers implies that no computational points coincide with the boundary of the computational domain. This not only leads to problems for implementing boundary conditions, as noted above, but it also imposes a requirement for solution averaging prior to output to a post processor. As is clear from Fig. 2.12, these averages typically must be constructed from solution values at at least four (in 2D, and six in 3D) locations surrounding the natural finite-difference grid point, and unless this is done carefully, significant numerical error can be introduced. (In the staggered-grid case, only two points per velocity component are needed to construct averages, even in 3D; but, of course, averaging of scalars must be done in the same way indicated here for the co-located grid.)

The Rhie & Chow Algorithm. As was mentioned earlier, the Rhie and Chow [38] approach was first presented in the context of a 2-D, steady-state SIMPLE algorithm. We will provide details of this algorithm in Chap. 3 and here be concerned only with the specific form of discretizations employed to circumvent the div-stability problem.

We begin by noting three specific points. First, no special treatment of second-order derivative diffusion terms has been employed for any of the methods treated herein. It appears that no such treatment is needed (but see Tafti [47]). Second, as we have already seen, fluxes must be calculated on cell walls for any approach based on a finite-volume formalism; in the case of co-located variables this can be done in the same way for all required variables. Finally, for the SIMPLE method, the pressure-gradient terms appear explicitly in the momentum equations.

In the Rhie & Chow algorithm, simple arithmetic averaging (which is second-order accurate) is employed for constructing fluxes on cell walls, and from consequences of the div-stability condition discussed earlier, we know this implies that pressure-gradient terms in the momentum equations must be approximated to a different order. Indeed, in the finite-element context one might use linear elements (second order) for velocity, and constant elements (first order) for pressure. In fact, Rhie and Chow introduce an alternative approximation of the pressure gradient based on a first-order difference approximation applied on cell walls.

It is clear that in general this lower-order accuracy might not be desirable in flow situations where accurate representation of pressure is crucial. On the other hand, we have earlier seen that theoretically it is not essential to calculate pressure accurately in order to obtain a correct velocity field. Moreover, as we will see in Chap. 3, details of the SIMPLE algorithm are such as to correct inaccuracies of the type induced by the above approximation as calculations are iterated to steady state.

The Zang et al. Procedure. Zang *et al.* [41] provide a cell-centered, unstaggered grid procedure for 3-D unsteady problems. Because their algorithm is based on a projection method for which pressure-gradient terms do not explicitly occur in the momentum equations, it is more convenient to alter discretization of velocity derivatives in order to circumvent the div-stability condition. This is done in [41] in both the advective terms of the momentum equations and in approximating the divergence of the velocity field appearing on the right-hand side of the (pseudo-) pressure Poisson equation. In either case, in order to approximate velocity derivatives at the cell-centered points, it is necessary to obtain velocity values interpolated to the cell walls. Because second-order centered discretizations were used for pseudo-pressure gradients in the projection step, consistent with standard second-order approximation of the Laplacian in the PPE, it is necessary to use something other than linear interpolation to obtain the cell-wall velocities.

Zang *et al.* [41] report using the QUICK interpolation of Leonard [48] (also see Fletcher [31]) given, *e.g.*, at the point $(i+1/2, j)$ as

$$\begin{aligned} u_{i+1/2,j} &= \frac{1}{2}(u_{i,j}+u_{i+1,j}) - \frac{q}{3}(u_{i-1,j}-2u_{i,j}+u_{i+1,j}) \\ &= -\frac{q}{3}u_{i-1,j} + \left(\frac{2}{3}q+\frac{1}{2}\right)u_{i,j} - \left(\frac{q}{3}-\frac{1}{2}\right)u_{i+1,j}\,, \end{aligned}$$

with q chosen to improve accuracy or stability. ($q = 3/8$ is generally associated with QUICK.) Observe that $q = 0$ corresponds to linear interpolation, which does not lead to satisfaction of the div-stability condition, but the results in [41] suggest that $q = 3/8$ does satisfy this.

We note that the required arithmetic for this form of interpolation is approximately double that of the basic arithmetic average employed on a staggered grid, but in addition the differencing, itself, is based on QUICK, so the total arithmetic for advective terms in the momentum equations is as much as a factor of four higher than would be required on a staggered grid. Moreover, the QUICK discretization of derivatives is only second-order accurate (but the above interpolation is third order), so a considerable amount of extra arithmetic beyond that needed for a staggered grid arrangement has been expended simply to satisfy the div-stability condition imposed by the unstaggered grid.

In concluding this brief section on co-located grid treatments we observe that the generally *ad hoc* approaches constructed to allay the div-stability condition difficulties are not guaranteed to be effective in all situations. Symptoms of failure include the typical checkerboard velocity and/or pressure fields discussed earlier. One of the simplest remedies is to use solutions (grid-function values) only from every second discrete point (in each coordinate direction). Hence, in 2D far more than four times the arithmetic required for a staggered-grid solution is expended to produce a solution on only one fourth as many grid points. Another alternative is to employ smoothing methods (to be discussed below in the context of aliasing of under-resolved calculations) to remove solution "oscillations" associated with checkerboarding. Both of these approaches treat symptoms rather than causes, and in the present case this is probably not the best strategy since use of a staggered grid removes the cause of the problems leading to checkerboard, non-physical discrete solutions.

2.3 Treatments for the Cell-*Re* Problem and Aliasing

In addition to the pressure-velocity decoupling treated in the preceding section, those writing CFD codes for solving the incompressible N.–S. equations must be able to handle yet another form of non-physical solutions, namely, those arising from the so-called cell-*Re* problem, and from aliasing. We will consider each of these in the current section. In the first subsection we will give a basic treatment of the cell-*Re* problem, indicating how it arises and some well-known remedies. In a second subsection we provide an introductory analysis of the more general problem of aliasing, the treatments of which generally also work for the cell-*Re* problem as well.

2.3.1 The cell-*Re* problem—its definition and treatment

In this subsection we begin by showing how the cell-*Re* problem arises as a consequence of specific properties of solutions to difference equations. This will provide some insight into how the problem might be treated, and we will introduce some examples of this. It will be seen that, contrary to what has sometimes been claimed, the problem does not arise simply from nonlinearities of the Navier–Stokes equations. Indeed, linear equations can, when discretized, exhibit the same kind of behavior as is seen in the N.–S. equation solutions. Thus, our analysis will be performed with linear model problems for which exact solutions can often be derived. Furthermore, introduction of time dependence has little effect on the basic phenomenon involved, so we will restrict attention to the simpler steady-state case.

A Burgers' Equation Model Problem

Burgers' equation has been widely used as a (usually) 1-D model of the incompressible (and, sometimes, compressible) momentum equations of the N.–S. system. The homogeneous Burgers' equation,

$$u_t + uu_x = \nu u_{xx}\,, \tag{2.69}$$

is one of the few nonlinear equations to possess an exact solution. Here, we will consider a significant simplification that will be adequate for our purposes. Namely, as noted above, we will treat the steady-state, linearized form

$$Uu_x - \nu u_{xx} = 0\,, \tag{2.70}$$

where U is an assigned constant.

This equation provides a 1-D model of the balance between advection and diffusion in a constant-velocity flow. (In this context, we must clearly view u as some other flow property, *e.g.*, temperature.) We now discretize Eq. (2.70) with centered differences to obtain

$$UD_{0,x}u_i - \nu D^2_{0,x}u_i = 0\,,$$

or

$$\frac{U}{2h}(u_{i+1} - u_{i-1}) - \frac{\nu}{h^2}(u_{i-1} - 2u_i + u_{i+1}) = 0\,,$$

where h is the discretization step size. Multiplication by h/U leaves this in the form

$$\frac{1}{2}(u_{i+1} - u_{i-1}) - \frac{\nu}{Uh}(u_{i-1} - 2u_i + u_{i+1}) = 0\,. \tag{2.71}$$

We now define the *cell Reynolds number* as

$$Re_h \equiv \frac{Uh}{\nu}\,. \tag{2.72}$$

We note that if we were considering, *e.g.*, convective heat transfer, then ν would be replaced with κ, the thermal diffusivity, and we would replace cell *Re* with the *cell Péclet number*, defined as

$$Pe_h \equiv \frac{Uh}{\kappa}\,. \tag{2.73}$$

Analogous terminology applies for any advective-diffusive (transport) equation. It is of interest to note that unlike the corresponding dimensionless counterparts, Re_h and Pe_h may be negative; that is, U may take on either sign. Furthermore, we point out that if the governing equation is already cast in dimensionless form so that, *e.g.*, $\nu \to 1/Re$, then the cell Reynolds number is calculated as $Re_h = Re\,h$, where Re is the usual dimensionless Reynolds number, and h is a dimensionless (scaled with the same length used to define Re) grid spacing. Obviously, in this formulation $Re_h \geq 0$ must hold.

Some Basic Theory of Difference Equations

In this subsection we will describe, both heuristically and with fairly rigorous mathematics (the theoretical structure of solutions to difference equations), how the cell-Re problem arises. The mathematical treatment will suggest ways to construct (partial) remedies, but we will also see that there is no completely universal and affordable (from the standpoint of computer run times) way to deal with this problem.

The starting point toward understanding what the cell-Re problem is comes from a property of elliptic and parabolic differential equations known as a *maximum principle.* There are numerous of these, and the reader is referred to the various references on PDEs already cited in these lectures for detailed treatments. Here, we will provide just an heuristic description. Suppose $\mathcal{L}$ is a linear partial differential operator of elliptic type, and consider the problem

$$\mathcal{L}u = 0 \qquad \text{on} \quad \Omega \subseteq \mathbb{R}^d\,, \quad d = 1, 2, 3\,,$$

with boundary conditions

$$u(\boldsymbol{x}) = g(\boldsymbol{x})\,, \qquad \boldsymbol{x} \in \partial\Omega\,,$$

where $g(\boldsymbol{x})$ is a prescribed function. Then the maximum principle states that the maximum and minimum values taken on by $u \in \overline{\Omega}$ must occur on $\partial\Omega$. We remark that this principle is widely used in the theory of elliptic PDEs to demonstrate uniqueness of solutions without ever having to actually construct the solution.

For our purposes herein it is useful to note that a similar principle exists for difference approximations to elliptic equations. Moreover, it can be shown (see [20]) that the following two conditions are sufficient to guarantee satisfaction of such a maximum principle for discretizations of the general form

$$\mathcal{L}_h u_{\boldsymbol{i}} = a_{\boldsymbol{0}} u_{\boldsymbol{i}} - \sum_{\boldsymbol{\alpha}=\boldsymbol{s}^-}^{\boldsymbol{s}^+} a_{\boldsymbol{\alpha}} u_{\boldsymbol{i}+\boldsymbol{\alpha}} = 0\,.$$

i) <u>nonnegativity</u>: $a_{\boldsymbol{0}} > 0$ and $a_{\boldsymbol{\alpha}} \geq 0$, $\boldsymbol{\alpha} \in [\boldsymbol{s}^-, \boldsymbol{s}^+]$, the set of neighbors of $\boldsymbol{i}$, where bold-faced quantities represent multi-indices with dimension consistent with the spatial dimension of the problem being considered.

ii) <u>diagonal dominance</u>: $\sum_{\boldsymbol{\alpha}=\boldsymbol{s}^-}^{\boldsymbol{s}^+} a_{\boldsymbol{\alpha}} \leq a_{\boldsymbol{0}} \;\forall\; \boldsymbol{i}$ in the grid function index set, and with strict inequality for at least one $\boldsymbol{i}$.

We note that these requirements do not depend on the spatial dimension of the problem or on the specific form of differencing used to construct the approximation. In particular, they are sufficient even on unstructured grids.

We remark that the importance of the maximum principle in the present context is with regard to local, rather than global (over the whole spatial domain) behavior. In particular, if we consider placing an "internal boundary" around each mesh star, the maximum principle implies that the grid function value at the point (i, j) cannot exceed that of any of its (nearest) neighbors, and because there is a corresponding minimum principle, the grid-function value cannot be lower than that of any neighbor. In particular, it cannot oscillate from point to point.

As we implied at the beginning of this section, this initial treatment is intended to be heuristic. We will now take a more precise look at the cell-Re problem by studying the theoretical behavior of solutions to linear difference equations. In particular, we will solve Eq. (2.71) analytically and investigate the nature of the solution(s) as cell-Re is varied.

If we use Eq. (2.72), the definition of Re_h, in (2.71) we obtain

$$\frac{Re_h}{2}(u_{i+1} - u_{i-1}) - (u_{i-1} - 2u_i + u_{i+1}) = 0\,,$$

or, after rearrangement,

$$-\left(1 + \frac{Re_h}{2}\right) u_{i-1} + 2u_i - \left(1 - \frac{Re_h}{2}\right) u_{i+1} = 0\,. \tag{2.74}$$

It is immediately clear from this form of the difference equation that $|Re_h| > 2$ will result in failure to satisfy the above the non-negativity condition, which in turn means (possible) failure of the maximum principle and consequent local maxima and minima in the interior of the solution. We will now show in some detail how this occurs. In order to do this we will need to introduce some facts regarding linear difference equations.

Definition 2.1 *The order of a difference equation is the difference between the highest and lowest subscripts appearing on its terms.*

Thus, in the present case we see that the order is $(i + 1) - (i - 1) = 2$. We also have the following.

Theorem 2.1 *The number of independent solutions to a linear difference equation is equal to its order.*

Further, we note that (linear) difference equations are solved (analytically) in a manner quite similar to what is done for differential equations, as we will now demonstrate for a simple example that is of the same form as Eq. (2.74). In particular, consider the general second-order linear difference equation

$$a_2 y_{i+1} + a_1 y_i + a_0 y_{i-1} = 0\,, \tag{2.75}$$

whose *characteristic equation* is

$$a_2 z^2 + a_1 z + a_0 = 0\,,$$

or

$$z^2 + \frac{a_1}{a_2} z + \frac{a_0}{a_2} = 0. \tag{2.76}$$

In the cases of second-, third- and fourth-order difference equations this characteristic equation can be solved exactly (in terms of elementary functions) since it is just a polynomial of degree equal to the order. In particular, the solutions to Eq. (2.76) are

$$z_\pm = \frac{-\dfrac{a_1}{a_2} \pm \sqrt{\left(\dfrac{a_1}{a_2}\right)^2 - 4\left(\dfrac{a_0}{a_2}\right)}}{2}\,, \tag{2.77}$$

and the corresponding independent solutions to Eq. (2.75) are

$$y_\pm = (z_\pm)^i\,, \qquad i = 1, \ldots, N\,, \tag{2.78}$$

with N being the number of discrete points in space, or discrete time indices, as appropriate.

The final piece of information needed for a sufficient understanding of solutions to difference equations is the following.

Theorem 2.2 *The general solution to the difference equation* (2.75) *is a linear combination of the independent solutions given by* (2.78).

Thus, we have

$$y = c_1 z_+^i + c_2 z_-^i \,, \tag{2.79}$$

where c_1 and c_2 are constants to be determined from either initial or boundary conditions associated with a specific problem.

Before continuing, we remark that this level of information on difference equations can be found in most standard texts on numerical analysis, and further details can be found in more advanced treatments such as given in the monograph by Mickens [49].

The Cell-*Re* Problem

At this point it is time to focus on the cell-Re "problem." In numerical computations involving transport equations the symptom of this problem is (non-physical) oscillation of the numerical solution from one grid point to the next, as displayed schematically in Fig. 2.13. It is clear from (2.78) and (2.79) that if z_+ and z_-

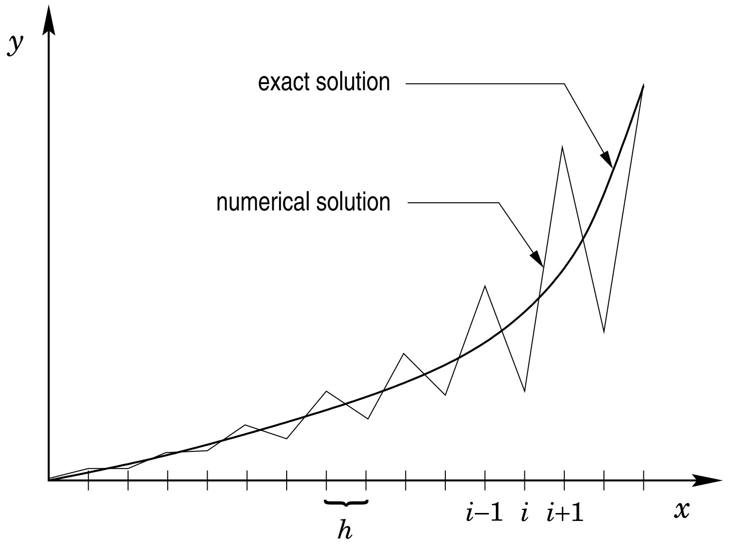

Figure 2.13: Grid point to grid point oscillations caused by cell-Re problem.

are not <u>both</u> nonnegative, it it possible for cell-Re "wiggles," as displayed in the figure, to occur. It should further be observed that the magnitude of these oscillations tends to increase with increasing solution magnitude, corresponding to increasing cell Re. Moreover, as we have already noted, such increases result in loss of the non-negativity and diagonal dominance properties of the difference equation.

As we have already indicated, our model Burgers' equation problem is in a form identical to that of Eq. (2.75), and it can be shown (an exercise for the reader) that solutions take the form

$$z_+ = \frac{1 + \frac{1}{2}Re_h}{1 - \frac{1}{2}Re_h} \,, \qquad \text{and} \qquad z_- = 1 \,. \tag{2.80}$$

We see from this that $z_+ \to 1$ as $Re_h \to 0$, and $z_+ \to -1$ as $Re_h \to \pm\infty$. Moreover, z_+ first becomes negative as $|Re_h|$ exceeds 2, and also $z_+ \to \pm\infty$ as $Re_h \to 2$ from below and above, respectively—in accord with the result we obtained from the earlier heuristic analysis based on the maximum principle. Thus, the well-known cell-Re restriction associated with centered-difference approximations can be derived purely from the mathematical theory of difference equations without invoking any physical arguments such as, for example, appealing to flow direction to suggest inadequacy of centered approximations.

We comment that failure to maintain $|Re_h| < 2$ represents a form of under resolution as can be seen from the definition in Eq. (2.72). But this is different from under resolution in the usual sense where the implication is that there are insufficient grid points to represent the solution to the differential equation. Rather, as is clear from the above discussion, $|Re_h| < 2$ simply results in undesirable properties in solutions to the difference equations. Nevertheless, these properties are quite similar to those occurring due to under resolution in the usual sense, suggesting that they might be treated in a similar way.

It is also worthwhile to plot values of z_+ as a function of Re_h. We note, from the definition of Re_h that we can view such a plot as representing effects of changing grid spacing with velocity and viscosity fixed, of changing velocity with grid spacing and viscosity fixed or of changing viscosity with velocity and grid spacing fixed. In actual physical problems any of these combinations might be appropriate (even within the same problem). Figure 2.14 displays the variation of the non-constant solution of Eq. (2.75)

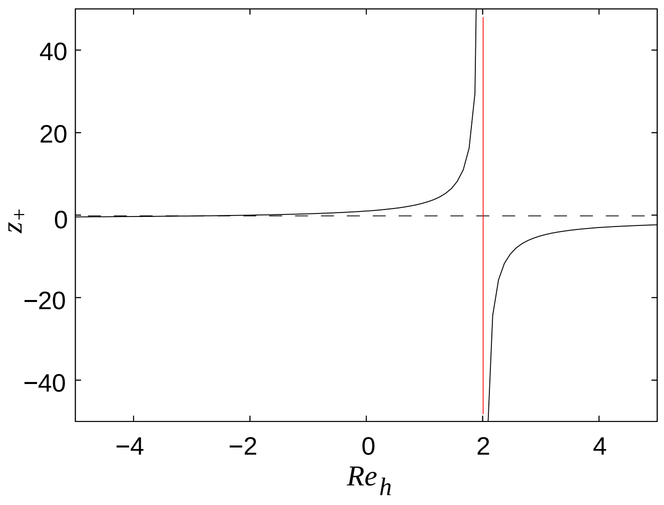

Figure 2.14: Dependence of z_+ on cell Re.

as a function of Re_h. There are several interesting points to be made with regard to this figure. First, we see as is already evident from the analytical formula that $z_+ < 0$ when $|Re_h| > 2$. Second, it is also clear that the range of cell Re over which the magnitude of z_+ is much greater than that of z_- is rather limited. This means that for certain combinations of boundary conditions the cell-Re problem might not be particularly significant even though the basic centered-difference approximation may have violated the discrete maximum principle. Moreover, we also see that the situation is, in general, much worse for $Re_h > 0$ than for $Re_h < 0$. Since U is the only factor in the definition of Re_h that can take on both signs, it is apparent that the cell-Re problem is evidently less severe in reversed-flow situations than in high-speed forward flows.

But in addition, it is important to recognize that the numerical solution actually computed depends on the constants c_1 and c_2 in Eq. (2.79), which are set by (in the present case) boundary conditions. If it happens that $c_2 \gg c_1$, then cell-Re effects will be significant only for Re_h slightly greater than two. Even so, in a high-speed viscous flow in which $|Re_h| \gg 2$ in much of the flow field, it is still likely that $|Re_h|$ will be only slightly greater than two near solid boundaries, in some regions of shear layers and in the cores of vortices. Thus, some cell-Re effects are likely to be seen in essentially any simulation that is even slightly under resolved. Moreover, in multi-dimensional flow fields the overall situation is considerably more complicated because all components of velocity, and their associated boundary conditions, contribute (sometimes in ways leading to local cancellations) to the final form of the solution. Indeed, in multi-dimensional problems there is no longer a single value of Re_h at each grid point; instead, there is, in general, a distinct value for each separate advective term of the momentum equations.

We now demonstrate for the model problem that the conditions needed to prevent oscillations in the exact solutions to the difference equation are, in fact, those that guarantee satisfaction of a maximum principle. We first note that along with the required nonnegativity of $z_\pm$, for physical problems we should also require $z_\pm \in \mathbb{R}$. This latter requirement implies that in Eq. (2.77)

$$\left(\frac{a_1}{a_2}\right)^2 \ge 4\left(\frac{a_0}{a_2}\right),$$

or

$$|a_1| \ge 2\sqrt{a_0a_2}\,. \tag{2.81}$$

Now if we are to permit equality to hold in (2.81), the solution given by Eq. (2.77) can be positive only if

$$\operatorname{sgn}(a_1a_2) = -1\,. \tag{2.82}$$

On the other hand, if strict inequality holds in (2.81), then we must require in addition to (2.82) that

$$\frac{a_1}{a_2} \ge \sqrt{\left(\frac{a_1}{a_2}\right)^2 - 4\left(\frac{a_0}{a_2}\right)}\,.$$

But this can be guaranteed only if

$$\operatorname{sgn}(a_0a_2) = +1\,. \tag{2.83}$$

We note that conditions (2.82) and (2.83), taken together, are equivalent to the usual non-negativity requirement.

We will now show that satisfaction of the inequality (2.81) is equivalent to diagonal dominance. Again using (2.82) and (2.83), we see that we can assume $a_0, a_2 \ge 0$ and $a_1 < 0$ hold. In this case, diagonal dominance would imply

$$|a_1| \ge |a_0| + |a_2|\,. \tag{2.84}$$

Now since a_0 and a_2 are nonnegative, their square roots are real, and we have

$$(\sqrt{a_0} - \sqrt{a_2})^2 \ge 0 \quad \Rightarrow \quad a_0 - 2\sqrt{a_0a_2} + a_2 \ge 0\,.$$

Thus,

$$2\sqrt{a_0a_2} \le a_0 + a_2 = |a_0| + |a_2|\,. \tag{2.85}$$

Now suppose diagonal dominance holds. Then

$$|a_1| \ge |a_0| + |a_2| \ge 2\sqrt{a_0a_2}\,;$$

hence, inequality (2.81) is satisfied. On the other hand, suppose (2.81) does not hold; *i.e.*,

$$|a_1| < 2\sqrt{a_0a_2}\,.$$

Then, from inequality (2.85) it follows that diagonal dominance does not hold, completing the equivalence proof.

Remedies for the Cell-*Re* Problem

In this subsection we will introduce several of the more widely-used treatments of the cell-*Re* problem. As we have hinted earlier, none of these provide a complete solution to the problem; moreover, from discussions in the preceding section, it would appear likely that there cannot be a single approach that is guaranteed to always work without causing other difficulties. The specific techniques we consider here are first-order upwinding, the so-called "hybrid" scheme, second-order upwinding and QUICK. Nearly all commercial CFD codes provide options for employing all of these except second-order upwinding (which is available in some code, but not on others)—in addition to use of standard centered differencing, and in some codes even further options are available, thus underscoring the significance attached to this problem.

To understand why this is so, recall the basic centered approximation of the Burgers' equation model problem, expressed here as

$$\left(1+\frac{Re_h}{2}\right)u_{i-1}-2u_i+\left(1-\frac{Re_h}{2}\right)u_{i+1}=0\,. \tag{2.86}$$

We again note that to prevent non-physical oscillatory solutions it is generally necessary to require

$$|Re_h|\le 2\,, \tag{2.87}$$

which is termed the *cell-Re restriction* for centered-difference approximations. The consequence of this restriction is readily seen. Namely, suppose the flow velocity is $\mathcal{O}(1)$ and viscosity $\nu \sim \mathcal{O}(10^{-4})$, both of which are reasonable magnitudes for physical problems. Now from inequality (2.87) it follows that the grid spacing h must satisfy

$$h\le\frac{2\nu}{|U|}\,. \tag{2.88}$$

Hence, in the present case we have $h \le 2{\times}10^{-4}$, which in a 3-D calculation on the unit cube would require over 10^{11} grid points. Clearly, this is a nearly intractable problem on current computing hardware, and it will probably not be routinely possible to achieve the industrial standard of "overnight turnaround" for problems of this size any time in the next 10 years.

First-Order Upwinding. So-called *first-order upwind* differencing was probably the first proposed remedy for the cell-*Re* problem. It was introduced on the basis of *ad hoc* "physical reasoning" (see, *e.g.*, Patankar [40]). Namely, it was argued that the reason centered differencing led to non-physical solution oscillations was that it could sense both upstream and downstream flow behavior, and it combined these in its representation of the velocity gradient. It was further argued that a difference approximation using only information carried in the flow direction would be more appropriate; *i.e.*, the difference approximations should be one-sided (or at least biased) into the flow direction. The simplest such procedure is the basic first-order difference.

For the Burgers' equation model problem with $U > 0$, this takes the form

$$UD_-u_i-\nu D_0^2u_i=0\,,$$

or after expanding the difference operator symbolism, introducing the definition of cell *Re* and rearranging the results,

$$\left(1+\frac{1}{Re_h}\right)u_{i-1}-\left(1+\frac{2}{Re_h}\right)u_i+\frac{1}{Re_h}u_{i+1}=0\,. \tag{2.89}$$

It is easily checked that this difference equation satisfies both the non-negativity and diagonal dominance conditions needed to guarantee that no interior maxima or minima exist, independent of the value of Re_h, and it is also easy to directly solve this difference equation, as we have already done for the centered-difference case. The result of this is that both independent roots of the characteristic equation analogous to Eq. (2.76) are nonnegative for any value of $Re_h > 0$. This latter requirement is guaranteed to hold, in general, if we switch to a forward-difference approximation in cases where $U < 0$. It is left as an exercise for the reader to analyze this case; the results are essentially the same as those just given. Thus, we can summarize the first-order upwind discretization as follows, expressed here for non-constant U in non-conserved form:

$$U_iu_x|_i=\begin{cases}U_iD_-u_i\,, & U_i>0\,,\\[2ex] U_iD_+u_i\,, & U_i<0\,.\end{cases} \tag{2.90}$$

There are at least two major difficulties with first-order upwind treatment of the cell-*Re* problem, and these were recognized fairly soon after its introduction. The analysis of this that we present here follows that of de Vahl Davis and Mallinson [50], who were among the first to analyze this approach in detail.

It is clear that the simple first-order differences employed in Eq. (2.90) degrade the overall accuracy of a discretization in any case that second-, or higher-, order methods are being used for terms other than advective terms, and in conjunction with this is the fact that this reduction in accuracy is global since, presumably, we are usually solving boundary value problems. But there are additional, more subtle, problems that arise from first-order upwind differencing. To see this we consider the dimensionless form of the Burgers' equation model problem,

$$Uu_x - \frac{1}{Re}u_{xx} = 0\,, \tag{2.91}$$

with $U > 0$. Equation (2.90) implies that upwind differencing will employ the backward difference

$$D_-u_i = \frac{u_i - u_{i-1}}{h}$$

in the advective term, and expansion of u_{i-1} yields the following:

$$u_{i-1} = u_i - hu_x\big|_i + \frac{h^2}{2}u_{xx}\big|_i - \frac{h^3}{6}u_{xxx}\big|_i \pm \cdots\,.$$

This shows that Eq. (2.91) can be expressed as

$$Uu_x - \left(\frac{1}{Re} + \frac{Uh}{2}\right)u_{xx} + \frac{h^2}{6}Uu_{xxx} \pm \cdots = 0\,,$$

or

$$Uu_x - \frac{1}{Re^*}u_{xx} + \mathcal{O}(h^2) = 0\,, \tag{2.92}$$

where Re^* is the *effective Reynolds number* defined as

$$Re^* \equiv \frac{1}{\dfrac{1}{Re} + \dfrac{Uh}{2}}\,. \tag{2.93}$$

Equation (2.92) is often termed the "modified equation," and its significance is that it, rather than the original PDE, is solved to within <u>second-order</u> accuracy instead of first-order accuracy implied by the discretization. This suggests that the numerical solution is a more accurate representation of solutions to the modified equations than to solutions of the actual (in the case of the N.–S. equations, physical) equation(s). Thus, it is important to consider this modified equation in more detail.

What is rather obvious is that the only difference between physical and modified equations in the case of first-order upwinding is the coefficient of the diffusion term. That is, in the modified equation the original physical Reynolds number is replaced with the effective Reynolds number, Re^* given in Eq. (2.93), and it is essential to understand the consequences of this. Indeed, it has often been argued that for very high Re, the flow is convection dominated, and since upwinding imposes the "correct physics," use of this approximation should be quite accurate. But this is a spurious argument, especially when simulating wall-bounded flows. In such flows, no matter what the value of Re, some regions of the flow will be significantly affected by viscous forces; consequently, if anything is done to the equations of motion to alter this aspect of the physics (the diffusion terms), it cannot be expected that accurate solutions will be obtained.

With this in mind, it is of interest to examine the specific effects of $Re \to \infty$ in the modified equation. In particular, consider the limit of Re^* as $Re \to \infty$. It is easily seen from Eq. (2.93) that

$$\lim_{Re\to\infty} Re^* = \frac{2}{Uh}\,. \tag{2.94}$$

Hence, as the physical Reynolds number approaches infinity, the computational one approaches a finite limit—in fact, one that could be quite small if coarse grids are employed. This provides an explanation for a phenomenon observed in early uses of first-order upwinding; namely, it was seen that solutions for a given physical situation and discrete approximation did not appear to change much with Re after the value of this parameter exceeded $\sim \mathcal{O}(10^3)$. Of course, in most situations this is counter intuitive on a physical basis, and it provided motivation for investigations of the sort we have just presented.

Clearly, the ultimate effect of first-order upwinding is to increase the coefficient of the diffusion term(s) in the governing equations. The factor producing this effect, $Uh/2$, (and similar factors arising in other methods) is often called *numerical diffusion* or *numerical viscosity.* Although these factors are related to, and in many ways similar to, "artificial viscosity," or "artificial dissipation," in the present lectures we will consider them to be distinct. In particular, numerical diffusion is, in general, an unwanted effect arising in an uncontrolled way from (usually) inadequate numerical analytic procedures—basically mistakes, whereas in essentially all cases artificial dissipation is added deliberately and at least to some extent, controllably, in a manner that has minimal effect on the physics represented by the final discrete equations. Furthermore, as is evident from (2.94), numerical diffusion can often completely dominate physical diffusion, and this is not usually the case with artificial dissipation except when it is employed with inviscid forms (the Euler equations) of the equations of motion.

The Hybrid Scheme. One of the early modifications that is still widely used today, especially as an option in commercial CFD software, is the so-called "hybrid" scheme of Spalding [51] (also, see [40]). In this approach the solution is tested locally to determine whether a centered difference cell-Re restriction violation is likely, and if it is the scheme is automatically switched to first-order upwinding. To construct this method for the model Burgers' equation problem, we now consider the conservation form and allow U to be the variable U_i, leading to

$$(Uu)_x - \nu u_{xx} = 0\,. \tag{2.95}$$

We first express the discrete equation in the general form valid for any 1-D three-point grid stencil,

$$a_1 u_{i-1} + a_2 u_i + a_3 u_{i+1} = 0\,,$$

and then define the difference equation coefficients as follows:

$$a_1 = \max\left[-U_{i-1}h, \nu - \frac{U_{i-1}h}{2}, 0\right]\,, \tag{2.96a}$$

$$a_2 = -a_1 - a_3 - h\left(U_{i+1} - U_{i-1}\right)\,, \tag{2.96b}$$

$$a_3 = \max\left[U_{i+1}h, \nu + \frac{U_{i+1}h}{2}, 0\right]\,. \tag{2.96c}$$

It is important to recognize that if use of the hybrid method results in switching to first-order upwinding at very many grid points, the solution behavior will be no different than that of first-order upwinding itself, and this is often the case. Thus, we do not recommend its use. We have included it here simply for the sake of completeness; as we have already mentioned, it is widely used in commercial CFD codes.

In summarizing this presentation on first-order upwind differencing, we again emphasize that, while it uniformly cures the cell-Re problem, its use results in a possibly drastic alteration of the physics being considered; the end result is that solutions obtained in this way cannot, in general, be trusted to accurately represent the physical phenomena being simulated. Thus, this approach should almost never be employed. But we note that there is one particular valid application for first-order upwind differencing. We will in Chap. 3 present a detailed algorithm for solving the N.–S. equations in which δ-form quasilinearization will be used to treat nonlinearities appearing in the momentum equations. In this particular instance it is quite convenient, and effective, to utilize first-order upwinding in constructing the left-hand side Jacobian matrix with respect to which the Newton iterations are computed. This is a valid approach provided the right-hand side of the equation is discretized in a physically-correct manner (*i.e.*, different from first-order upwinding) because approximate Jacobian matrices can be useful in Newton iteration procedures, and the one produced via first-order upwinding is numerically very stable.

Second-Order Upwinding. Here we will very briefly treat an approach at one time favored by Shyy [52] and others, namely, use of second-order one-sided differences with switching similar to that employed in first-order upwinding in order to difference into the flow direction. We again employ the conserved-form Burgers' equation, (2.95), as our model problem. Then second-order upwinding takes the form

$$(Uu)_{x,i} = \frac{1}{2h}\begin{cases} 3U_iu_i - 4U_{i-1}u_{i-1} + U_{i-2}u_{i-2}\,, & U_i > 0\,, \\ -U_{i+2}u_{i+2} + 4U_{i+1}u_{i+1} - 3U_iu_i\,, & U_i < 0\,. \end{cases} \tag{2.97}$$

As is well known, this discretization is second-order accurate. Moreover, it turns out that the leading terms of the modified equation are identical to those of the original equation. Thus, no physics has been compromised by this approximation. It can also be checked (by setting $U \equiv$ const. that the roots of the corresponding difference equation are all nonnegative, independent of the value of cell *Re*. Hence, formally, there is no cell-*Re* restriction associated with this approximation. It should also be noted that neither nonnegativity of the difference equation coefficients nor diagonal dominance hold for this scheme, implying that it does not satisfy a maximum principle in the sense described earlier. The lesson to be taken from this is that while satisfaction of these properties is desirable, it is not necessary in general, and more emphasis should be placed on finding the solutions to the difference equation.

Finally, we observe that actual computational experience with second-order upwinding has been rather mixed. Shyy [52] claims it is the best of numerous methods that he tested. But other investigators have shown less favorable results. Furthermore, it is clear that Eq. (2.97) leads to a five-point discrete representation, resulting in a more complicated matrix structure if implicit time stepping is employed in the solution algorithm. Nevertheless, second-order upwinding is often an available option for treating advective terms of the N.–S. equations in commercial CFD codes.

QUICK. The *quadratic upstream interpolation for convective kinematics* scheme of Leonard [48] is one of the most widely-used methods for treating the cell-*Re* problem. It is available as an option in most commercial CFD packages, and primarily for that reason we include it herein. The formula given below corresponds to the same Burgers' equation model problem we have been considering throughout these discussions.

$$(Uu)_{x,i} = \frac{1}{8h}\begin{cases} 3U_{i+1}u_{i+1} + 3U_iu_i - 7U_{i-1}u_{i-1} + U_{i-2}u_{i-2}\,, & U_i > 0\,, \\ -U_{i+2}u_{i+2} + 7U_{i+1}u_{i+1} - 3U_iu_i - 3U_{i-1}u_{i-1}\,, & U_i < 0\,. \end{cases} \tag{2.98}$$

It is worthwhile to note that Leonard originally indicated third-order accuracy for this method, but this was with respect to a location in the grid stencil that did not correspond to any grid point. Careful analysis shows the scheme to be second-order accurate. This confusion likely still persists to the present; at least some commercial flow codes claim to have on option for third-order "upwinding," and it is often a form of QUICK that is being used. It is also useful to note that although this is a switching scheme like the two preceding upwind methods, it is not truly an upwind approach, but rather an "upwind biased" scheme, because its stencil extends to both sides of the grid point in question.

As was true of the second-order upwind method, QUICK does not lead to nonnegativity of difference equation coefficients, nor do corresponding coefficients exhibit diagonal dominance. But unlike the second-order upwinding procedure, QUICK $\underline{\text{does}}$ have a cell-*Re* restriction—and it is nearly as severe as that found for simple centered-difference approximations; *viz.*,

$$|Re_h| \le 8/3\,. \tag{2.99}$$

Nevertheless, computational experience with QUICK has been generally favorable.

There are numerous other upwind-like treatments of the cell-*Re* problem. Some have at times been fairly widely used while most have been rather seldom employed. We do not intend to attempt an encyclopedic coverage of such methods, in large part because all have their shortcomings; and the ones discussed here are most often used. It is worth mentioning, however, that a related treatment has received considerable attention in recent years, and it is usually quite successful. It is to employ techniques originally derived for use in shock capturing associated with solving the Euler equations of compressible flow. There are several classes of these methods, and we merely mention them here: flux-corrected transport (FCT), total variation diminishing (TVD) and essentially non-oscillatory (ENO). While use of such methods can be highly effective in terms of removing non-physical oscillations, they are in general somewhat more difficult to code, and they typically increase required arithmetic very significantly.

In closing this section we provide in Table 2.1 a summary of the methods for treating the cell-Re problem that we have discussed in at least a cursory way. The table contains the independent roots of the difference equation and the cell-Re restriction for the Burgers' equation model problem, if any, for each procedure.

Table 2.1 Difference equation roots and cell-Re restrictions for model problem

Scheme	Roots of Difference Equation	Cell-*Re* Restriction
1st-order upwind	$1+\lvert Re_h\rvert\,,\ 1$	*none*
2nd-order upwind	$\dfrac{\left(1+\frac{3}{2}\lvert Re_h\rvert\right)\pm\sqrt{\left(1+\frac{3}{2}\lvert Re_h\rvert\right)^2-2\lvert Re_h\rvert}}{2}\,,\ 1$	*none*
2nd-order centered	$\dfrac{1+\frac{1}{2}Re_h}{1-\frac{1}{2}Re_h}\,,\ 1$	$\lvert Re_h\rvert\le 2$
QUICK	$\dfrac{\left(1+\frac{3}{4}\lvert Re_h\rvert\right)\pm\sqrt{\left(1+\frac{3}{4}\lvert Re_h\rvert\right)^2-\frac{1}{2}\lvert Re_h\rvert\left(1+\frac{3}{8}\lvert Re_h\rvert\right)}}{2\left(1+\frac{3}{8}\lvert Re_h\rvert\right)}\,,\ 1$	$\lvert Re_h\rvert\le 8/3$

Use of absolute-value symbols in all of the formulas except that for centered differencing arises from the fact that all other techniques involve a switch in formulas so as to always be differencing in the (mainly) flow direction. As a consequence, there are actually two possible difference equations in each such case; but their roots are the same if one expresses them in terms of $|Re_h|$.

2.3.2 Treatment of effects of aliasing

In this section we will first indicate in fairly precise detail just what "aliasing" is, and how it arises. We will see that its symptoms are somewhat similar to those of the cell-Re problem treated above, but its root cause is rather different. We will then consider techniques for treating this problem. It will be seen that since aliasing arises from an inability of a numerical approximation to represent high-wavenumber components in its solution, the remedy must always be some form of smoothing, or mollification, that introduces additional dissipation into the numerical scheme, and thus damps effects of high wavenumbers.

In the context of the N.–S. equations there have been three main approaches used to do this: *i*) flux modification, *ii*) artificial dissipation and *iii*) filtering. We mentioned the first two of these during earlier discussion of the cell-Re problem; both have received considerable attention in the CFD literature, especially in the context of shock capturing in high-speed compressible flow simulations. Here, however, we will emphasize the last possibility. This approach has been utilized mainly in the context of meteorological flows, but Yang and McDonough [53] and McDonough and Yang [54] have recently demonstrated its effectiveness in simulations involving a Burgers' equation model of turbulence. Moreover, applications of this approach to 2-D incompressible N.–S. flows have previously been presented by McDonough *et al.* [55]

and [56], and to 3-D incompressible flows by McDonough and Yang [53]. Similar techniques are employed by Visbal and Gaitonde [58] and others.

How Aliasing Occurs

A straightforward, intuitive view of aliasing comes from the following simple plot, Fig. 2.15, in which we demonstrate an extreme case of under sampling of a signal. The basic signal is a sine function (plus its first subharmonic) with wavelength h, and it is being sampled at this same wavelength, presumably

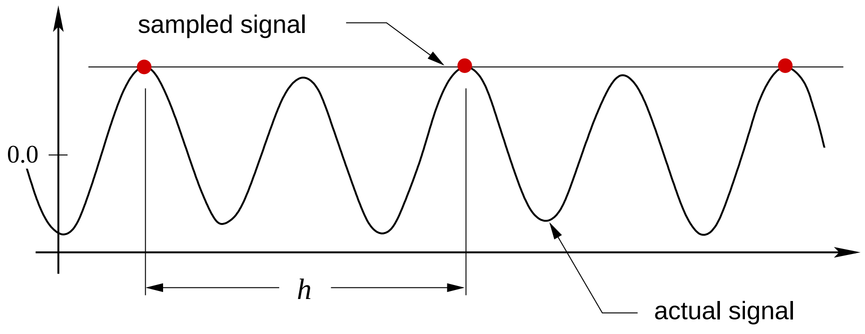

Figure 2.15: Under-sampled oscillatory function demonstrating effects of aliasing.

for later reconstruction. It is evident that the sampled signal is a (nonzero) constant. Moreover, it can also be seen that a Fourier analysis of the actual signal would show that its lowest mode (the average) has zero magnitude while the lowest (and only) mode of the sampled signal has magnitude near unity. We see from this that not only has the sampled signal lost the high-wavenumber content of the original signal, but it has also misrepresented even the lowest wavenumber, which in principle it should have been capable of representing. Thus, what we term as aliasing has two main effects. It results in failure to represent high-wavenumber (and/or high-frequency) behavior, and (as it turns out, a consequence of this) it incorrectly represents lower wavenumbers.

The mathematics associated with this has been known for a long time, especially in the context of signal processing. Here, we provide a description taken from the text by Ames [59]. Let $f(x)$ be a function in $L^2(-1,1)$, and consider its Fourier representation given by

$$f(x) = \sum_{k=-\infty}^{\infty} a_k e^{ik\pi x} \tag{2.100}$$

with

$$a_k = \frac{1}{2}\int_{-1}^{1} f(x) e^{-ik\pi x}\, dx\,.$$

Now if one partitions the interval $[-1,1]$ with $2N$ uniformly-spaced points such that $x_j = j/N$, for $-N \le j < N$, one can construct the (exact) Fourier polynomial which when evaluated at $x = x_j$ gives the value

$$f_j = \sum_{m=-N}^{N-1} A_m e^{im\pi j/N}\,,$$

where

$$A_m = \frac{1}{2N}\sum_{j=-N}^{N-1} f_j e^{-im\pi j/N}\,. \tag{2.101}$$

The basic question at this point is, "How are the A_ms, obtained from the discrete approximation, related to the actual Fourier coefficients, the a_ks?" To find this relationship we evaluate $f(x)$ at the discrete point $x = x_j = j/N$:

$$f(x_j) = \sum_{k=-\infty}^{\infty} a_k e^{ik\pi x_j} = \sum_{k=-\infty}^{\infty} a_k e^{ik\pi j/N} \, .$$

At this point we observe that due to periodicity of the complex exponential there can be only $2N$ distinct values of $e^{ik\pi j/N}$. Then for any finite N we can rewrite the infinite sum as

$$\sum_{k=-\infty}^{\infty} a_k e^{ik\pi j/N} = \sum_{k=-\infty}^{\infty} \sum_{n=-N}^{N-1} a_{n+2Nk} e^{i(n+2Nk)\pi j/N} \, . \tag{2.102}$$

We now substitute the right-hand side of Eq. (2.102) for f_j into Eq. (2.101) to obtain

$$\begin{aligned} A_m &= \frac{1}{2N} \sum_{j=-N}^{N-1} \sum_{k=-\infty}^{\infty} \sum_{n=-N}^{N-1} a_{n+2Nk} e^{i(n+2Nk)\pi j/N} e^{-im\pi j/N} \\ &= \sum_{k=-\infty}^{\infty} a_{m+2Nk} \\ &= a_m + \sum_{|k|>0} a_{m+2Nk} \, , \end{aligned} \tag{2.103}$$

with the second equality following from discrete orthonormality of the complex exponentials.

The first thing to notice regarding this result is that if it happens that only $2N$ Fourier coefficients of a signal are nonzero, the coefficients of the Fourier polynomial will agree with these. Similarly, if the function being represented by the Fourier series and Fourier polynomial is very smooth so that high-wavenumber coefficients in the Fourier series go to zero rapidly as wavenumbers become large, the Fourier polynomial coefficients will not differ greatly from the Fourier series coefficients. On the other hand, if the function f is not very smooth so that high-wavenumber coefficients have significant magnitude, and at the same time N is not sufficiently large, the difference between a_m and A_m will be significant because a_{m+2Nk} will still be large enough, even at very high values of k (and for all m, including $m = 0$), to contribute to the discrepancy. This is the fundamental cause of aliasing.

At this point one might question what any of this has to do with the finite-difference/finite-volume approximations that are the main topic of these lectures. But the answer to this is quite straightforward. It is easily checked that the relationship between the discrete function values f_j and the coefficients A_m of the Fourier polynomial in Eq. (2.101) is a linear transformation consisting of a $2N\times 2N$ matrix. That is, given any set of $2N$ function values we can construct the $2N$ A_ms. Furthermore, the result of computing with a finite-difference or finite-volume method is a grid function consisting of a finite number of values that is supposed to approximate, in some sense, the exact solution to the PDE that has been discretized. We also expect, based on discussions in Chap. 1, that this exact solution possesses a Fourier series representation. Thus, our finite-difference approximation is, to within a linear transformation, a Fourier polynomial intended to approximate the Fourier series of the exact solution. Hence, it is clear that if there are insufficient grid points in the representation, the effect will be identical to under sampling of a signal, and the result will be aliased. Conversely, a fully-resolved, fine-grid solution will not exhibit any effects of aliasing.

It is important to recognize that in the case of attempting to solve the N.–S. equations, this situation is exacerbated by nonlinearity capable of generating high-wavenumber modes—even when they are not present in initial data—in the time evolving solutions of these equations. This combination makes the problem of aliasing very serious, especially for high-Re flows, and it is thus crucial to be able to treat it effectively.

Treatment of the Aliasing Problem

We have already mentioned the three main approaches currently employed to treat aliasing, and we have also noted that independent of the details the end result must be an increase in dissipation in order for a treatment to be effective. As noted earlier, flux-modification schemes can provide a suitable approach despite the fact that they were originally designed to mitigate a rather different problem (Gibbs phenomenon oscillations in the presence of shocks), but one exhibiting similar symptoms. The same can be said for use of artificial dissipation. The main difference between these approaches is in the details of implementation. Flux modification is specifically numerical (*i.e.*, it is applied to the discrete equations), whereas introduction of artificial dissipation can be viewed also in an analytical context. Furthermore, flux-modification methods are relatively difficult to implement while addition of artificial dissipation terms is fairly straightforward, except possibly near boundaries. But it should also be mentioned that the end result of flux modification is introduction of additional truncation error of a diffusive nature. Thus, in principle, given a flux-modification scheme it is possible to find an equivalent artificial dissipation, although this is not generally easy. Finally, we note (as we have already hinted) that considerable programming effort and/or extra floating-point arithmetic is involved with implementing either of these general classes of methods, especially the former.

In the current lectures we will present a different treatment of the aliasing problem. This approach is not new; indeed, it is actually older than flux-modification and cell-Re treatments (that sometimes are mistakenly attempted). It is use of post-processing filters as is routinely done in essentially all signal processing. (After all, a collection of numbers comprising a discrete solution can easily be interpreted as a "signal.") Use of filters in the manner to be described here was probably first introduced by Shuman [25] and analyzed in considerable detail by Shapiro [60]—but not in the context of CFD. Our description herein will be based on the modern treatment of PDEs given in Chap. 1 of these lectures.

We first recall the idea of a solution operator discussed in Chap. 1 and consider a discrete case of this. Recall that a solution operator is a mathematical construct that acts on initial data (prescribed at $t = t_0$) to produce a solution at a later time $t > t_0$. For example, if $u_0(x)$ provides initial data for some PDE and S is its solution operator, then we can formally express the solution at time t as

$$u(t) = S(t)u_0 \,.$$

It is useful for our purposes here to associate $S(t)$ with some form of (temporal) integration over the interval $(t_0, t]$, although this is not the only possible form (recall the solution operator for the heat equation, Eq. (1.17)).

We next observe that discrete solution operators can be easily constructed via formal numerical integration of this form. Consider a nonlinear PDE with the representation

$$u_t = \mathcal{N}(u)\,,$$

where $\mathcal{N}(u)$ represents some, generally, nonlinear (*e.g.*, Navier–Stokes) spatial operator. We can integrate this to obtain

$$u(t) = u(t_0) + \int_{t_0}^{t} \mathcal{N}(u)\, d\tau\,,$$

and replacing the integral on the right-hand side with a quadrature will yield a discrete solution operator. For example, if we were to use simple forward-Euler time integration, we would obtain

$$u^{n+1} = u^n + k\mathcal{N}(u^n)\,,$$

and we can express this formally as

$$u_h^{n+1} = S_h(k)u_h^n \tag{2.104}$$

to indicate that spatial discretization has been done with a grid of spacing h, and discrete time integration is being performed with time step k.

We next recall that one of the tools introduced in Chap. 1 for treating nonclassical solutions to PDEs was use of mollification. This was applied both to initial data, and to solutions as they evolved, by composition with the solution operator. We also noted at that time that mollification is actually just a filtering process. In particular, mollified variables were constructed as

$$u_\epsilon(x) = \int_{-\epsilon}^{\epsilon} u(y)\delta_\epsilon(x-y)\,dy\,, \tag{2.105}$$

where δ_ϵ was a normalized C_0^∞ function with support $\to 0$ as $\epsilon \to 0$. This corresponds to a filter with kernel δ_ϵ. We also observed that a very typical δ_ϵ possessed a graph that did not differ significantly from a triangular shape (except that it is C^∞). We will utilize these ideas here to construct a numerical mollification process that can be composed with discrete solution operators as was done for the continuous case in Chap. 1.

We now define a function δ_h (which is not C^∞ but which does have compact support) as indicated in Fig. 2.16. Notice, in particular, that the support of δ_h is slightly greater than $2h$ about any particular

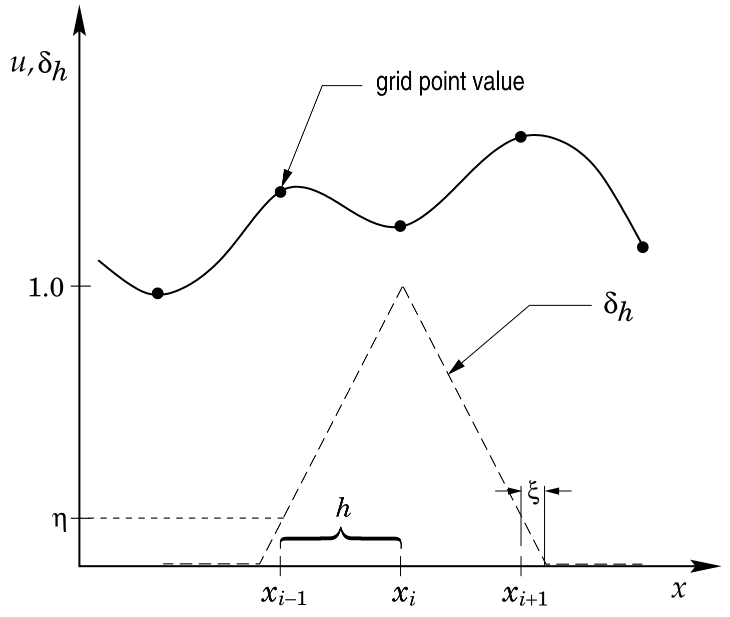

Figure 2.16: Schematic of a discrete mollifier.

point x_i, and that δ_h is constructed from two straight lines. It is easily seen from the figure that for any x_i

$$\delta_h(x_i - h) = \delta_h(x_i + h) = \frac{\xi}{h+\xi} \equiv \eta\,,$$

where ξ is an adjustable parameter, and such that $\xi \ll h$ is assumed to hold. We remark that while this appears very different from δ_ϵ shown in Fig. 1.3, as noted in Majda *et al.* [61], the analytical form of δ_ϵ cannot be effective in the presence of finite grid spacing, and must be modified.

Next we define the discretely-mollified grid function

$$\widetilde{u}_h(x_i) = \int_{x_i-h}^{x_i+h} u_h(y)\delta_h(x_i-y)\,dy \equiv F(\delta_h)u_h\,. \tag{2.106}$$

We can now express the solution given in Eq. (2.104) in terms of the discrete solution operator as

$$u_h^{n+1} = S_h(k)F(\delta_h)u_h^n\,.$$

We first observe that formally integration in (2.106) should be over the interval $[x_i - h - \xi, x_i + h + \xi]$, but there are no grid function values in the border regions of length ξ, so results (numerical) from the above formula would be unchanged by including this. We now perform trapezoidal quadrature between $x_i - h$ and x_i, and again between x_i and $x_i + h$. This results in

$$\begin{aligned}\widetilde{u}_h(x_i) &= \frac{h}{2}\left[2u_h(x_i)\delta_h(x_i - x_i) + u_h(x_{i-1})\delta_h(x_i - x_i + h) + u_h(x_{i+1})\delta_h(x_i - x_i - h)\right] \\ &= \frac{h}{2}\left[\eta u_h(x_{i-1}) + 2u_h(x_i) + \eta u_h(x_{i+1})\right] \\ &= \frac{\eta h}{2}\left[u_h(x_{i-1}) + \frac{2}{\eta}u_h(x_i) + u_h(x_{i+1})\right] . \qquad (2.107)\end{aligned}$$

But for proper normalization of this discrete case we must require $\widetilde{u}_h(x_i) = u_h(x_i)$ if $u_h \equiv$ const. Thus, we must have

$$\widetilde{u}_h(x_i) = \frac{u_h(x_{i-1}) + \beta u_h(x_i) + u_h(x_{i+1})}{2+\beta}\,, \qquad (2.108)$$

with $\beta \equiv 2/\eta$. This is the filter used by Shuman [25], and we call β the *filter parameter.*

It is clear that $\eta \to 0$ as $\xi \to 0$, and in this limit $\beta \to \infty$. This has the same effect as $\epsilon \to 0$ in the definition of mollifier given in Eq. (2.105) and would be equivalent if $h \to 0$ also holds. Furthermore, it is clear from Eq. (2.106), the definition of the discrete mollifier, that $\widetilde{u}_h(x_i) \to u_h(x_i)$ as $h \to 0$. While this is not immediately obvious from the final form of the Shuman filter given in Eq. (2.108), it can be demonstrated via a simple truncation error analysis which we now carry out as follows.

We first express (2.108) in the more concise notation

$$\widetilde{u}_i = \frac{u_{i-1} + \beta u_i + u_{i+1}}{2+\beta}\,, \qquad (2.109)$$

and then expand u_{i-1} and u_{i+1} in Taylor series:

$$u_{i-1} = u_i - hu_x|_i + \frac{h^2}{2}u_{xx}|_i - \frac{h^3}{6}u_{xxx}|_i \pm \cdots\,,$$

and

$$u_{i+1} = u_i + hu_x|_i + \frac{h^2}{2}u_{xx}|_i + \frac{h^3}{6}u_{xxx}|_i + \cdots\,.$$

(We remark here that existence of derivatives needed to construct these expansions is, in general, open to question, especially for the N.–S. equations, and it may be necessary to view these in the sense of distributions, as discussed in Chap. 1.) Then substitution of these into the above yields

$$\widetilde{u}_i = u_i + \frac{h^2}{2+\beta}u_{xx}|_i + \mathcal{O}(h^4)\,. \qquad (2.110)$$

This representation highlights two important features of this filter. First, it demonstrates that u_i is being replaced with a quantity explicitly containing added dissipation—one of the main requirements for successful treatment of aliasing. In particular, it can be seen that the dominant truncation error is diffusive, corresponding to addition of a scaled (discrete inthis case) Laplacian. At the same time, the actual amount of added diffusion is controllable through the parameter β and the grid spacing h. Thus, even though a modified equation would contain extra diffusion at the level of the (physical) second-order operators, this goes to zero with h^2 rather than only as h in first-order upwinding. Moreover, it turns out that the $\mathcal{O}(h^4)$ term is anti-diffusive, leading to some cancellation of the effects at second order.

There are several additional items that should be investigated for the filter given in Eq. (2.109). The first of these is what is usually termed “frequency response” in the context of signal processing (in which the signals are typically functions of time), and which we will here more appropriately for our purposes call *wavenumber response* since we will usually apply the filter spatially. The wavenumber response shows the effect of the filter on the magnitude of the Fourier coefficients as a function of wavenumber.

There are several ways in which this can be derived. Here, we will begin by writing the Fourier coefficient corresponding to each term in the expansion given in Eq. (2.110) but now expressed completely analytically as

$$\widetilde{u} = u + \frac{1}{2+\beta}\left(h^2 u_{xx} + \frac{h^4}{12}u_{xxxx} + \frac{h^6}{360}u_{xxxxxx} + \cdots\right). \tag{2.111}$$

Then it is easily checked that the Fourier coefficients of a filtered and unfiltered function are related as

$$\widetilde{a}_m = \left[1 - \frac{1}{2+\beta}\left(m^2h^2 - \frac{m^4h^4}{12} + \frac{m^6h^6}{360} \mp \cdots\right)\right] a_m\,. \tag{2.112}$$

Furthermore, we see that with $\theta \equiv mh$ the expansion inside the parentheses is just $2(1-\cos\theta)$. Thus, we obtain

$$\widetilde{a}_m = \left[1 - \frac{2}{2+\beta}\left(1-\cos mh\right)\right] a_m\,, \tag{2.113}$$

with scaling such that if there are N grid points, $h = \pi/N$.

It is of interest to observe that as $h \to 0$ (or $N \to \infty$), $\widetilde{a}_m \to a_m\ \forall\ m$, independent of the value of the parameter β. At the same time, for fixed h, as $\beta \to \infty$ we obtain the same result. In addition, we can see the specific wavenumber response for fixed h, as a function of wavenumber, plotted in Fig. 2.17. From the figure we see that for small values of β the high-wavenumber content of the Fourier representation is almost

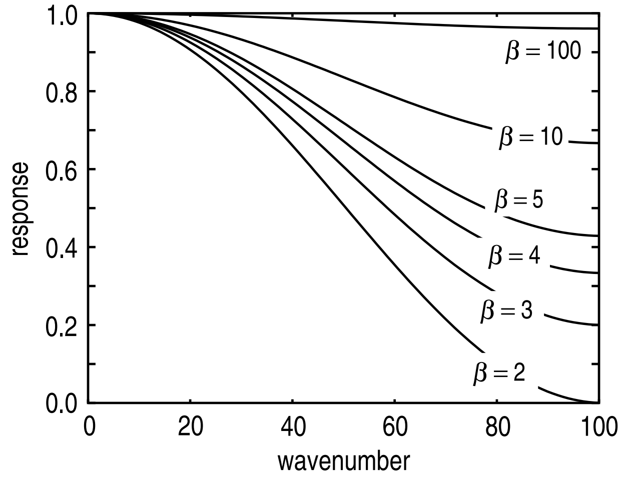

Figure 2.17: Wavenumber response of Shuman filter for various values of filter parameter.

completely removed—exactly what is needed to treat aliasing (recall Eq. (2.103)), and as β is increased less of the original Fourier coefficients is removed by the filter.

The next task is to determine, at least in a qualitative way, the effects of filtering on the numerical solution. We will again employ Burgers' equation, but unlike what was done in our previous analyses we will retain both the time dependence and the nonlinearity because these both are major aspects of the filtering problem, as will be apparent. We remark that in the case of steady solutions, if they are computed directly (as opposed to employing pseudo-time integration) it is possible to apply the filter only once, at the end of the calculation. In this case, the error is given in Eq. (2.110) which indicates second-order accuracy.

The case of time evolution is more complicated. We have already implied that the process involves composition of the filter and the discrete solution operator, so our analysis must reflect this. In particular, especially because of the nonlinearities present in Burgers' equation (and analogous ones in the N.–S.

equations), it is generally necessary to apply the filter after every time step because new high-wavenumber information will have been generated. For simplicity, we consider a simple forward-Euler integration of Burgers' equation; that is, the semi-discrete solution operator takes the form

$$u^{n+1} = u^n + k\left[\nu u^n_{xx} - \left(u^{n\,2}\right)_x\right], \tag{2.114}$$

where k again denotes a time step (Δt).

The first step is to note that because of the composition of discrete solution and filtering operators, the values of u^n on the right-hand side of this equation must be the filtered ones. Thus, from (2.110) we have, to leading order,

$$\begin{aligned} u^{n+1} &= \widetilde{u}^n + k\left[\nu\widetilde{u}^n_{xx} - \left(\widetilde{u}^{n\,2}\right)_x\right] \\ &= u^n + \frac{h^2}{2{+}\beta}u^n_{xx} + k\left[\nu u^n_{xx} + \frac{h^2}{2{+}\beta}u^n_{xxxx} - \left(\left(u^n + \frac{h^2}{2{+}\beta}u^n_{xx}\right)^2\right)_x\right] \\ &= u^n + \frac{h^2}{2{+}\beta}u^n_{xx} + k\left[\nu u^n_{xx} + \frac{h^2}{2{+}\beta}u^n_{xxxx} - \left(u^{n\,2} + \frac{2h^2}{2{+}\beta}u^n u^n_{xx} + \frac{h^4}{(2{+}\beta)^2}u^{n\,2}_{xx}\right)_x\right]. \end{aligned} \tag{2.115}$$

We next regroup terms in powers of the grid spacing h to obtain

$$u^{n+1} = u^n + k\left[\nu u^n_{xx} - \left(u^{n\,2}\right)_x\right] + \frac{h^2}{2{+}\beta}u^n_{xx} + \frac{kh^2}{2{+}\beta}\left[u^n_{xxxx} - 2\left(u^n u^n_{xx}\right)_x\right] - \frac{kh^4}{(2{+}\beta)^2}\left(u^{n\,2}_{xx}\right)_x .$$

This demonstrates the effect of the discrete solution operator on the filtered solution from the previous time step. But to complete the current time step, and thus be ready to repeat the process, we must filter the result at time level $n+1$. This will permit us to see the effects of repeated application of the filter.

Thus, we must consider

$$\begin{aligned} \widetilde{u}^{n+1} = \widetilde{u}^n &+ k\left[\nu\widetilde{u}^n_{xx} - \widetilde{(u^{n\,2})}_x\right] \\ &+ \frac{h^2}{2{+}\beta}\widetilde{u}^n_{xx} + \frac{kh^2}{2{+}\beta}\left(\widetilde{u}^n_{xxxx} - 2\widetilde{(u^n u^n_{xx})}_x\right) - \frac{kh^4}{(2{+}\beta)^2}\widetilde{(u^{n\,2}_{xx})}_x . \end{aligned} \tag{2.116}$$

It should first be observed that the effects of filtering at this point are somewhat different from what we have already analyzed. The preceding analysis merely treated the effects of using a filtered result, but now we must determine specifically what are the effects of filtering in the first place. Clearly, the filter commutes with all of the linear operators; so there is nothing new with regard to any of these terms. But it does not commute with the nonlinear terms, and these must be treated differently.

We first observe that the two filtered nonlinear terms in the second line of (2.116) are higher order, and it is not imperative that we consider them in detail. On the other hand, the nonlinear term in the first line of this equation is at lowest order and must be treated. It is easily shown via usual truncation error analysis that

$$\widetilde{u^2_i} \equiv \frac{1}{2{+}\beta}\left(u^2_{i-1} + \beta u^2_i + u^2_{i+1}\right) = u^2_i + \frac{2h^2}{2{+}\beta}\left(u_i u_{i,x}\right)_x + \mathcal{O}(h^4). \tag{2.117}$$

It then follows that the nonlinear term in the main part of the filtered result from the discrete solution operator (2.116) is

$$\left(\widetilde{u^2}\right)_x = \left(u^2\right)_x + \frac{2h^2}{2{+}\beta}\left(uu_x\right)_{xx} + \mathcal{O}(h^4). \tag{2.118}$$

Thus, as in the case of the linear terms, the leading truncation error occurs at second order in h. But the form of this error is important to note. As observed by Shapiro [60], symmetry of the Shuman filter formula with respect to the discrete point at which it is applied leads to zero phase error in the filtered result—a desirable property. But we can see from the form of the truncation error shown here that now third-order (dispersive) derivatives occur, as do terms associated with nonlinear diffusion. Thus, the qualitative behavior is not easily predicted, but clearly one should expect some generation of phase error upon repeated application of the nonlinear solution operator.

Figure 2.18 displays a comparison of a single time slice from a solution to Burgers' equation having

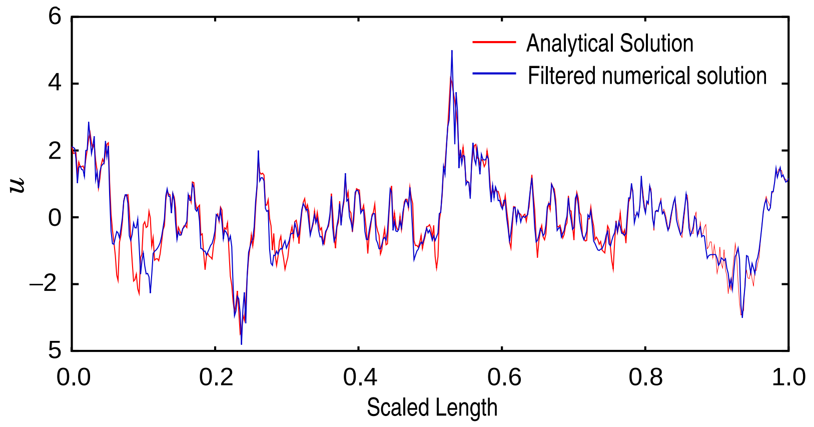

Figure 2.18: Shuman filter applied to solution of Burgers' equation.

the exact solution indicated in the figure (see Yang and McDonough [53]). In attempts to compute this solution directly (without treatment of aliasing) grids of greater than 4096 points were necessary. The grid employed for the calculations displayed here utilizing the filter contained only 512 points. It is clear that the main features of the exact solution have been captured in the coarse-grid filtered solution, but at the same time one can also see occasional significant phase errors (*e.g.*, near $x = 0.1$) as predicted by the preceding analysis.

We now note that introduction of Eq. (2.110), and additional results analogous to (2.117), into the remaining terms of (2.116) will show that with each application of the filter an additional factor of the basic truncation error is added. Thus, after n time steps, the dominant error arising from the filtering process is

$$n\frac{h^2}{2+\beta}u^n_{xx}\,. \tag{2.119}$$

(In fact, this occurs also for other terms in the overall truncation error expansion. We leave demonstration of this as an exercise for the reader.) It is important to realize that if the number of time steps becomes excessive, this term could potentially completely damp essentially all aspects of the computed solution; this is the major disadvantage in this approach. But it is seen from (2.119) that at least in principle this can be controlled with the parameter β, and it is generally found that this is adequate to guarantee robust behavior of this treatment of aliasing.

Now that it has been demonstrated that the Shuman filter is able to accomplish the desired goal, the remaining issue is the cost of the additional required arithmetic. It is well known that flux-modification approaches can easily double the required arithmetic, while use of artificial dissipation is considerably less expensive (and often less effective) if it can be implemented explicitly (which is not always the case). It is clear from Eq. (2.109) that in 1D the Shuman filter requires only two adds and two multiplies per grid point, per application (time step). The forward-Euler discrete solution operator uses five adds and five multiplies per grid point, per time step. Thus, even in the context of this very simple explicit solution operator, application of the filter increases total arithmetic by less than 50%, which is less than the increase in arithmetic incurred by use of typical (explicit) fourth-order artificial dissipation.

The final topic upon which we will touch briefly regarding use of filtering is the multi-dimensional case. All of our analyses have been one-dimensional, but we note that there is a formal, although *ad hoc*, extension to multi-dimensions, which in 2D is

$$\widetilde{u}_{i,j} = \frac{u_{i-1,j} + u_{i,j-1} + \beta u_{i,j} + u_{i,j+1} + u_{i+1,j}}{4+\beta}\,. \tag{2.120}$$

Moreover, an extension of this form to 3D is obvious. We note that all of the preceding development relies on uniform grid spacing in Cartesian coordinates. Little beyond preliminary studies has been completed for more complicated problem domains and non-uniform gridding. Moreover, a rigorous extension of the Shuman filter to higher dimensions is likely to result in a more complex grid stencil than the one corresponding to (2.120).

2.4 Summary

In this chapter we have presented a number of somewhat special aspects of the incompressible Navier–Stokes equations that require more than just a basic knowledge of numerical analysis to successfully treat them. We began the chapter by discussing the various forms of the N.–S. equations that have been employed through the years in efforts to mitigate some of the difficulties that arise in numerical simulations. We observed that of these only the stream function/vorticity and primitive-variable formulations have seen wide application. But in modern 3-D simulations the latter of these is essentially the only viable approach because the former has no direct extension from 2D to 3D. The primitive-variable form, itself, has numerous distinct subforms, all of which are analytically equivalent but which produce different behaviors in numerical implementations.

Following the discussions of N.–S. equation forms we considered the problem of pressure-velocity coupling. We began with fairly detailed descriptions of the three main types of grid structures that have been employed in efforts to achieve this coupling; these are: *i*) (natural) unstaggered, *ii*) partially staggered and *iii*) (fully) staggered. There is also an important modification of the unstaggered grid, often termed nonstaggered, or more appropriately, co-located. It was noted that there is a very fundamental mathematical constraint, known as the div-stability condition, which imposes restrictions on the combination of grid structure and form of discretization of the governing equations that must be used to guarantee that discretely divergence-free velocity fields converge to continuous divergence-free ones as grid spacing is refined. These restrictions are automatically satisfied by centered discretizations on a staggered grid, and with modifications to the discretization of either the advective terms of the momentum equations or treatment of the pressure Poisson equation, or both, they can be satisfied on co-located (cell-centered, unstaggered) grids. We observed that failure to satisfy the div-stability condition can lead to so-called "checkerboard" solutions containing non-physical, and yet very stable, oscillatory behaviors. The natural unstaggered and partially staggered grid configurations are prone to producing such behaviors.

The final topics treated in this chapter, like checkerboarding, are associated with the tendency of discrete solutions to the N.–S. equations to exhibit physically unrealistic oscillations. These can arise from either (or both) of two forms of under resolution, and their treatment has constituted a significant portion of the archival literature of CFD. The first of these, termed the cell-Re problem, is very specifically a property of the difference approximations employed to treat the combination of advection and diffusion terms, as appear in any transport equation—even linear ones. In this linear case it is possible to exactly solve the corresponding difference equations, thus permitting detailed analysis of the cell-Re problem and its origins. We presented such analyses above, and we discussed several proposed remedies. But it must be emphasized that the most important consequence of these analyses is that there is no single modification to the discrete equations that is guaranteed to always be effective for all flow conditions, or sometimes even at all locations of a flow under fixed conditions.

The second topic covered in the last section of this chapter was that of aliasing. We described the basic nature of aliasing—what it is, and how it arises—and provided a specific representation of its effects in terms of a Fourier expansion. We noted that the basic problem arises when discretization step sizes are too large to permit resolution of high-wavenumber components of the true solution. It can be expected from the form of the Fourier representation that the observed consequence of aliasing will be spurious oscillations in a computed solution—the same symptom as seen in the cell-Re problem. It is well known in the field of signal processing that the universal treatment of aliasing is application of a filter to the aliased signal. (Of course, the filter needed for this is not universal, and depends strongly on the type of signal being treated.) Moreover, we saw in Chap. 1 that the analytical treatment of non-smooth solutions (those having significant high-wavenumber content) to PDEs was use of a mollifier, which is in fact a filter. Thus, in the present chapter we derived a discrete form of the typical analytical mollification operator and found that this was precisely the Shuman filter [25]. We then showed in a formal way how this can be applied to treat both cell-Re and aliasing problems. In particular, it was shown how to construct a formal discrete solution operator that employs the filter to smooth the solution at the end of each time step (post processing) before starting the next time step.

Chapter 3

Solution Algorithms for the N.–S. Equations

In this chapter we will present many of the main solution algorithms that have been developed and used through the years in efforts to solve the incompressible Navier–Stokes equations. We will do this by means of a more or less chronological ordering to provide a sense of how the current understanding of solution methods has evolved. For each method treated we will usually provide a fairly complete derivation along with details of implementation, and in most cases a detailed pseudo-language algorithm.

Thus, in the first section we will derive and discuss the marker-and-cell (MAC) method of Harlow and Welch [36]. We follow this with a presentation of the SOLA method. At this point, it turns out that at least two methods were introduced nearly simultaneously: these are artificial compressibility and projection methods. We will treat these in this order because there is a fairly direct relationship between artificial compressibility and SOLA, and also between projection methods and the SIMPLE algorithms that will be the last to be discussed, but are at present the most widely used—especially in commercial CFD software.

Before beginning these treatments we again present the Navier–Stokes equations in the form that they will most often be encountered in the remainder of these lectures; namely

$$u_t + (u^2)_x + (u\,v)_y = -p_x + \nu\Delta u\,, \tag{3.1a}$$

$$v_t + (u\,v)_x + (v^2)_y = -p_y + \nu\Delta v\,, \tag{3.1b}$$

and

$$u_x + v_y = 0\,. \tag{3.2}$$

We observe that the algorithms to be presented in the sequel are of two main types. MAC, SIMPLE and the projection methods employ a pressure Poisson equation to satisfy the divergence-free constraint, Eq. (3.2), while SOLA and the artificial compressibility method attempt to satisfy this condition by directly solving modifications of (3.2). Both approaches have been fairly successful; the latter is widely used in the context of finite-element implementations. On the other hand, essentially all commercial CFD software makes use of the former approach. In what follows we will have the opportunity to compare these two classes of techniques.

3.1 The Marker-and-Cell Method

The marker-and-cell technique was first proposed by Harlow and Welch [36] as a method for solving free-surface problems, but the staggered arrangement of dependent variables proved to be so effective at preventing checkerboard solutions (because it automatically satisfies the div-stability condition) that it has become an important method in its own right. Here, we will put little emphasis on the free-surface aspects of its implementation. We begin with formal discretization of the momentum equations, followed by more detailed treatment of the PPE. We then discuss implementation of boundary conditions for this latter equation and briefly describe computation of marker trajectories. We then conclude with a coarse-grained pseudo-language algorithm suitable for MAC implementation.

3.1.1 Momentum equations

The momentum equations (3.1) are discretized in space in the typical staggered-grid (control-volume) manner presented in Chap. 2 with simple forward-Euler integration employed in time except for pressure-gradient terms which are formally integrated via backward Euler. Thus, the discrete forms of these equations are

$$u_{i,j}^{n+1} = u_{i,j}^n + k\left\{\nu\left(D_{0,x}^2 + D_{0,y}^2\right)u_{i,j}^n - \left[D_{+,x}p_{i,j}^{n+1} + D_{+,x}\left(\overline{u}_{i,j}^{n\,2}\right) + D_{-,y}\left(\widetilde{u}_{i,j}^n\overline{v}_{i,j}^n\right)\right]\right\}\,, \tag{3.3a}$$

$$v_{i,j}^{n+1} = v_{i,j}^n + k\left\{\nu\left(D_{0,x}^2 + D_{0,y}^2\right)v_{i,j}^n - \left[D_{+,y}p_{i,j}^{n+1} + D_{-,x}\left(\widetilde{u}_{i,j}^n\overline{v}_{i,j}^n\right) + D_{+,y}\left(\widetilde{v}_{i,j}^{n\,2}\right)\right]\right\}\,. \tag{3.3b}$$

We note that the formal difference operator notation is to be interpreted in the context of the control-volume analysis of these equations in Chap. 2, as presented in expanded form in the semi-discrete equations (2.61) and (2.62). In particular, the indices correspond to those of grid cells, and not grid points, and as a consequence what we have denoted as forward and backward differences are actually centered across a single grid cell.

3.1.2 The pressure Poisson equation

The distinguishing feature of the MAC method is its treatment of pressure. Recall that the pressure Poisson equation is derived by taking the divergence of the N.–S. equations and then invoking the divergence-free condition, Eq. (3.2) in the present case, to effect a simplification. But as noted in Chap. 2, rather than leading to a pressure field that enforces the divergence-free constraint, this requires that the velocity field already be divergence free in order to calculate a correct pressure. A somewhat different approach is taken in constructing MAC techniques. The first step is the usual one, *viz.*, calculate the divergence of the momentum equations to obtain

$$(u_x + v_y)_t + \left(u^2\right)_{xx} + (u\,v)_{xy} + (u\,v)_{yx} + \left(v^2\right)_{yy} = -\Delta p + \nu\Delta\left(u_x + v_y\right)\,. \tag{3.4}$$

For notational convenience we define

$$D \equiv u_x + v_y\,, \tag{3.5}$$

and write the preceding equation as

$$\Delta p = \nu\Delta D - \left[\left(u^2\right)_{xx} + (u\,v)_{xy} + (u\,v)_{yx} + \left(v^2\right)_{yy}\right] - D_t\,. \tag{3.6}$$

We remark that in contrast to the earlier use of the PPE in Chap. 2, in this case the computed pressure field has the ability to respond to lack of satisfaction of the divergence-free constraint on the velocity field. Nevertheless, it is important to recognize that this alone does not guarantee satisfaction of this constraint. As noted by Roache [1], failure to include the term D_t in (3.6) leads to a nonlinear instability in the momentum equation solution which has been caused by failure to maintain the divergence-free property of the velocity field. On the other hand, omitting the Laplacian of divergence term seems to not necessarily be damaging, especially if ν is very small; successful algorithms have been constructed without this term. The discrete form of Eq. (3.6) is

$$\left(D_{0,x}^2 + D_{0,y}^2\right)p_{i,j}^{n+1} = \nu\left(D_{0,x}^2 + D_{0,y}^2\right)D_{i,j}^n - \left[D_{0,x}^2\left(\overline{u}_{i,j}^2\right) + 2D_{0,x}D_{0,y}\left(\widetilde{u}_{i,j}\overline{v}_{i,j}\right) + D_{0,y}^2\left(\widetilde{v}_{i,j}^2\right)\right]^n - \left(\frac{D_{i,j}^{n+1} - D_{i,j}^n}{k}\right)\,, \tag{3.7}$$

where $\overline{u}_{i,j}$, $\overline{v}_{i,j}$, $\widetilde{u}_{i,j}$ and $\widetilde{v}_{i,j}$ were defined in Eqs. (2.62) which we repeat here for easy reference:

$$\overline{u}_{i,j} = \frac{1}{2}(u_{i,j} + u_{i-1,j})\,, \tag{3.8a}$$

$$\overline{v}_{i,j} = \frac{1}{2}(v_{i,j} + v_{i+1,j})\,, \tag{3.8b}$$

$$\widetilde{u}_{i,j} = \frac{1}{2}(u_{i,j} + u_{i,j+1})\,, \tag{3.8c}$$

$$\widetilde{v}_{i,j} = \frac{1}{2}(v_{i,j} + v_{i,j-1})\,. \tag{3.8d}$$

We note that the divergence, D, is approximated in the manner described in Chap. 2, namely,

$$D_{i,j} = \frac{1}{h_x}(u_{i,j} - ui - 1, j) + \frac{1}{h_y}(v_{i,j} - v_{i,j-1})\,.$$

Now because we want the divergence of the velocity field to be zero at the end of time step $n+1$ we formally set $D^{n+1} = 0$ in Eq. (3.7) and write this as

$$\left(D_{0,x}^2 + D_{0,y}^2\right) p_{i,j}^{n+1} = \nu \left(D_{0,x}^2 + D_{0,y}^2\right) D_{i,j}^n \;-\; \Big[D_{0,x}^2\left(\overline{u}_{i,j}^2\right) + 2D_{0,x}D_{0,y}\left(\widetilde{u}_{i,j}\,\overline{v}_{i,j}\right)$$
$$+\; D_{0,y}^2\left(\widetilde{v}_{i,j}^2\right)\Big]^n + \frac{1}{k}D_{i,j}^n\,. \tag{3.9}$$

It is important to note the time levels used for the various terms in this equation. The velocities (and divergences) are all evaluated at time level n to produce pressure at time level $n+1$ consistent with forward-Euler integration used for the momentum equations. We remark that because D^{n+1} has been set to zero, this procedure is explicit—no iteration between pressure and momentum equations is needed within a time step. But since p^{n+1} is now available for use in these equations, backward Euler is (formally) employed for the pressure gradient terms, as indicated in Eqs. (3.3).

The new aspect of MAC that was not treated earlier in Chap. 2 when staggered gridding was introduced is the right-hand side of the PPE. Here we will first consider this away from boundaries at an arbitrary interior grid cell. We begin by recalling that divergence, like all other scalars, is calculated at the cell centers, *i.e.*, at the same locations as is the pressure. Furthermore $D_{i,j}$, itself, is easily calculated at all interior cell centers. This implies that away from boundaries we can easily implement the approximation

$$\left(D_{0,x}^2 + D_{0,y}^2\right) D_{i,j} = \frac{1}{h_x^2}\left(D_{i-1,j} - 2D_{i,j} + D_{i+1,j}\right) + \frac{1}{h_y^2}\left(D_{i,j-1} - 2D_{i,j} + D_{i,j+1}\right) \tag{3.10}$$

by first calculating the $D_{i,j}$s at all cell centers and then directly evaluating this expression at each required location.

Next we consider the remaining terms on the right-hand side of Eq. (3.9). We observe that $\overline{u}_{i,j}$ and $\widetilde{v}_{i,j}$ are also cell-centered quantities, implying that

$$D_{0,x}^2\overline{u}_{i,j}^2 = \frac{1}{h_x^2}\left(\overline{u}_{i-1,j}^2 - 2\overline{u}_{i,j}^2 + \overline{u}_{i+1,j}^2\right)\,, \tag{3.11}$$

and

$$D_{0,y}^2\widetilde{v}_{i,j}^2 = \frac{1}{h_y^2}\left(\widetilde{v}_{i,j-1}^2 - 2\widetilde{v}_{i,j}^2 + \widetilde{v}_{i,j+1}^2\right)\,. \tag{3.12}$$

What remains are approximations of the mixed derivatives of products of the various averaged velocity components. Figure 3.1 displays the various averages as defined in Eqs. (3.8) that are needed for these calculations. First consider $D_{0,x}D_{0,y}\left(\widetilde{u}_{i,j}\overline{v}_{i,j}\right)$. We see from the figure that $\widetilde{u}_{i,j}$ and $\overline{v}_{i,j}$ are both defined on the natural finite-difference grid points occurring at the corners of the grid cell. At the same time, it is clear

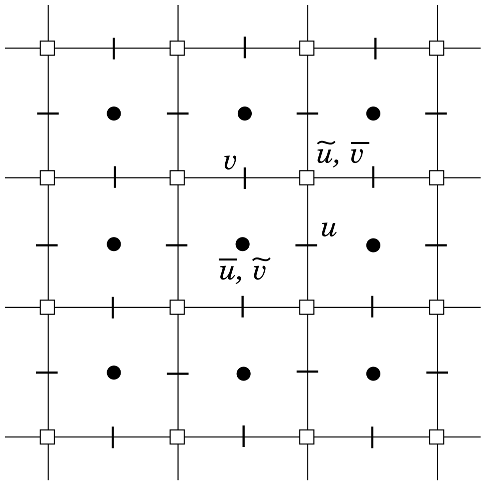

Figure 3.1: Location of averaged velocity components needed for mixed derivative approximations.

that the desired mixed derivative approximation must be at the cell center because this is where pressure is defined. We can construct this as a centered difference approximation with indexing that associates the upper right-hand finite-difference grid point of a cell with the cell index (i, j), using grid spacings h_x and h_y. Thus, we obtain

$$D_{0,x}D_{0,y}\left(\widetilde{u}\,\overline{v}\right)_{i,j} = \frac{1}{h_x h_y}\left((\widetilde{u}\,\overline{v})_{i,j} - (\widetilde{u}\,\overline{v})_{i-1,j} - (\widetilde{u}\,\overline{v})_{i,j-1} + (\widetilde{u}\,\overline{v})_{i-1,j-1}\right) . \tag{3.13}$$

This completes discretization of the right-hand side of the MAC pressure poisson equation at interior grid points.

3.1.3 Implementation near boundaries

We have previously given a thorough treatment of the usual boundary conditions for velocity and pressure on a staggered grid in Chap. 2. But further treatment is needed here for the right-hand side of the PPE. In particular, because of the details of staggered-grid discretizations, if the right-hand side function contains any derivative approximations, information on, or beyond, the adjacent boundary will be needed. In this section we will show how each of the terms in the right-hand side of (3.9) is treated at the boundaries of the domain, analogous to what we have already done for interior locations. We will do this at a corner of the domain; this represents the severest case because it demands boundary treatments in two directions simultaneously. Away from corners the treatment will be similar, but only one direction will require special attention. Figure 3.2 provides a modification of Fig. 3.1 that emphasizes the lower right-hand corner of the boundary region of the computational domain.

We begin by observing that the first term on the right-hand side of Eq. (3.9) is approximated exactly as in Eq. (3.10); that is

$$\left(D_{0,x}^2 + D_{0,y}^2\right) D_{N_x,2} = \frac{1}{h_x^2}\left(D_{N_x-1,2} - 2D_{N_x,2} + D_{N_x+1,2}\right) + \frac{1}{h_y^2}\left(D_{N_x,1} - 2D_{N_x,2} + D_{N_x,3}\right) . \tag{3.14}$$

This is made possible by the fact that the divergence-free condition is required to hold on all boundaries (because it is needed to set the pressure—which, formally, must be calculated on all boundaries on which Neumann conditions are imposed), and this implies that, *e.g.*, $D_{N_x+1,2} = -D_{N_x,2}$ and $D_{N_x,1} = -D_{N_x,2}$. The remainder of the points appearing in the above approximation are interior grid point values of $D_{i,j}$ and so need no special treatment.

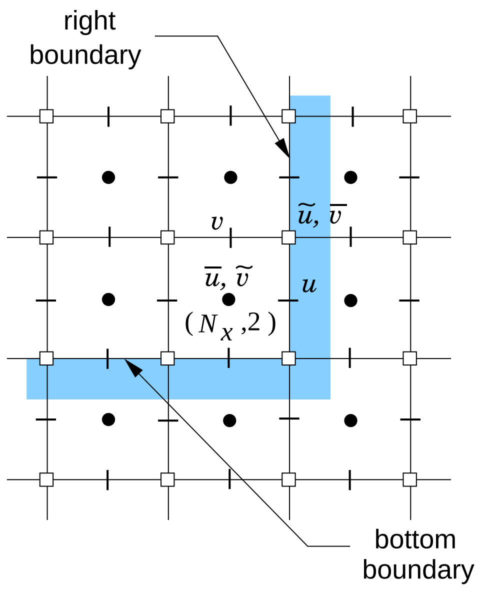

Figure 3.2: Near-boundary grid cells for treatment of right-hand side of Eq. (3.9).

Next we consider Eq. (3.11) evaluated at the point $(N_x, 2)$. This now takes the form

$$D^2_{0,x}\overline{u}^2_{N_x,2} = \frac{1}{h_x^2}\left(\overline{u}^2_{N_x-1,2} - 2\overline{u}^2_{N_x,2} + \overline{u}^2_{N_x+1,2}\right) ,$$

where only the term $\overline{u}^2_{N_x+1,2}$ needs any special attention. But this value occurs at the cell center in the image cell to the right of the interior point $(N_x, 2)$. The u-component of velocity on the vertical boundary of this cell must satisfy a zero mass flux condition because of the solid wall. As a consequence, we must have $\overline{u}^2_{N_x+1,2} = -\overline{u}^2_{N_x,2}$, and this completes evaluation of this approximation. The approximation of $D^2_{0,y}\widetilde{v}^2_{i,j}$ corresponding to Eq. (3.12) is done in a completely analogous way, using the same reasoning.

Similarly, approximation of the first mixed-derivative term, $D_{0,x}D_{0,y}\left(\widetilde{u}\,\overline{v}\right)_{i,j}$, given in Eq. (3.13), is completely trivial. Three of its four entries on the right-hand side are products of velocity components residing on physical (solid) boundaries, and the remaining entry is an interior point. Hence, no special treatment is needed.

3.1.4 Marker particle trajectories for free surface tracking

The final item we treat with regard to the MAC method is its ability to predict the location of free surfaces. This is accomplished by introducing massless marker particles in the flow and following their Lagrangian trajectories as the flow field evolves in time. Figure 3.3 provides an indication of how this is done in practice. In particular, at the time marker particles are first introduced it is common to place four or five markers in each grid cell, as noted in [1]. Because these are massless, they follow the velocity field to produce Lagrangian *pathlines*, and this is implemented by noting that the (x, y) position of a particle is given by

$$\frac{dx_p}{dt} = u\,, \qquad \frac{dy_p}{dt} = v\,, \tag{3.15}$$

or

$$x_p^{n+1} = x_p^n + \int_{t^n}^{t^{n+1}} u\,dt\,, \tag{3.16a}$$

$$y_p^{n+1} = y_p^n + \int_{t^n}^{t^{n+1}} v\,dt\,. \tag{3.16b}$$

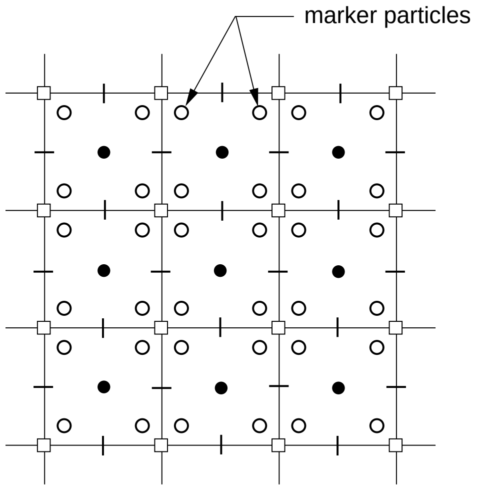

Figure 3.3: Introduction of massless marker particles.

Forward-Euler integration, *e.g.*, then leads to

$$x_p^{n+1} = x_p^n + ku_p \,, \tag{3.17a}$$

$$y_p^{n+1} = y_p^n + kv_p \,, \tag{3.17b}$$

where u_p and v_p represent appropriately interpolated velocity components at the location (x_p, y_p), as discussed next, and k is the time step used to integrate the momentum equations.

Clearly, there are two main problems associated with this approach. The first is interpolation of the velocity field at each time step to provide computed results at correct Lagrangian locations so that the numerical integration can be conducted. It has been typical to employ multi-linear interpolation, but obviously for complicated flow behavior this would prove to be rather inaccurate except on very fine grids. Thus, higher-order methods might be desirable, including use of cubic splines. It should also be noted that on highly non-orthogonal generalized-coordinate grids it may be necessary to employ contravariant velocity components

The second problem is the numerical integration itself. Here, we have employed simple forward-Euler integration as was done in the original presentation of the MAC method in [36]. This is consistent with the order of accuracy of the velocity field simulation. But, especially if a higher-order integration scheme is used to solve the momentum equations, as will be done later for other techniques (and could be done here, as well), then it would be desirable to employ more accurate integrations to obtain the particle trajectories. Another approach is to use multiple time steps of smaller size than the momentum equation time steps to produce smoother particle trajectories.

It should be clear that the initial placement of marker particles indicated in Fig. 3.3 is rather arbitrary, as is the number of particles per grid cell. Handling details such as these is generally left to the specific implementation. A related problem is introduction of new particles at later times in situations that result in some (or even all) of the original particles exiting the computational domain. There are numerous cases to be dealt with in this regard, and methods of treatment tend to be *ad hoc.* Thus, we will not provide further discussions here.

Finally, we mention that despite the fact that the MAC method is intended for treating free surfaces, no effects of surface tension have been incorporated in this basic formulation. Correspondingly, no specific effort is made to guarantee satisfaction of free-surface kinematic conditions. Despite all this, MAC has been a quite successful tool, both for simulating free-surface flows, and for incompressible flow in general—and it was the first method to have this capability. Nevertheless, there are by now more modern (and better) approaches, and we will discuss these in the sequel. Before doing that, however, we will end this section with a presentation of the formal MAC algorithm and some closing discussions associated with it.

3.1.5 The MAC algorithm

In this final subsection we present a coarse-grained pseudo-language algorithm that summarizes the essential parts of the MAC approach.

Algorithm 3.1 *(Marker and Cell) Suppose n time steps have been completed. To advance the solution to time level $n+1$, perform the following steps:*

1. *load right-hand side of discrete PPE* (3.9) *using time level n velocity results to calculate divergences and other terms;*

2. *solve the PPE for time level $n+1$ pressure using any appropriate elliptic solver;*

3. *apply explicit Euler integration to advance the momentum equations to time level $n+1$ using time level n information for all terms except those corresponding to pressure gradients;*

4. *if marker particles are active, perform required interpolations and ODE integrations of Eqs.* (3.16) *to advance particle locations to the $n+1$ time level.*

It should be noticed that because forward-Euler (explicit) integration is used in the momentum equations, the time stepping is only conditionally stable. In particular, if *Re* is small, we expect the time step to be limited by diffusive stability requirements similar to those needed for the heat equation. On the other hand, if *Re* is relatively large we would expect that a CFL-type condition would be required. Results presented in Fletcher [31] confirm the first of these, but a somewhat different requirement is found in the second case. According to [31], linear stability analyses carried out by Peyret and Taylor [62] show that the time step size for a 2-D problem is limited by

$$k \le \min\left[\left(\frac{1}{4}Reh^2\right), \left(\frac{4/Re}{\max_{i,j}\left(|u_{i,j}|+|v_{i,j}|\right)^2}\right)\right]. \tag{3.18}$$

This stability condition is quite restrictive, and in recent years the trend has been toward use of implicit (and mixed) schemes in the present context.

Finally, it must be cautioned that although the MAC algorithm formally leads to satisfaction of the divergence-free constraint, it does this indirectly through details of the PPE; so fairly tight convergence tolerances must be employed for the PPE iterations to be sure the pressures computed from Eq. (3.9) actually correspond to a mass-conserved velocity field. Moreover, the approximation used for the time derivative of the divergence on the right-hand side of this equation introduces further error with respect to satisfying the divergence-free constraint, and the velocity field at time level $n+1$ is calculated only from the momentum equation, thus introducing additional divergence. Finally, if we recall the theoretical analyses of the N.–S. equations from Chap. 1, we recognize that in the context of the MAC algorithm it is essential that the initial velocity field be very accurately divergence free. On the other hand, consistent with well-posedness requirements for the N.–S. equations, it is not necessary to specify initial data for the pressure field.

3.2 SOLA Method

The SOLA algorithms were originally developed at the Los Alamos National Laboratories as a simplification of the MAC methods. Solution of the PPE at each time step in MAC procedures was considered overly expensive, and in addition it was recognized that for many flows there was not any real need to track the Lagrangian marker particles. Ultimately, versions of SOLA were developed to treat free surfaces in a rather different way (the "volume-of-fluid," VOF approach) from that done in MAC methods. In the present discussions, however, we will consider only the simplest form of the algorithm. With only a few minor modifications, we will basically follow the original treatment of Hirt *et al.* [63].

We begin by recalling that there are two main classes of approaches to satisfying the divergence-free constraint associated with solving the incompressible N.–S. equations: *i*) direct solution of the continuity equation, and *ii*) use of a PPE. The MAC method treated in the preceding section is an example of the latter, and SOLA is based on the former. This is accomplished by constructing a fixed-point iteration of the continuity equation as follows.

As we have noted previously, although the pressure does not appear explicitly in the divergence of the velocity field, it is certainly present implicitly since the velocity components are calculated from the momentum equations which contain the pressure gradient. Thus, we can formally express the solenoidal condition as

$$D(p) = 0\,, \tag{3.19}$$

where D represents the divergence as in the preceding section.

Now if we were to solve this locally in a single grid cell via Newton iterations we would obtain

$$p^{(m+1)}_{i,j} = p^{(m)}_{i,j} - \frac{D_{i,j}}{\partial D_{i,j}/\partial p_{i,j}}\,, \tag{3.20}$$

where the superscript (m) is an iteration counter. We next calculate $\partial D_{i,j}/\partial p_{i,j}$ using the discrete form of the momentum equations substituted into the divergence. Since SOLA employs forward-Euler time integration, as used in MAC methods, the discrete momentum equations take the form

$$u^{n+1}_{i,j} = u^n_{i,j} + k\left\{\nu\left(D^2_{0,x} + D^2_{0,y}\right)u^n_{i,j} - \left[D_{+,x}p_{i,j} + D_{+,x}\left(\overline{u}^{n\,2}_{i,j}\right) + D_{-,y}\left(\widetilde{u}^n_{i,j}\overline{v}^n_{i,j}\right)\right]\right\}\,, \tag{3.21a}$$

$$v^{n+1}_{i,j} = v^n_{i,j} + k\left\{\nu\left(D^2_{0,x} + D^2_{0,y}\right)v^n_{i,j} - \left[D_{+,y}p_{i,j} + D_{-,x}\left(\widetilde{u}^n_{i,j}\overline{v}^n_{i,j}\right) + D_{+,y}\left(\widetilde{v}^{n\,2}_{i,j}\right)\right]\right\}\,. \tag{3.21b}$$

These equations are the same as Eqs. (3.3) used in the MAC method except we have suppressed the time level indexing on pressure. It will be seen that in the context of the SOLA algorithm these equations provide only estimates of the velocity field, and these are, in turn, corrected in a manner that will later be seen to be closely related to Leray projection.

We now differentiate Eq. (3.21a) with respect to x, thus obtaining an expression for the first term in the discrete divergence $D_{i,j}$, and then we differentiate the result with respect to $p_{i,j}$. This leads to

$$\begin{aligned}\frac{\partial}{\partial p_{i,j}}u^{n+1}_{x,i,j} &\simeq -\frac{\partial}{\partial p_{i,j}}\left(D_{-,x}D_{+,x}p_{i,j}\right)\\ &= \frac{2k}{h^2}\,.\end{aligned} \tag{3.22}$$

Exactly the same result is obtained by differentiating (3.21b) with respect to y and again differentiating with respect to $p_{i,j}$. Thus, we can combine these to obtain

$$\frac{\partial D_{i,j}}{\partial p_{i,j}} = \frac{4k}{h^2}\,, \tag{3.23}$$

and it follows that (3.20) can be written as

$$p^{(m+1)}_{i,j} = p^{(m)}_{i,j} - \frac{h^2}{4k}D_{i,j}\,, \tag{3.24}$$

or

$$\delta p_{i,j} = -\frac{h^2}{4k}D_{i,j}\,. \tag{3.25}$$

We next observe that because discretization of the momentum equations is explicit, a change in pressure as given in (3.25) will be the only direct effect leading to a change in velocity. In particular, replacing $u_{i,j}$ with $u_{i,j} + \delta u_{i,j}$ and $p_{i,j}$ with $p_{i,j} + \delta p_{i,j}$ in Eq. (3.21a) shows that

$$\delta u_{i,j} = \frac{k}{h}\delta p_{i,j} \,,$$

and thus

$$u_{i,j}^{n+1\,(m+1)} = u_{i,j}^{n+1\,(m)} + \frac{k}{h}\delta p_{i,j} \,. \tag{3.26}$$

We obtain a completely analogous result from the y-momentum equation:

$$v_{i,j}^{n+1\,(m+1)} = v_{i,j}^{n+1\,(m)} + \frac{k}{h}\delta p_{i,j} \,. \tag{3.27}$$

Now note that the iterations indicated in Eqs. (3.24) through (3.27) are intended to achieve mass conservation; *viz.*, $\delta p_{i,j} \to 0$ as $D_{i,j} \to 0$. But this implies that whenever $u_{i,j}$ and $v_{i,j}$ are modified, it will be necessary to also adjust $u_{i-1,j}$ and $v_{i,j-1}$ in the same grid cell in order to satisfy the divergence-free constraint for the entire grid cell. Thus, we also perform the following calculations:

$$u_{i-1,j}^{n+1\,(m+1)} = u_{i-1,j}^{n+1\,(m)} - \frac{k}{h}\delta p_{i,j} \,, \tag{3.28}$$

and

$$v_{i,j-1}^{n+1\,(m+1)} = v_{i,j-1}^{n+1\,(m)} - \frac{k}{h}\delta p_{i,j} \,. \tag{3.29}$$

The minus signs on $\delta p_{i,j}$ in this second set of formulas should be noted. These arise directly from the requirement that $D_{i,j} = 0$ in each grid cell when evaluated with the <u>new</u> iterate. In particular, observe that for uniform grid spacing that is equal in both directions we have

$$D_{i,j}^{(m)} = \frac{1}{h}\left(u_{i,j}^{(m)} - u_{i-1,j}^{(m)} + v_{i,j}^{(m)} - v_{i,j-1}^{(m)}\right) \,. \tag{3.30}$$

If one solves each of Eqs. (3.26) through (3.29) for the value at the m^{th} iterate and substitutes the result into (3.30), one obtains

$$D_{i,j}^{(m)} = D_{i,j}^{(m+1)} - \frac{4k}{h^2}\delta p_{i,j} \,.$$

But we want the divergence to be zero at the advanced iteration; that is, $D_{i,j}^{(m+1)} = 0$ should hold. This implies that

$$D_{i,j}^{(m)} = -\frac{4k}{h^2}\delta p_{i,j} \,,$$

which is identical to (3.25) after rearrangement. Clearly, this can only be achieved with the combination of signs employed in the above equations. Indeed, a more intuitive sign combination would lead to the useless result $D_{i,j}^{(m)} = 0$.

In executing the SOLA algorithm the preceding equations, (3.24) through (3.29) are evaluated successively for each grid cell. This represents a Gauss–Seidel iteration, suggesting that introduction of a relaxation parameter, ω, might be of value in speeding convergence. In SOLA this is done by replacing (3.25) with

$$\delta p_{i,j} = -\frac{\omega h^2}{4k} D_{i,j}^{(m)} \,. \tag{3.31}$$

In [63] it is recommended that ω be selected from $0 < \omega < 2$, as would be the case for successive overrelaxation applied directly to the PPE; but we know from the theory of linear fixed-point iteration that, in general, ω should be increased as h is decreased. In addition we would expect that ω might decrease as k is decreased if we assume there is an optimal value of the parameter

$$C \equiv -\frac{\omega h^2}{4k} \tag{3.32}$$

corresponding to a minimum of the spectral radius of the iteration matrix that arises from this iteration process. We also note that because the point iteration scheme constructed with the preceding equations is so similar to SOR, it would be reasonable to consider various other forms such as line SOR and red-black ordering.

We are now prepared to provide a pseudo-language algorithm by means of which SOLA can be implemented.

Algorithm 3.2 *(SOLA) Suppose n time steps have been completed. Then velocities and pressures for time step $n+1$ can be calculated as follows:*

1. *Estimate $\left\{u_{i,j}^{n+1}\right\}$ and $\left\{v_{i,j}^{n+1}\right\}$; using a forward-Euler discretization of the momentum equations with the time-level n pressure, Eqs.* (3.21)*.*

2. *Compute a mass-conserved velocity field, and the corresponding pressure, by carrying out the iterations corresponding to Eqs.* (3.24) *through* (3.29)*.*

This algorithm is quite straightforward and easily coded. Boundary condition implementation is the same as for any other staggered-grid algorithm, so we will not present any further details. In contrast to the MAC method, the divergence-free condition is explicitly satisfied at the end of each time step; but the manner in which this is done is such as to potentially cause considerable error in the discrete momentum equations. So although in principle one might expect to be able to employ SOLA for time-accurate simulations, this has not been very successful. In particular, besides the error in the momentum balances induced by this approach, the iteration procedure is actually no faster than very primitive procedures for solution of the Poisson equation for pressure. Thus, the original goal of improving the efficiency of MAC methods has not actually been realized with SOLA.

3.3 Artificial Compressibility

Of the methods treated so far in these lectures, artificial compressibility is probably the most widely-used. It is a second example of an attempt to solve the continuity equation (in contrast to solving a pressure Poisson equation) as was SOLA, and we will see that it is in some respects very similar to SOLA. The method of artificial compressibility was originated by Chorin [64] and [65] and was developed into a practical method by Kwak and coworkers (see, *e.g.*, Kwak *et al.* [66]). The first work by Chorin actually preceded development of SOLA by nearly a decade, but artificial compressibility did not become well known or widely used until the work by Kwak. Here, we will very briefly outline the early approach of Chorin and then discuss changes made by Kwak in producing the NASA code known as INS3D which is documented in the report by Rogers *et al.* [67]. We will follow this with some more recent updates due to Soh [68].

In the original development of the artificial compressibility method Chorin [64] and [65] starts with the usual incompressible N.–S. equations in unconserved form but replaces the divergence-free condition with

$$\rho_t + u_x + v_y = 0\,, \tag{3.33}$$

and defines an "artificial compressibility" by

$$\delta_a \equiv \frac{\rho}{p}\,, \tag{3.34}$$

where ρ is density, and p is pressure. Recall from basic fluid mechanics that *compressibility* is defined as

$$\delta \equiv \frac{\partial \rho}{\partial p} \,. \tag{3.35}$$

It should be observed that although Eq. (3.33) is intended to appear as the compressible continuity equation, it is, in fact, quite different. (Indeed, it is not even dimensionally consistent.) Furthermore, it is easy to see that the divergence-free condition will be recovered only as $\rho_t \rightarrow 0$, so this is intrinsically a steady-state approach.

In [65] Eq. (3.34) is solved for density in terms of pressure, and the result is substituted into the momentum equations while taking δ_a to be constant. Then (3.33) is discretized via centered differences in both space and time (except adjacent to boundaries, where one-sided spatial discretizations are employed). The momentum equations are approximated with the Dufort–Frankel scheme (see [20]), and all calculations are performed on an unstaggered grid. The solution is marched forward in (pseudo) time until

$$\frac{\partial \rho}{\partial t} = \delta_a \frac{\partial p}{\partial t} = 0 \,.$$

At this point the velocity field is divergence free, and the discrete momentum equations have also been satisfied. But it is important to note, as hinted above, that this scheme is only consistent with the steady form of the N.–S. equations. Chorin [65] views δ as a relaxation parameter similar to those used in SOR schemes, and in addition he employs red-black ordering in his solution procedure.

Kwak *et al.* [66] present a fully-implicit version of artificial compressibility, also employing an unstaggered grid. They replace Eq. (3.33) with

$$\frac{\partial p}{\partial t} + \frac{1}{\delta_a}(u_x + v_y) = 0 \,, \tag{3.36}$$

and they provide the following bounds on the artificial compressibility:

$$\mathcal{O}(h) \lesssim \delta_a \ll \frac{1}{\left[1 + \frac{4}{Re}\left(\frac{L}{L_\omega}\right)^2 \left(\frac{L_p}{L}\right)\right]^2 - 1} \,. \tag{3.37}$$

As discussed in [66], the left-hand inequality arises from stiffness considerations associated with typical solution procedures. In the right-hand inequality L is a characteristic length used to define the Reynolds number, and L_ω and L_p are lengths over which vorticity and pressure, respectively, propagate in a single time step.

In [66] a generalized-coordinate formulation of the N.–S. equations is utilized with trapezoidal time integration and centered spatial differencing. Because this combination of discretization methods has no dissipation, a fourth-order artificial dissipation term is employed. The implicit formulation is fully coupled and solved at each time step using δ-form Douglas and Gunn time splitting; however, quasilinearization is not employed. Also, the fourth-order dissipation is replaced with a second-order term on the left-hand side of the discrete equations to preserve the block tridiagonal matrix structure. The calculations are marched forward in (pseudo) time until a steady state is reached, as was the case with Chorin's implementation.

Artificial compressibility has also been implemented on staggered grids, for example by Soh [68]; we will provide a modification of this treatment herein. We note from the start that we will consider computing only steady solutions. The nature of the basic formulation is such that the time scales of temporal derivatives in the continuity and momentum equations would need to be very different in order for unsteady calculations to be performed correctly, and this is quite inefficient. Thus we express the incompressible N.–S. equations as

$$u_\tau + \left(u^2\right)_x + (uv)_y = -p_x + \nu\Delta u\,, \tag{3.38a}$$

$$v_\tau + (uv)_x + \left(v^2\right)_y = -p_x + \nu\Delta u\,, \tag{3.38b}$$

$$p_\tau + \frac{1}{\delta_a}(u_x + v_y) = 0\,, \tag{3.38c}$$

where τ is pseudo time.

We will employ the control-volume centered-difference approximations as done previously because we are again working with a staggered grid, and because only steady solutions are to be considered we will implement a backward-Euler temporal integration procedure with linearization about the previous time step in the momentum equations. The continuity equation will be discretized with forward Euler in time and centered (with respect to a single cell) differencing in space. Thus, the system of fully-discrete equations takes the form

$$\left\{1 - k\left[\nu\left(D^2_{0,x} + D^2_{0,y}\right) - D_{+,x}\left(\overline{u}^n_{i,j}\,\cdot\right) - D_{-,y}\left(\overline{v}^n_{i,j}\,\cdot\right)\right]\right\} u^{n+1}_{i,j} = u^n_{i,j} - kD_{+,x}p_{i,j}\,, \tag{3.39a}$$

$$\left\{1 - k\left[\nu\left(D^2_{0,x} + D^2_{0,y}\right) - D_{+,x}\left(\widetilde{u}^n_{i,j}\,\cdot\right) - D_{-,y}\left(\widetilde{v}^n_{i,j}\,\cdot\right)\right]\right\} v^{n+1}_{i,j} = v^n_{i,j} - kD_{+,y}p_{i,j}\,, \tag{3.39b}$$

$$p^{n+1}_{i,j} = p^n_{i,j} - \frac{k}{\delta_a h}\left(D_{-,x}u_{i,j} + D_{-,y}v_{i,j}\right)\ . \tag{3.39c}$$

In these equations the notation $D_{+,x}\left(\overline{u}^n_{i,j}\,\cdot\right)$, *e.g.*, has the following interpretation:

$$\begin{aligned}
D_{+,x}\left(\overline{u}^n_{i,j}\,\cdot\right) &= D_{+,x}\left(\overline{u}^n_{i,j}\,u^{n+1}_{i,j}\right)\\
&\to D_{+,x}\left(\overline{u}^n_{i,j}\overline{u}^{n+1}_{i,j}\right)\\
&= \frac{1}{h}\left(\overline{u}^n_{i+1,j}\overline{u}^{n+1}_{i+1,j} - \overline{u}^n_{i,j}\overline{u}^{n+1}_{i,j}\right)\\
&= \frac{1}{2h}\left[\overline{u}^n_{i+1,j}\left(u^{n+1}_{i+1,j} + u^{n+1}_{i,j}\right) - \overline{u}^n_{i,j}\left(u^{n+1}_{i,j} + u^{n+1}_{i-1,j}\right)\right]\,,\\
&= \frac{1}{2h}\left[\overline{u}^n_{i+1,j}u^{n+1}_{i+1,j} + \left(\overline{u}^n_{i+1,j} + \overline{u}^n_{i,j}\right)u^{n+1}_{i,j} - \overline{u}^n_{i,j}u^{n+1}_{i-1,j}\right]\,,
\end{aligned} \tag{3.40}$$

with similar expansions for the various other nonlinear terms in Eqs. (3.39a) and (3.39b). Also we note that time-level notation has been suppressed on velocity component entries in the continuity equation (3.39c) because this will depend on details of the implementation, as will be discussed below. Furthermore, it is of interest to observe that Eq. (3.39c) is of precisely the same form as the pressure update formula in SOLA, Eq. (3.24).

We are now prepared to consider some details of implementation of artificial compressibility. There are actually several ways by which this might be done; here we will provide a pseudo-language algorithm that is similar in structure to a MAC method. In particular, the first step at each new pseudo-time step will be computation of pressure at the new time level. Once this has been done the momentum equations will be solved for $\{u_{i,j}\}$ and $\{v_{i,j}\}$. Although these will not be mass-conserved, we expect that as $\tau \to \infty$ mass conservation will occur. The pseudo-language algorithm is the following.

Algorithm 3.3 *(Artificial Compressibility) Suppose n pseudo-time steps have been completed. The $n+1^{th}$ step is calculated as follows.*

1. *Update pressure to time level $n+1$ using Eq. (3.39c) with velocities from the n^{th} time level.*

2. *Solve the momentum equations* (3.39a) *and* (3.39b) *for velocity components u and v using advanced time level pressure computed in step 1., and employing a fully implicit backward-Euler time stepping procedure implemented with Douglas–Rachford time splitting.*

3. *Test convergence to steady state; if not converged, return to step 1.*

There are a number of observations to be made regarding this algorithm. Probably the most important is that even if $\partial p/\partial t \to 0$ occurs, there is nothing to guarantee, *a priori*, that the associated pressure is in any way related to the physical pressure. In particular, Eq. (3.39c) depends on a parameter δ_a (the artificial compressibility) that is only approximately related to true physical compressibility, and the continuity equation being employed is very different from the actual compressible continuity equation. On the other hand, the pressure field obtained via (3.39c) is such as to lead to satisfaction of the divergence-free constraint, but it does not satisfy any particular boundary conditions. Moreover, because the computed pressure is in no (theoretical) way directly related to the pressure gradient appearing in the momentum equations (in contrast to the case of pressures computed from a true pressure Poisson equation, as in the MAC method), it is difficult to justify its use in the momentum equations as done here. We again emphasize that it is not even a dimensionally-correct quantity. Finally, the pressure computed from Eq. (3.39c) is unique only up to an arbitrary function of (x, y). Thus, pressure gradients used in the momentum equations could be very inaccurately predicted. Nevertheless, the steady-state pressures produced by this algorithm generally appear to be at least qualitatively in accord with physical expectations although it is not clear that many detailed investigations of this have been conducted to ascertain the level of absolute accuracy.

We next observe that there are other alternatives to implementing artificial compressibility. In light of the close relationship to SOLA, an obvious one would be to compute a provisional velocity from the momentum equations at the beginning of each time step, as done in SOLA, and follow this with a pressure update obtained from Eq. (3.39c). This could then be iterated (within the momentum equation time step) with the SOLA velocity updates to produce a divergence-free velocity field at the end of each time step, thus permitting solution of time-dependent problems. There are also other alternatives, but it is not clear that any of these deserve much serious attention because the algorithms to be presented in the next two sections of these notes have proven to be so effective that it is unlikely that other approaches will be very competitive.

3.4 Projection Methods

Projection methods as a class of Navier–Stokes equation solution procedures can be viewed as arising in the work of Leray [2] and [3] dating back to the early 20^{th} Century, but they were introduced in the context of numerical simulations of the N.–S. equations by Chorin [64] and [69]. The basic ideas, however, were already known to Krzywicki and Ladyzhenskaya [70], although they were not implemented in computational procedures. Furthermore, Fujita and Kato [71] employed similar ideas for proofs of existence and uniqueness of N.–S. solutions.

One of the most attractive features of projection methods is that unlike all of the previously discussed techniques which have been constructed mainly on the basis of *ad hoc* (often physical) arguments, projection methods, at least in an abstract setting, (disregarding what actually is necessary to produce an implementation) have a quite sound mathematical underpinning.

There have been numerous versions of the projection method following its introduction by Chorin, but it is not the purpose of these notes to provide a detailed survey of these. We will briefly review Chorin's original approach because it contains elements that are important for a basic understanding of the method in general, and we will then proceed to a fairly detailed description of a modern approach that contains parts of the method presented by Kim and Moin [46] and also aspects studied by Gresho [72]. We will carry this out first in an analytical framework in which it is easy to understand the basic principles being used, and we then provide details needed for implementation of a computational algorithm. We note that much has been accomplished by various investigators, beyond what we present in these lectures, including early work of Fortin *et al.* [73] published soon after Chorin's method, a second-order method by Bell *et al.* [74] and fairly recent studies by Shen [75] and [76], to name only a few.

3.4.1 Outline of Chorin's projection method

The basic idea underlying construction of any projection method is the following theorem often referred to as the *Hodge decomposition theorem*, as mentioned in Chap. 1.

Theorem 3.1 *Every vector-valued function* $\boldsymbol{v} \in L^2(\Omega)$, $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, *can be decomposed as*

$$\boldsymbol{v} = \nabla q + \nabla\times\boldsymbol{\phi},$$

where q *and* $\boldsymbol{\phi}$ *are appropriately smooth in the sense of distributions.*

This theorem is stated more precisely, and proven (for $\Omega \subset \mathbb{R}^2$), in Girault and Raviart [77], and it is of course reminiscent of the Helmholtz–Leray projection discussed in Chap. 1, which should not be surprising. Clearly, since curl of a gradient is automatically zero, as is the divergence of a curl (demonstration of which we leave to the reader), it follows that in words we could state this theorem as "Any L^2 vector-valued function can be decomposed as the sum of curl-free and divergence-free parts."

Chorin makes direct use of this in the following way. Write the momentum equations as

$$\boldsymbol{u}_t + \nabla p = \boldsymbol{F(u)}\,, \tag{3.41}$$

where, as usual, $\boldsymbol{u} = (u, v)^T$, and where $\boldsymbol{F(u)}$ must contain the advective and diffusive terms of the N.–S. equations. Now if we suppose $\boldsymbol{F(u)} \in L^2$, (*i.e.*, $\boldsymbol{u} \in H^2$) the Hodge decomposition theorem implies that it can be expressed as a sum of curl-free and divergence-free parts. But it is clear that the second term on the left in (3.41) is curl-free, and moreover if we assume the velocity field is divergence-free, as it must be for an incompressible flow, then the first term on the left is divergence free.

We observe that specific details of application of these ideas are somewhat complicated, and the resulting algorithm is no longer widely used (but see various works by Bell and coworkers, *e.g.*, [74]); so we will not spend much time on this. But we do note that from the standpoint of implementation, the above equation was split in the sense of fractional-step operator splitting (as in Yanenko [78]), discretized on an unstaggered grid, and stepped forward in time using the Dufort–Frankel procedure (see, *e.g.*, [20]). The spatial discretizations were centered except near boundaries, where they were one-sided, and the algorithm employed an iteration of velocity and pressure similar to what we have already seen in the SOLA algorithm. It is of interest to note that Chorin's algorithm preceded SOLA by nearly 10 years, but the authors of SOLA were apparently not aware of its existence. Finally, we note that as in Rhie and Chow [38], the div-stability condition was circumvented via alterations to both advective term discretizations (only near boundaries) and use of specific averaging procedures in constructing pressure gradient terms. It does not appear, however, that these authors were aware of Chorin's work, which preceded theirs by nearly 15 years.

3.4.2 Analytical construction of a modern projection method

In this subsection we provide a detailed analysis of a more modern projection method, probably first used by Kim and Moin [46] in a form somewhat similar to what we present. Considerable analysis of many of the steps employed can be found in the paper by Gresho [72], where the method emphasized here is termed *projection-1.*

We begin with the incompressible Navier–Stokes equations expressed in conservation form as

$$\boldsymbol{u}_t + \nabla\cdot\left(\boldsymbol{u}^2\right) = -\nabla p + \nu\Delta\boldsymbol{u}\,, \tag{3.42a}$$

$$\nabla\cdot\boldsymbol{u} = 0\,. \tag{3.42b}$$

These equations can be viewed as being either two or three dimensional; our treatment would be the same in either case, so we will simply employ the vector notation given here until we begin construction of computational algorithms.

The first step in most basic projection methods is to solve the momentum equations without pressure gradient terms, and then "solve" a simple equation including the pressure gradient. Formally, this is a fractional-step procedure, as studied extensively by Yanenko [78], which we can express as

$$\widehat{\boldsymbol{u}}_t + \nabla\cdot\left(\widehat{\boldsymbol{u}}^2\right) = \nu\Delta\widehat{\boldsymbol{u}}\,, \tag{3.43a}$$

$$\boldsymbol{u}_t = -\nabla p\,. \tag{3.43b}$$

The first of these is a vector system of Burgers' equations to be solved for $\widehat{\boldsymbol{u}}$, an "auxiliary velocity" that is <u>not</u> divergence free, which is relatively easy, assuming that such difficulties as the cell-*Re* problem and aliasing can be handled. The second equation is converted to a pressure Poisson equation whose solution leads to construction of the Leray projector needed to obtain the required divergence-free solution.

Taking the divergence of Eq. (3.43b) yields

$$\nabla\cdot\boldsymbol{u}_t = (\nabla\cdot\boldsymbol{u})_t = -\nabla\cdot\nabla p = -\Delta p\,.$$

At this point we recognize that this is not the pressure Poisson equation for the true pressure (recall Eq. (2.22)), and we introduce the notation $\phi \sim p$ for the "pseudo pressure." Thus, we write the above as

$$-\Delta\phi = (\nabla\cdot\boldsymbol{u})_t\ , \tag{3.44}$$

and we approximate the right-hand side as

$$(\nabla\cdot\boldsymbol{u})_t \simeq \frac{(\nabla\cdot\boldsymbol{u})^{n+1} - (\nabla\cdot\widehat{\boldsymbol{u}})}{k}\,. \tag{3.45}$$

Here, as usual, k denotes a time step size (the <u>same</u> one used to numerically solve Eq. (3.43a)).

Now we require $(\nabla\cdot\boldsymbol{u})^{n+1} = 0$ as was done in the MAC method, and this allows us to write (3.44) as

$$\Delta\phi = \frac{\nabla\cdot\widehat{\boldsymbol{u}}}{k}\,. \tag{3.46}$$

It is clear that for any $\widehat{\boldsymbol{u}}$, no matter how obtained, if ϕ satisfies (3.46), then constructing $\boldsymbol{u}^{n+1}$ as

$$\boldsymbol{u}^{n+1} = \widehat{\boldsymbol{u}} - k\nabla\phi \tag{3.47}$$

implies $(\nabla\cdot\boldsymbol{u})^{n+1} = 0$, as required. The reader will recognize Eq. (3.47) as the Leray projector of Chap. 1, except for the factor k. The need for this factor will become clear as we proceed.

A few observations regarding this procedure are in order. First, it is equivalent to Gresho's *projection-1* [72] method provided we set

$$p^{n+1} = \phi\,,$$

with p being the physical pressure. Considerable analysis of this is provided in [72], but the most notable result is one given by Kim and Moin [46]; namely

$$p = \phi + \mathcal{O}(k/Re)\,. \tag{3.48}$$

We remark that a proof of this is alluded to in [46], but the approach suggested is spurious except possibly in the 2-D case where it can be guaranteed that N.–S. solutions are unique. Even then, however, the same uniqueness would need to be proven for the discretized equations in order for the proposed "proof" to be valid.

Second, we should observe that because of the fractional-step construction of this projection method, independent of the integration procedure used to advance the momentum equations in time, the final result for $\boldsymbol{u}$ can have order of accuracy no better than $\mathcal{O}(k)$—and it could be worse. We can, in fact, obtain a rough estimate in the following way. For simplicity consider a forward-Euler time integration of the momentum equations, so we can express the semi-discrete form of Eq. (3.43a) as

$$\widehat{\boldsymbol{u}} = \boldsymbol{u}^n + k\left[\nu\Delta\boldsymbol{u}^n\nabla\cdot\boldsymbol{u}^{n\,2}\right] .$$

Then from (3.47) it follows that

$$\widehat{\boldsymbol{u}} = \boldsymbol{u}^{n+1} + k\nabla\phi ,$$

and substitution of this into the above yields, after rearrangement,

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + k\left[\nu\Delta\boldsymbol{u}^n - \nabla\cdot\boldsymbol{u}^{n\,2} - \nabla\phi\right] . \tag{3.49}$$

We first notice that this would be exactly the forward-Euler integration of the momentum equations (and, thus, would be first-order accurate in time) if ϕ were replaced with the physical pressure p. But from Eq. (3.48) we see that ϕ and p differ by $\mathcal{O}(k)$, so this substitution will result in a local $\mathcal{O}(k^2)$ error, and thus a global error that is $\mathcal{O}(k)$. We emphasize that this is separate from the integration error. It comes from the fractional-steps operator splitting, and it cannot be removed by simply increasing the order of accuracy of the integration scheme. Nevertheless, the overall projection scheme constructed in this way is first order in time and is thus an acceptable practical approach. Moreover, the same result can be obtained for the implicit backward-Euler time integration method. We leave demonstration of this as an exercise for the reader.

Finally, we note that for simulations of fluid flow in which high-order accuracy is needed for the pressure (*e.g.*, flows with free surfaces in which pressure is involved in a force balance needed to set the location of the free surface), it may be necessary to solve the actual Poisson equation for pressure. This can now be done without any but the usual numerical approximations because the velocity field is divergence free, and Eq. (2.22) can be used directly with appropriate boundary conditions given in Eqs. (2.35) and (2.36), in contrast to what was necessary, for example, in the MAC method. The disadvantage of this is, of course, that an additional PPE must be solved, and this can be quite expensive. In cases where the physical pressure is not needed in the solution process, but might be desired at particular times during the simulation, it is possible to perform this second PPE solve only at those desired times because the exact pressure is not needed in advancing the projection method forward in time.

3.4.3 Numerical implementation of projection methods

As we have already noted, there have been numerous versions of the basic idea first implemented by Chorin [64], and what is to be presented here is a fairly modern approach, due in part to Kim and Moin [46] and to Gresho [72]. We begin by providing an outline of the steps needed to implement this procedure, and we follow this with subsections detailing each of these steps. Finally, we will present a fairly detailed, but still coarse-grained, pseudo-language algorithm casting the original outline in a more computational form.

Outline of Basic Steps

The basic steps needed to construct a projection method of the form we will consider herein are the following.

1. Solve the "Burgers' equation" form of the momentum equations via the following steps/techniques:
    (a) generalized trapezoidal integration in time;
    (b) δ-form quasilinearization (or Newton's method);
    (c) spatial discretization of right-hand sides of equations;

   (d) spatial discretization of left-hand sides of equations;
   (e) Douglas & Gunn time splitting;
   (f) boundary condition treatment during Newton iterations.

2. Filter solution $\widehat{\boldsymbol{u}}$ obtained from Burgers' equation, if necessary.

3. Perform Leray projection to obtain divergence-free advanced time level solution, $\boldsymbol{u}^{n+1}$:

   (a) construct divergence of $\widehat{\boldsymbol{u}}$;
   (b) solve PPE for pseudo pressure ϕ;
   (c) use ϕ to construct Leray projector, and apply to $\widehat{\boldsymbol{u}}$.

4. Solve true PPE for physical pressure, if needed.

5. Prepare results for output to post processing software.

We observe that Chorin's original projection method was used by him only for steady-state problems. Indeed, it does not seem possible to rigorously prove that its temporal order of accuracy is any better than $\mathcal{O}(k^{1/4})$. However, this low accuracy appears to be caused by near-boundary discretizations employed to circumvent the div-stability condition on the (natural) unstaggered grid employed by Chorin [64]. On staggered grids as we are using herein, the preceding algorithm is temporally $\mathcal{O}(k)$, and we will later see that it is relatively easy to upgrade it to $\mathcal{O}(k^2)$ accuracy, at least for the velocity components. Thus, it is a viable procedure for time-dependent problems and, in fact, probably the best that is available.

The fact that this approach is useful for such problems leads us to summarize it via a formal discrete solution operator expressed as

$$\boldsymbol{u}^{n+1} = P_L F(\delta_h) S_h(k) \boldsymbol{u}^n \,, \tag{3.50}$$

where S_h is any chosen basic discrete solution operator constructed on a grid of spacing h and evolving the (discrete) solution for a time step k; $F(\delta_h)$ is a discrete mollifier with kernel δ_h such as the Shuman filter [25] described in Chap. 2, and P_L is the Leray projector of Chap. 1. We will consider each of these in more detail in what follows, but we emphasize here that (3.50) shows that filtering $\underline{\text{must}}$ be done $\underline{\text{before}}$ projection to guarantee the required divergence-free velocity field.

Solution of Burgers' equation form of momentum equations

As the preceding outline shows, there are numerous tasks associated with this first step corresponding to construction of the discrete solution operator S_h, and from the standpoint of programming a computer code this is the most difficult part. We will be working with the staggered-grid formulation of the governing equations, but most of what we do here could also be done on cell-centered, co-located grids, as presented in Zang *et al.* [41]. We begin by presenting the staggered-grid finite-volume equations in a formal analytical form that follows directly from Eqs. (3.3)—by working in reverse, from discrete to continuous—to provide a reminder of the various averages that must be employed in this formulation:

$$u_t + \left(\overline{u}^2\right)_x + \left(\widetilde{u}\,\overline{v}\right)_y = -p_x + \nu\Delta u \,, \tag{3.51a}$$

$$v_t + \left(\widetilde{u}\,\overline{v}\right)_x + \left(\widetilde{v}^2\right)_y = -p_y + \nu\Delta v \,, \tag{3.51b}$$

where " $^-$ " and " $^\sim$ " denote horizontal and vertical averages, repectively, as given in Eqs. (3.8). To obtain the corresponding Burgers' equations we simply delete the pressure-gradient terms and introduce the " $^\frown$ " notation to indicate *auxiliary velocity* components.

Generalized Trapezoidal Integration. We can now immediately apply generalized trapezoidal integration to these equations to obtain, for example, for the x-momentum equation,

$$u^{n+1} = u^n + \theta k \Big[\nu\Delta u - \left(\overline{u}^2\right)_x - \left(\widetilde{u}\,\overline{v}\right)_y\Big]^{n+1} + (1-\theta)k\Big[\nu\Delta u - \left(\overline{u}^2\right)_x - \left(\widetilde{u}\,\overline{v}\right)_y\Big]^n \,. \tag{3.52}$$

Here, θ is a user-provided parameter that when set equal to 1.0 yields a backward-Euler integration scheme, and when set to 0.5 produces trapezoidal integration. As is also clear, $\theta = 0$ in (3.52) yields the explicit forward-Euler method, which is rather trivial and will not be specifically treated herein. (The " $\widehat{\ }$ " notation has been temporarily suppressed for simplicity, but it is understood that the solution to this equation is $\widehat{u}$ in the absence of a pressure-gradient term.) We remark that the backward-Euler scheme can be valuable when only a steady-state solution is sought because this form is usually somewhat more stable for nonlinear problems than is the trapezoidal method, and temporal accuracy is of no importance for a steady-state solution. Thus, with $\theta = 1$ it is possible to employ larger time step sizes k, and reach a steady state somewhat sooner—in principle. But the reader should be cautioned that very large time steps are not guaranteed to be optimal because the essential aspect of any such calculation (equivalent to a fixed-point iteration for the steady-state solution) is the spectral radius of the corresponding iteration matrix (see [44]), and this is not necessarily a monotone function of the time step size k.

Quasilinearization. Quasilinearization (the Newton–Kantorovich procedure—see, *e.g.*, Ames [59]) is, in one form or another, the most widely-used linearization of the N.–S. equations. In many problems it is completely equivalent to ordinary Newton iteration, but in the present case this is not true due to averaging of the velocity field needed to construct the control-volume implementation on a staggered grid. In this section we will see that it is possibly better to completely discretize the governing equations, and then apply the usual Newton method approach rather than linearizing first and then discretizing the linear PDEs, as done in quasilinearization. Nevertheless, we provide a complete treatment of quasilinearization, if for no other reason than to contrast the two approaches in the case of a staggered-grid implementation.

We will demonstrate the difficulty that arises by first considering the x-direction advection term from the x-momentum equation in detail: that is, $\left(\overline{u}^2\right)_x$. We initially perform quasilinearization in the usual way by defining

$$F(u) \equiv u^2 \,, \tag{3.53}$$

and writing

$$F(u) \simeq u^{(m)\,2} + \left(\frac{\partial F}{\partial u}\right)^{(m)} \delta u + \cdots \,,$$

where, as usual, the superscript (m) denotes an iteration counter, and

$$\delta u \equiv u - u^{(m)} \,. \tag{3.54}$$

We now observe that because some terms in Eq. (3.52) involve averaged quantities, and others do not, it is somewhat difficult to directly apply the above definitions on a discrete, pointwise basis. We must ultimately solve a system for δu consisting of the Jacobian matrix (constructed from the $(\partial F/\partial u)^{(m)}$) and a right-hand side function, but the right-hand side will be expressed in terms of both averaged and unaveraged quantities on a staggered grid, ultimately introducing discrete values of δu other than simply $\delta u_{i,j}$ at each grid point (*i.e.*, in each grid cell). This then complicates construction of the left-hand side (Jacobian) matrix; but if this is carried out properly, the usual band structure is not altered.

To see this, replace u with $\overline{u}$ in Eq. (3.53), as required by (3.52):

$$F(\overline{u}) = \overline{u}^2 \,.$$

Now substitute the definition of $\overline{u}$, Eq. (3.8a), on the right-hand side, and suppress the constant j indexing for simplicity, to obtain

$$\begin{aligned} F_i(\overline{u}_i) &= \left[\frac{1}{2}\left(u_i + u_{i-1}\right)\right]^2 \\ &= \frac{1}{4}\left(u_{i-1}^2 + 2u_iu_{i-1} + u_i^2\right) \,. \end{aligned}$$

This shows that, discretely, $F_i(\overline{u}_i)$ depends on u at two different grid points, implying there must be two different contributions to δu, *viz.*, δu_i and δu_{i-1}.

Thus, in this case we see that

$$\begin{aligned} F_i(\overline{u}_i) &= F_i(\overline{u}_i^{(m)}) + \left(\frac{\partial F}{\partial u_{i-1}}\right)_i^{(m)} \delta u_{i-1} + \left(\frac{\partial F}{\partial u_i}\right)_i^{(m)} \delta u_i + \cdots \\ &= \overline{u}_i^{(m)\,2} + \overline{u}_i^{(m)} \delta u_{i-1} + \overline{u}_i^{(m)} \delta u_i + \cdots . \end{aligned} \tag{3.55}$$

It then follows, using the D_+ difference operator applied with cell indexing, that

$$\begin{aligned} (F_i(\overline{u}_i))_x &= \left(\overline{u}_i^{(m)\,2}\right)_x + \left(\overline{u}_i^{(m)} \delta u_{i-1}\right)_x + \left(\overline{u}_i^{(m)} \delta u_i\right)_x + \cdots \\ &\simeq \frac{1}{h}\left(\overline{u}_{i+1}^{(m)\,2} - \overline{u}_i^{(m)\,2}\right) + \frac{1}{h}\left(\overline{u}_{i+1}^{(m)} \delta u_i - \overline{u}_i^{(m)} \delta u_{i-1}\right) + \frac{1}{h}\left(\overline{u}_{i+1}^{(m)} \delta u_{i+1} - \overline{u}_i^{(m)} \delta u_i\right) , \end{aligned} \tag{3.56}$$

rather than simply

$$(F_i(\overline{u}_i))_x = \frac{1}{h}\left(\overline{u}_{i+1}^{(m)\,2} - \overline{u}_i^{(m)\,2}\right) + \frac{1}{h}\left(\overline{u}_{i+1}^{(m)} \delta u_{i+1} - \overline{u}_i^{(m)} \delta u_i\right) ,$$

as would be obtained without taking into account the dependence of F_i on both u_i and u_{i-1}.

If, on the other hand, we quasilinearize with respect to $\overline{u}$ instead of with respect to u, despite the fact that it is δu (and not $\delta\overline{u}$) that is needed, we obtain

$$F(\overline{u}) \equiv \overline{u}^{(m)\,2} + \left(\frac{\partial F}{\partial \overline{u}}\right)^{(m)} \delta\overline{u} + \cdots , \tag{3.57}$$

and in this case

$$\left(\frac{\partial F}{\partial \overline{u}}\right)^{(m)} = 2\overline{u}^{(m)} .$$

Thus, we have

$$F(\overline{u}) = \overline{u}^{(m)\,2} + 2\overline{u}^{(m)} \delta\overline{u} + \cdots ,$$

and

$$\begin{aligned} \left(\overline{u}^{(m)\,2}\right)_x\Big|_i &\simeq \frac{1}{h}\left(\overline{u}_{i+1}^{(m)\,2} - \overline{u}_i^{(m)\,2}\right) + \frac{2}{h}\left(\overline{u}_{i+1}^{(m)} \delta\overline{u}_{i+1} - \overline{u}_i^{(m)} \delta\overline{u}_i\right) \\ &= \frac{1}{h}\left(\overline{u}_{i+1}^{(m)\,2} - \overline{u}_i^{(m)\,2}\right) + \frac{1}{h}\left(\overline{u}_{i+1}^{(m)} (\delta u_{i+1} + \delta u_i) - \overline{u}_i^{(m)} (\delta u_i + \delta u_{i-1})\right) \\ &= \frac{1}{h}\left(\overline{u}_{i+1}^{(m)\,2} - \overline{u}_i^{(m)\,2}\right) + \frac{1}{h}\left(\overline{u}_{i+1}^{(m)} \delta u_i - \overline{u}_i^{(m)} \delta u_{i-1}\right) + \frac{1}{h}\left(\overline{u}_{i+1}^{(m)} \delta u_{i+1} - \overline{u}_i^{(m)} \delta u_i\right) , \end{aligned} \tag{3.58}$$

which now is the same as (3.56). We again remind the reader that indexing is that of cells, and not of grid points (see Fig. 2.10), and as a consequence what appear as one-sided first-order difference approximations are actually centered with respect to an appropriate cell (and are thus, formally, second-order accurate).

We see that although Eqs. (3.56) and (3.58) are identical, the analysis leading to the latter is somewhat nonintuitive. As a consequence, it is possibly better in this situation to directly apply Newton's method to the fully-discrete equations rather than attempting to use quasilinearization in this somewhat ambiguous situation; we will do this in the next section. In any case, it is clear from these results that the Jacobian matrix needed on the left-hand side of the equation is somewhat different than would be the case in the absence of averaging.

Since we see that the correct result can be recovered by a slight modification of the usual approach to quasilinearization, we will carry this out here for the second advective term of the x-momentum equation, and check this later via Newton's method. Thus, we consider

$$G(\widetilde{u}) \equiv \widetilde{u}\,\overline{v}\,, \tag{3.59}$$

Then formal quasilinearization with respect to $\widetilde{u}$ results in

$$G(\widetilde{u}) = (\widetilde{u}\,\overline{v})^{(m)} + \left(\frac{\partial G}{\partial \widetilde{u}}\right)^{(m)} \delta\widetilde{u} + \cdots\,,$$

where we have not linearized with respect to $\overline{v}$ because we intend to ultimately employ a sequential solution process for the momentum equations, and in this case nothing is gained by introducing terms associated with $\delta\overline{v}$ in the x-momentum equation. But we note that if a fully-coupled solution algorithm were employed, it would be necessary to also include terms associated with $\delta\overline{v}$.

From Eq. (3.59) we see that

$$\frac{\partial G}{\partial \widetilde{u}} = \overline{v}\,,$$

so we have

$$G(\widetilde{u}) = (\widetilde{u}\,\overline{v})^{(m)} + \overline{v}^{(m)}\delta\widetilde{u} + \cdots\,. \tag{3.60}$$

Then differentiation with respect to y, as required in (3.52), yields

$$(G(\widetilde{u}))_y = (\widetilde{u}\,\overline{v})_y^{(m)} + \left(\overline{v}^{(m)}\delta\widetilde{u}\right)_y + \cdots\,,$$

and evaluation at a vertical grid location j (x-direction indices are now constant, and suppressed) results in

$$\begin{aligned}
(\widetilde{u}\,\overline{v})_y\Big|_j &= \frac{1}{h}\left(\widetilde{u}_j\,\overline{v}_j - \widetilde{u}_{j-1}\,\overline{v}_{j-1}\right)^{(m)} + \frac{1}{h}\left(\overline{v}_j^{(m)}\delta\widetilde{u}_j - \overline{v}_{j-1}^{(m)}\delta\widetilde{u}_{j-1}\right)\\
&= \frac{1}{h}\left(\widetilde{u}_j\,\overline{v}_j - \widetilde{u}_{j-1}\,\overline{v}_{j-1}\right)^{(m)} + \frac{1}{2h}\left(\overline{v}_j^{(m)}(\delta u_j + \delta u_{j+1}) - \overline{v}_{j-1}^{(m)}(\delta u_{j-1} + \delta u_j)\right)\\
&= \frac{1}{h}\left(\widetilde{u}_j\,\overline{v}_j - \widetilde{u}_{j-1}\,\overline{v}_{j-1}\right)^{(m)} + \frac{1}{2h}\left(\overline{v}_j^{(m)}\delta u_j - \overline{v}_{j-1}^{(m)}\delta u_{j-1}\right) + \frac{1}{2h}\left(\overline{v}_j^{(m)}\delta u_{j+1} - \overline{v}_{j-1}^{(m)}\delta u_j\right).
\end{aligned} \tag{3.61}$$

This completes the analyses needed to construct the Jacobian matrix corresponding to quasilinearization of the x-momentum equation. In particular, at each grid point the elements of this matrix are just the coefficients of $\delta u_{i-1,j}$, $\delta u_{i,j}$ and $\delta u_{i+1,j}$, all of which can be read from (3.58) and (3.61).

For the sake of completeness we present analogous results for the advective terms of the y-momentum equation, but leave details of their derivations to the interested reader. We have

$$(\widetilde{u}\,\overline{v})_x\Big|_i = \frac{1}{h}\left(\widetilde{u}_i\,\overline{v}_i - \widetilde{u}_{i-1}\,\overline{v}_{i-1}\right)^{(m)} + \frac{1}{2h}\left(\widetilde{u}_i^{(m)}\delta v_i - \widetilde{u}_{i-1}^{(m)}\delta v_{i-1}\right) + \frac{1}{2h}\left(\widetilde{u}_i^{(m)}\delta v_{i+1} - \widetilde{u}_{i-1}^{(m)}\delta v_i\right), \tag{3.62}$$

and

$$\left(\widetilde{v}^{(m)\,2}\right)_y\Big|_j = \frac{1}{h}\left(\widetilde{v}_{j+1}^{(m)\,2} - \widetilde{v}_j^{(m)\,2}\right) + \frac{1}{h}\left(\widetilde{v}_{j+1}^{(m)}\delta\widetilde{v}_{j+1} - \widetilde{v}_j^{(m)}\delta\widetilde{v}_j\right) + \frac{1}{h}\left(\widetilde{v}_{j+1}^{(m)}\delta\widetilde{v}_j - \widetilde{v}_j^{(m)}\delta\widetilde{v}_{j-1}\right)\,, \tag{3.63}$$

with

$$\delta v \equiv v - v^{(m)}\,, \tag{3.64}$$

analogous to Eq. (3.54).

At this point only treatment of linear terms remains, but of course there is no actual linearization to be done—linearization of a linear operator reproduces the linear operator. One simply needs to substitute the definitions of δu and δv, Eqs. (3.54) and (3.64), respectively, into the corresponding time level $n+1$ entries of the Burgers' equation forms of the x- and y-momentum equations. Thus, for example, from Eq. (3.52) we consider

$$\begin{aligned}\Delta u &= \Delta\left(u^{(m)} + \delta u\right)\\ &\simeq \left(D^2_{0,x} + D^2_{0,y}\right)\left(u^{(m)}_{i,j} + \delta u_{i,j}\right)\,,\end{aligned}\tag{3.65}$$

and we observe that these contributions consist only of grid-cell values, and not their averages; hence, they are simpler to treat.

We will close this section on quasilinearization by presenting the complete δ-form of the momentum equations and noting some of the features of this representation. The preceding analyses can be combined to produce the following semi-discrete (discrete in time only) result:

$$\begin{aligned}\left\{I-\theta k\left[\nu\Delta(\,\cdot\,)-\left(\overline{u}^{(m)}\,\cdot\right)_x-\left(\overline{v}^{(m)}\,\cdot\right)_y\right]\right\}\delta u &= u^n-u^{(m)}+\theta k\left[\nu\Delta u^{(m)}-\left(\overline{u}^{(m)\,2}\right)_x-\left(\widetilde{u}^{(m)}\overline{v}^{(m)}\right)_y\right]\\ &\quad+(1-\theta)k\left[\nu\Delta u^n-\left(\overline{u}^{n\,2}\right)_x-\left(\widetilde{u}^n\overline{v}^n\right)_y\right]\,,\end{aligned}\tag{3.66a}$$

and

$$\begin{aligned}\left\{I-\theta k\left[\nu\Delta(\,\cdot\,)-\left(\widetilde{u}^{(m)}\,\cdot\right)_x-\left(\widetilde{v}^{(m)}\,\cdot\right)_y\right]\right\}\delta v &= v^n-v^{(m)}+\theta k\left[\nu\Delta v^{(m)}-\left(\widetilde{u}^{(m)}\overline{v}^{(m)}\right)_x-\left(\widetilde{v}^{(m)\,2}\right)_y\right]\\ &\quad+(1-\theta)k\left[\nu\Delta v^n-\left(\widetilde{u}^n\overline{v}^n\right)_x-\left(\widetilde{v}^{n\,2}\right)_y\right]\,.\end{aligned}\tag{3.66b}$$

In the above representations notation such as, *e.g.*, $\left(\overline{u}^{(m)}\,\overline{\cdot}\right)_x$ implies that an appropriately-averaged value of δu is to be inserted into the slot indicated by the " $\cdot$ " and then discretized by a chosen method consistent with what was done earlier in arriving at results given in Eqs. (3.58) and (3.61).

It should be observed that the right-hand sides of these equations are, in fact, the complete original equations in the limit $m \to \infty$, and they can be directly (explicitly) evaluated for any m; the left-hand sides are equivalent to the Jacobian matrices of a Newton's method. In this regard we note that the former should be discretized to gain as much accuracy as is appropriate, and the left-hand side should be approximated so as to obtain maximum numerical stability. In particular, it is not necessary that the left-hand sides of (3.66) be the exact Jacobian matrix corresponding to the right-hand sides, although if the differences are too great the favorable convergence properties of Newton's method may be lost. We will make use of these ideas in what follows.

Spatial Discretization of Right-Hand Side. Spatial discretization of the right-hand sides of Eqs. (3.66) is straightforward, at least if relatively low-order approximations are employed as we will do here. We will carry out details for the x-momentum equation, and simply state the corresponding result for y momentum.

We begin by introducing formal difference operators into the right-hand side of Eq. (3.66a) in place of the differential operators: thus, we obtain

$$\begin{aligned}u^n_{i,j}-u^{(m)}_{i,j} &+ \theta k\left[\nu\left(D^2_{0,x}+D^2_{0,y}\right)u^{(m)}_{i,j}-D_{+,x}\left(\overline{u}^{(m)\,2}_{i,j}\right)-D_{-,y}\left(\widetilde{u}^{(m)}_{i,j}\,\overline{v}^{(m)}_{i,j}\right)\right]\\ &+ (1-\theta)k\left[\nu\left(D^2_{0,x}+D^2_{0,y}\right)u^n_{i,j}-D_{+,x}\left(\overline{u}^{n\,2}_{i,j}\right)-D_{-,y}\left(\widetilde{u}^n_{i,j}\,\overline{v}^n_{i,j}\right)\right]\,.\end{aligned}$$

The next step is to expand the difference operators. We do this only for the first line of terms involving the m^{th} quasilinearization iterate because this is all that will be needed to construct the Newton method

Jacobian matrix, as we will do momentarily, and the second line of terms will take an identical form with respect to discretization but is not needed in the Jacobian matrix. We thus obtain

$$\begin{aligned} u^n_{i,j} - u^{(m)}_{i,j} \;+\; \theta k\Bigg[\nu\left(\frac{1}{h_x^2}(u_{i-1,j} - 2u_{i,j} + u_{i+1,j})^{(m)} + \frac{1}{h_y^2}(u_{i,j-1} - 2u_{i,j} + u_{i,j+1})\right)^{(m)} \\ - \frac{1}{h_x}\left(\overline{u}^2_{i+1,j} - \overline{u}^2_{i,j}\right)^{(m)} - \frac{1}{h_y}\left(\widetilde{u}_{i,j}\,\overline{v}_{i,j} - \widetilde{u}_{i,j-1}\,\overline{v}_{i,j-1}\right)^{(m)}\Bigg]. \end{aligned} \tag{3.67}$$

The analogous result for the y-momentum equation is

$$\begin{aligned} v^n_{i,j} - v^{(m)}_{i,j} \;+\; \theta k\Bigg[\nu\left(\frac{1}{h_x^2}(v_{i-1,j} - 2v_{i,j} + v_{i+1,j})^{(m)} + \frac{1}{h_y^2}(v_{i,j-1} - 2v_{i,j} + v_{i,j+1})\right)^{(m)} \\ - \frac{1}{h_x}\left(\widetilde{u}_{i,j}\,\overline{v}_{i,j} - \widetilde{u}_{i-1,j}\,\overline{v}_{i-1,j}\right)^{(m)} - \frac{1}{h_y}\left(\overline{v}^2_{i,j+1} - \overline{v}^2_{i,j}\right)^{(m)}\Bigg]. \end{aligned} \tag{3.68}$$

The next step obviously must be introduction of appropriate definitions of the averaged quantities so that it will be possible to apply Newton's method only to unaveraged ones. (Note that this was necessary for quasilinearization as well—but was less obvious.) Recall that we are solving the momentum equations sequentially, so the x-momentum equation will be linearized only with respect to the discrete entries of u (and not with respect to those of v)—and, conversely for the y-momentum equation. This corresponds to what is sometimes termed a "diagonal" Newton method. Thus, in Eq. (3.67) we will need to insert the definitions of $\overline{u}$ and $\widetilde{u}$ (but not that of $\overline{v}$ from Eqs. (3.8a) and (3.8c), respectively, into the advective terms. For the first of these we obtain

$$\begin{aligned} \overline{u}^2_{i+1,j} - \overline{u}^2_{i,j} &= \frac{1}{4}\left[(u_{i+1,j} + u_{i,j})^2 - (u_{i,j} + u_{i-1,j})^2\right] \\ &= \frac{1}{4}\left[u^2_{i+1,j} + 2\,(u_{i+1,j} - u_{i-1,j})\,u_{i,j} - u^2_{i-1,j}\right], \end{aligned} \tag{3.69}$$

where we have suppressed the iteration index since it is understood that this will occur on all terms. Similarly, for the second advective term we have

$$\widetilde{u}_{i,j}\,\overline{v}_{i,j} - \widetilde{u}_{i,j-1}\,\overline{v}_{i,j-1} = \frac{1}{2}\left[(u_{i,j} + u_{i,j+1})\overline{v}_{i,j} - (u_{i,j-1} + u_{i,j})\overline{v}_{i,j-1}\right]. \tag{3.70}$$

The fully-discrete time-level $n+1$ part of the right-hand side of the x-momentum equation can now be expressed at iteration m as

$$\begin{aligned} \mathcal{F}^{n+1}_x = -\,u^{(m)}_{i,j} \;+\; \theta k\Bigg\{\nu\left[\frac{1}{h_x^2}(u_{i-1,j} - 2u_{i,j} + u_{i+1,j})^{(m)} + \frac{1}{h_y^2}(u_{i,j-1} - 2u_{i,j} + u_{i,j+1})^{(m)}\right] \\ - \frac{1}{4h_x}\left[u^2_{i+1,j} + 2\,(u_{i+1,j} - u_{i-1,j})\,u_{i,j} - u^2_{i-1,j}\right]^{(m)} \\ - \frac{1}{2h_y}\left[(u_{i,j} + u_{i,j+1})\overline{v}_{i,j} - (u_{i,j-1} + u_{i,j})\overline{v}_{i,j-1}\right]^{(m)}\Bigg\}. \end{aligned} \tag{3.71}$$

An analogous expression holds for the y-momentum equation:

$$\begin{aligned} \mathcal{F}^{n+1}_y = -\,v^{(m)}_{i,j} \;+\; \theta k\Bigg\{\nu\left[\frac{1}{h_x^2}(v_{i-1,j} - 2v_{i,j} + v_{i+1,j})^{(m)} + \frac{1}{h_y^2}(v_{i,j-1} - 2v_{i,j} + v_{i,j+1})^{(m)}\right] \\ - \frac{1}{2h_x}\left[(v_{i,j} + v_{i,j+1})\widetilde{u}_{i,j} - (v_{i,j-1} + v_{i,j})\widetilde{u}_{i-1,j}\right]^{(m)} \\ - \frac{1}{4h_y}\left[v^2_{i,j+1} + 2\,(v_{i,j+1} - v_{i,j-1})\,v_{i,j} - v^2_{i,j-1}\right]^{(m)}\Bigg\}. \end{aligned} \tag{3.72}$$

At this point we recognize that there is a completely analogous term $\mathcal{F}_x^n$ corresponding to known time level n information, and that at each time step we require iteration until

$$\mathcal{F}_x(u_{i-1,j}, u_{i,j-1}, u_{i,j}, u_{i,j+1}, u_{i+1,j}) = 0\,,$$

to within a specified tolerance, with $\mathcal{F}_x \equiv \mathcal{F}_x^{n+1} + \mathcal{F}_x^n$, and similarly for $\mathcal{F}_y$.

The Jacobian matrix required for implementing a Newton iteration procedure is obtained by differentiating the right-hand sides of (3.71) and (3.72)—the $n+1$ time level part of $\mathcal{F}_x$ and $\mathcal{F}_y$—with respect to each of the five entries of u (or v) for every grid point. The corresponding matrices, $J(\mathcal{F}_x)$ and $J(\mathcal{F}_y)$, will each have the structure of a discrete Laplacian, and the system to be solved at each Newton iteration within a time step is of the form, *e.g.*,

$$J(\mathcal{F}_x)\delta u = -\mathcal{F}_x\,,$$

with δu as defined earlier in Eq. (3.54) in the context of quasilinearization. An analogous equation holds for δv in terms of functions $\mathcal{F}_y$ for solving the y-momentum equation. We remark that one iteration will usually involve computing a single solution of each of the x- and y-momentum equations rather than separate converged iterations of each in succession.

Spatial Discretization of Left-Hand Side. What now remains is to construct the elements of $J(\mathcal{F}_x)$ and $J(\mathcal{F}_y)$ to obtain the left-hand sides of Eqs. (3.66). We again emphasize that construction of the Jacobian matrices requires derivatives with respect only to variables at time level $n+1$ and that these components must be associated with the iteration counter m which we have suppressed in Eqs. (3.71) and (3.72). We also note that parts of $J(\mathcal{F}_x)$ and $J(\mathcal{F}_y)$ associated with the discrete Laplacian already appear explicitly in Eqs. (3.66), as do the factors θk. Nevertheless, we will include these in the expressions that follow so that for each $\delta u_{i,j}$ and $\delta v_{i,j}$ the elements of the corresponding row of the Jacobian matrix are completely known. Thus, for $J(\mathcal{F}_x)$ we have

$$\frac{\partial \mathcal{F}_x^{n+1}}{\partial u_{i-1,j}^{(m)}} = -\frac{\theta k \nu}{h_x^2} - \frac{1}{2h_x}\left(u_{i,j} + u_{i-1,j}\right)\,, \tag{3.73a}$$

$$\frac{\partial \mathcal{F}_x^{n+1}}{\partial u_{i,j-1}^{(m)}} = -\frac{\theta k \nu}{h_y^2}\,, \tag{3.73b}$$

$$\frac{\partial \mathcal{F}_x^{n+1}}{\partial u_{i,j}^{(m)}} = 1 - 2\theta k \nu\left(\frac{1}{h_x^2} + \frac{1}{h_y^2}\right) + \frac{1}{2h_x}\left(u_{i+1,j} - u_{i-1,j}\right) + \frac{1}{2h_y}\left(\overline{v}_{i,j} - \overline{v}_{i,j-1}\right)\,, \tag{3.73c}$$

$$\frac{\partial \mathcal{F}_x^{n+1}}{\partial u_{i,j+1}^{(m)}} = -\frac{\theta k \nu}{h_y^2}\,, \tag{3.73d}$$

$$\frac{\partial \mathcal{F}_x^{n+1}}{\partial u_{i+1,j}^{(m)}} = -\frac{\theta k \nu}{h_x^2} - \frac{1}{2h_x}\left(u_{i+1,j} + u_{i,j}\right)\,. \tag{3.73e}$$

Similarly, for the components of $J(\mathcal{F}_y)$ we have

$$\frac{\partial \mathcal{F}_y^{n+1}}{\partial v_{i-1,j}^{(m)}} = -\frac{\theta k\nu}{h_x^2}\,, \tag{3.74a}$$

$$\frac{\partial \mathcal{F}_y^{n+1}}{\partial v_{i,j-1}^{(m)}} = -\frac{\theta k\nu}{h_y^2} - \frac{1}{2h_y}\left(v_{i,j} + v_{i,j-1}\right)\,, \tag{3.74b}$$

$$\frac{\partial \mathcal{F}_y^{n+1}}{\partial v_{i,j}^{(m)}} = 1 - 2\theta k\nu\left(\frac{1}{h_x^2} + \frac{1}{h_y^2}\right) + \frac{1}{2h_x}\left(\widetilde{u}_{i,j} - \widetilde{u}_{i-1,j}\right) + \frac{1}{2h_y}\left(v_{i,j+1} - v_{i,j-1}\right)\,, \tag{3.74c}$$

$$\frac{\partial \mathcal{F}_y^{n+1}}{\partial v_{i,j+1}^{(m)}} = -\frac{\theta k\nu}{h_y^2} + \frac{1}{2h_y}\left(v_{i,j+1} + v_{i,j}\right)\,, \tag{3.74d}$$

$$\frac{\partial \mathcal{F}_y^{n+1}}{\partial v_{i+1,j}^{(m)}} = -\frac{\theta k\nu}{h_x^2}\,. \tag{3.74e}$$

We remark that these matrix elements are precisely the same as would be obtained from a properly-constructed quasilinearization as presented in Eqs. (3.56) through (3.65). We leave the task of demonstrating this for the interested reader.

The final item to consider when treating the left-hand sides of Eqs. (3.66) is introduction of first-order upwinding to guarantee satisfaction of a discrete maximum principle for equations for δu and δv. This generally leads to improved stability of the overall algorithm, but we emphasize that it does not remedy the cell-*Re* problem for the complete equations because it is actually the right-hand sides which are being solved (for u and v). Use of upwinding for the left-hand side advective terms is common practice while, as we indicated in Chap. 2, treatment of the cell-*Re* problem in general (here, in our right-hand side equations) is done in many ways.

The basic form of the first-order upwind-difference formula is given in Eq. (2.90). We repeat this here in conserved form presently being employed:

$$\left(\overline{u}_{i,j}^{(m)}\cdot\right)_x \simeq D_{\mp}\left(\overline{u}_{i,j}^{(m)}\cdot\right) \equiv \begin{cases} D_-\left(\overline{u}_{i,j}^{(m)}\cdot\right)\,, & \overline{u}_{i,j}^{(m)} > 0\,,\\ D_+\left(\overline{u}_{i,j}^{(m)}\cdot\right)\,, & \overline{u}_{i,j}^{(m)} < 0\,. \end{cases} \tag{3.75}$$

Some care must be exercised when implementing this basic formula in the context of δ-form linearization. As implied by Fig. 3.4 shown here, it is important to recall that it is now the dependent-variable increments

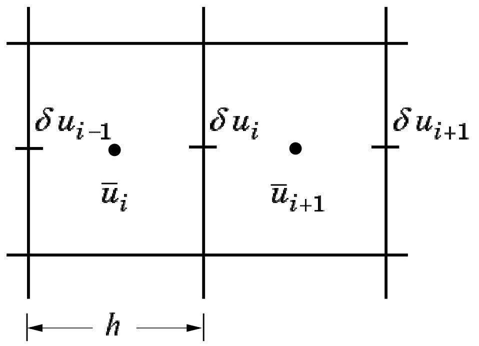

Figure 3.4: Dependent-variable arrangement for δ-form upwind differencing.

that occur at the staggered-grid points. This makes upwind differencing particularly straightforward; in particular, it is constructed across the natural pressure grid cell with the direction of differencing selected on the basis of the average velocity at the cell center. Thus, for example, if $\overline{u}_{i,j}^{(m)} > 0$, then we employ the backward difference

$$\left(\overline{u}_{i,j}^{(m)}\delta u_{i,j}\right)_x \simeq D_{-,x}\left(\overline{u}_{i,j}^{(m)}\delta u_{i,j}\right) = \frac{1}{h_x}\left(\overline{u}_{i,j}^{(m)}\delta u_{i,j} - \overline{u}_{i-1,j}^{(m)}\delta u_{i-1,j}\right)\,.$$

Similar results hold for all other discrete advective terms, derivation of which we leave as an exercise for the reader.

There are several comments of importance regarding use of upwinding in this manner. The first, and possibly most important, is that it preserves tridiagonality of system matrices; this will be particularly important for implementation of efficient time-splitting solution procedures to be discussed in the sequel. Associated with this is the fact that the Jacobian matrix elements that arise from (first-order) upwinding are somewhat simpler than those given above in Eqs. (3.73) and (3.74). Second, as we have already emphasized, use of this approximate Jacobian matrix usually produces no serious side effects other than to slightly slow convergence of Newton iterations; but this is typically minor (or even nonexistent) in the context of sequential solution algorithms as being employed herein. We point out, however, that the discrete representation of the N.–S. equations comprises a nonlinear algebraic system, and as such it can potentially possess multiple solutions—some physical, and others not. (Indeed, there is growing experimental evidence that non-unique physical solutions are possible.) This opens the possibility that inaccurate Jacobian matrices could lead to convergence of Newton iterations to a non-physical solution. We simply note this here in passing because in most cases it is evident that this does not occur.

We conclude this section with a fully-discrete representation of the Burgers'-equation form of the momentum equations that will be of use in the next section where we discuss preferred solution procedures. Here, we will employ the simplified form of upwind differencing just discussed for left-hand side matrices and the usual centered approximations on the right-hand sides. Difference-operator notation will be used throughout with the understanding that this should be replaced with detailed formulas given in the preceding analyses. We again remind the reader that the forward and backward operators appearing in the right-hand sides represent differences taken across grid cells, and thus are actually centered. The left-hand side one-sided differences are also across a grid cell; but as indicated in Fig. 3.4, they are not centered with respect to the δ variable being differenced.

Thus, if we let $\boldsymbol{u}$ represent the grid function $\{u_{i,j}\}$ for $i,j = 1,2,\ldots,N_x,N_y$, and similarly for $\delta\boldsymbol{u}$, $\boldsymbol{v}$ and $\delta\boldsymbol{v}$, we have

$$\begin{aligned}\Big\{\boldsymbol{I} - \theta k\Big[\nu\left(D_{0,x}^2 + D_{0,y}^2\right)(\,\cdot\,) - D_{\mp,x}\left(\overline{\boldsymbol{u}}^{(m)}\cdot\right) - D_{\mp,y}\left(\widetilde{\boldsymbol{v}}^{(m)}\cdot\right)\Big]\Big\}\delta\boldsymbol{u} &= \boldsymbol{u}^n - \boldsymbol{u}^{(m)}\\ &+ \theta k\Big[\nu\left(D_{0,x}^2 + D_{0,y}^2\right)\boldsymbol{u}^{(m)} - D_{+,x}\left(\overline{\boldsymbol{u}}^{(m)\,2}\right) - D_{-,y}\left(\widetilde{\boldsymbol{u}}^{(m)}\overline{\boldsymbol{v}}^{(m)}\right)\Big]\\ &+ (1-\theta)k\Big[\nu\left(D_{0,x}^2 + D_{0,y}^2\right)\boldsymbol{u}^n - D_{+,x}\left(\overline{\boldsymbol{u}}^{n\,2}\right) - D_{-,y}\left(\widetilde{\boldsymbol{u}}^n\overline{\boldsymbol{v}}^n\right)\Big]\,,\end{aligned} \tag{3.76a}$$

$$\begin{aligned}\Big\{\boldsymbol{I} - \theta k\Big[\nu\left(D_{0,x}^2 + D_{0,y}^2\right)(\,\cdot\,) - D_{\mp,x}\left(\widetilde{\boldsymbol{u}}^{(m)}\cdot\right) - D_{\mp,y}\left(\overline{\boldsymbol{v}}^{(m)}\cdot\right)\Big]\Big\}\delta\boldsymbol{v} &= \boldsymbol{v}^n - \boldsymbol{v}^{(m)}\\ &+ \theta k\Big[\nu\left(D_{0,x}^2 + D_{0,y}^2\right)\boldsymbol{v}^{(m)} - D_{-,x}\left(\widetilde{\boldsymbol{u}}^{(m)}\overline{\boldsymbol{v}}^{(m)}\right) - D_{+,y}\left(\widetilde{\boldsymbol{v}}^{(m)\,2}\right)\Big]\\ &+ (1-\theta)k\Big[\nu\left(D_{0,x}^2 + D_{0,y}^2\right)\boldsymbol{u}^n - D_{-,x}\left(\widetilde{\boldsymbol{u}}^n\overline{\boldsymbol{v}}^n\right) - D_{+,y}\left(\widetilde{\boldsymbol{v}}^{n\,2}\right)\Big]\,.\end{aligned} \tag{3.76b}$$

Douglas and Gunn Time Splitting of Momentum Equations. The basic approach we shall describe here is widely used in the context of modern projection methods, but it can easily be applied to time-dependent, multi-dimensional PDEs, in general. Indeed, it is often used as part of the solution procedure for artificial compressibility and modern MAC methods described earlier when implicit time-integration methods are employed.

We begin by recalling a few basic facts regarding this approach (more detail can be found in the original paper by Douglas and Gunn [43] and in [44]), and follow this with details of implementation in the context of δ-form quasilinearization of the N.–S. equations, as just presented. The first of these facts is that any system of PDEs, when discretized with a two-level finite-difference/finite-volume method, can be cast in the form

$$\left(\boldsymbol{I} + \boldsymbol{A}^{n+1}\right)\boldsymbol{u}^{n+1} + \boldsymbol{B}^n\boldsymbol{u}^n = \boldsymbol{s}^n\,, \tag{3.77}$$

where $\boldsymbol{u}$ is a grid function intended to provide an approximate solution to a multi-dimensional vector system of PDEs, say $\boldsymbol{u}(x,y,t) = (u(x,y,t), v(x,y,t))^T$, as in the 2-D incompressible Navier–Stokes equations. In (3.77) the symbols $\boldsymbol{I}$, $\boldsymbol{A}$ and $\boldsymbol{B}$ are matrices of appropriate dimension for the given discretization, and $\boldsymbol{s}^n$ represents a collection of all previously-known results (from the preceding time step) plus any prescribed forcing functions. Observe that the term $\boldsymbol{B}^n\boldsymbol{u}^n$ can be (and often is) included in $\boldsymbol{s}^n$.

Because we are employing a sequential solution procedure, the system (3.77) will be decomposed into separate subsystems corresponding to the two dependent variables (u and v—or, as we will see below, δu and δv). These will be solved separately with coupling provided through the same iterations used to treat nonlinearities; *i.e.*, no additional re-coupling iterations will be needed for the decomposed system. Thus, we will solve the following system, one equation at a time:

$$(\boldsymbol{I} + \boldsymbol{A}_u^{n+1})u = \boldsymbol{S}_u^n\,, \tag{3.78a}$$

$$(\boldsymbol{I} + \boldsymbol{A}_v^{n+1})v = \boldsymbol{S}_v^n\,, \tag{3.78b}$$

where $\boldsymbol{S}^n \equiv \boldsymbol{s}^n - \boldsymbol{B}^n u^n$. The matrices $\boldsymbol{A}_u^{n+1}$ and $\boldsymbol{A}_v^{n+1}$ are generally of the form of discrete Laplacians, and the goal of any time-splitting procedure is to reduce these to sequences of tridiagonal (or other compactly-banded) matrices by solving in separate (1-D) spatial directions during each fraction of a given time step. Thus, for either of the above matrices, $\boldsymbol{A}_u^{n+1}$ or $\boldsymbol{A}_v^{n+1}$, we write

$$\boldsymbol{A} = \boldsymbol{A}_x + \boldsymbol{A}_y\,, \tag{3.79}$$

where both right-hand side matrices are tridiagonal if standard second-order (or lower) discretizations have been employed on a structured grid. (We observe that this is <u>not</u> generally true in any other situation, rendering finite-element methods and/or any method(s) utilizing an unstructured grid considerably less efficient.) On the other hand, use of typical (*e.g.*, centered) higher-order accurate discretizations on a structured grid will usually lead to compactly-banded matrices (*modulo* boundary condition implementations), as is well known.

There are several different forms of the Douglas and Gunn (D–G) approach (see, *e.g.*, [44]), but the one we demonstrate here is generally used in CFD. In terms of Eq. (3.77), it is constructed as follows. First, (in 2D) define

$$\delta u^{(1)} \equiv u^{(1)} - u^n\,, \qquad \text{and} \qquad \delta u^{(2)} \equiv u^{(2)} - u^n\,. \tag{3.80}$$

In the present case of 2-D problems, $u^{n+1} = u^{(2)}$ holds. Now use Eq. (3.79) to express any scalar component of (3.77) as

$$(\boldsymbol{I} + \boldsymbol{A}_x^{n+1} + \boldsymbol{A}_y^{n+1})u^{n+1} + \boldsymbol{B}^n u^n = s^n\,,$$

which under the basic form of D–G splitting leads to

$$\begin{aligned}(\boldsymbol{I} + \boldsymbol{A}_x)u^{(1)} &= s^n - \boldsymbol{A}_y u^n - \boldsymbol{B}u^n\,,\\ (\boldsymbol{I} + \boldsymbol{A}_y)u^{(2)} &= s^n - \boldsymbol{A}_x u^{(1)} - \boldsymbol{B}u^n\,.\end{aligned}$$

Subtracting the second of these from the first leaves

$$u^{(1)} - u^{(2)} + \boldsymbol{A}_y(u^n - u^{(2)}) = 0\,,$$

and adding and subtracting u^n yields

$$u^{(1)} - u^n - u^{(2)} + u^n - \boldsymbol{A}_y \delta u^{(2)} = 0\,,$$

or

$$\delta u^{(1)} - \delta u^{(2)} - \boldsymbol{A}_y \delta u^{(2)} = 0\,,$$

and, finally,

$$(\boldsymbol{I} + \boldsymbol{A}_y)\delta u^{(2)} = \delta u^{(1)}\,.$$

This is the form taken by all equations of a D–G δ-form splitting (typically one for each spatial direction) except the first. To obtain this first equation, for $\delta u^{(1)}$, we subtract $(\boldsymbol{I} + \boldsymbol{A}_x)u^n$ from both sides of the original first split equation:

$$(\boldsymbol{I} + \boldsymbol{A}_x)u^{(1)} - \boldsymbol{A}_x u^n = s^n - (\boldsymbol{I} + \boldsymbol{A}_x + \boldsymbol{A}_y)u^n - \boldsymbol{B}^n u^n\,,$$

or

$$(\boldsymbol{I} + \boldsymbol{A}_x)\delta u^{(1)} = s^n - (\boldsymbol{I} + \boldsymbol{A})u^n - \boldsymbol{B}^n u^n\,.$$

Thus, in 2D the complete set of δ-form D–G split equations corresponding to (3.77) is

$$(\boldsymbol{I} + \boldsymbol{A}_x)\delta u^{(1)} = s^n - \boldsymbol{A}u^n - \boldsymbol{B}u^n\,, \tag{3.81a}$$

$$(\boldsymbol{I} + \boldsymbol{A}_y)\delta u^{(2)} = \delta u^{(1)}\,, \tag{3.81b}$$

and

$$u^{n+1} = u^n + \delta u^{(2)}\,. \tag{3.82}$$

Several observations are in order regarding this time-splitting procedure. First it is, of course, not necessary to perform the first split-step calculation in the x direction—the y direction is also valid—although this is typical. Second, it should be clear that each of Eqs. (3.81a) and (3.81b) consist of N_y (N_x) linear systems, each containing N_x (N_y) equations with a sparse, compactly-banded matrix representation. Thus, one pass through these equations requires only $\mathcal{O}(N)$ arithmetic operations with $N \equiv N_x N_y$, the total number of discrete points in a 2-D grid. In addition, parallelization can be easily employed within each of the two main sets of calculations represented by these equations; on theother hand, these sets of equations must be solved sequentially in order to maintain formal order of accuracy of the underlying discretizations.

Next, we observe that the right-hand side of (3.81a) is the complete original (unsplit) equation. This fact implies that each split step is consistent with the unsplit difference equation. Hence, $u^{(1)} = u^n + \delta u^{(1)}$ is a consistent approximation to u^{n+1}. We note that such consistency is somewhat rare for split schemes (*cf.*, Yanenko [78]), and in part because of this Douglas and Gunn [43] were able to prove the following basic properties of this method:

i) accuracy of the split scheme is formally of the same order in both space and time as that of the unsplit scheme;

ii) stability of the split scheme is at least as strong as that of the unsplit scheme;

iii) (not proven in [43], but fairly obvious) boundary conditions implemented as numerical modifications to any split part of the matrix $\boldsymbol{A}^{n+1}$ should always be evaluated at time level $n+1$.

We remark that failure of the first of these in many split schemes has been at least partly responsible for somewhat reduced use of these methods in recent years. On the other hand, the second property is sometimes used to improve stability of explicit methods.

Finally, the third feature is important for maintaining accuracy when implementing time-dependent boundary conditions. It is has long been understood that this poses a difficulty for split schemes. Treatment of this is given in [20], among other places, and much effort was expended in the 1970s and 80s on developing splitting methods and specifying when in time boundary conditions should be evaluated to avoid possible loss of accuracy. The basic difficulty is that, although formally each split step corresponds to a fraction of the total numerical time step, there is often no straightforward way to specify what the fraction is. It is apparent that researchers in the 70s and 80s were generally not aware of the D–G results, or if they were they did not completely comprehend them. In particular, since each split step provides a consistent approximation to the $n+1$ time-level solution, it is obvious that all boundary conditions implemented in the context of the matrix $\boldsymbol{A}^{n+1}$ should be evaluated at time t^{n+1}, and correspondingly, those associated with $\boldsymbol{B}^n$ should be evaluated at t^n, with no intermediate fractional discrete times employed.

We now present details needed to handle D–G time splitting of the discrete equations that arise from δ-form quasilinearization of (or use of Newton's method for) the N.–S. equations given earlier in Eqs. (3.76). Inspection of these shows their form, for example, for x momentum, to be

$$(\boldsymbol{I}+\boldsymbol{A}^{n+1})\delta u = s^n\,, \tag{3.83}$$

where s^n now contains time level n information associated with $\boldsymbol{B}^n u^n$ as discussed above. In addition, both $\boldsymbol{A}^{n+1}$ and s^n contain information from iteration m, but such information is completely known. With respect to the nonlinear terms, we are seeking an update to δu, say $\delta u^{(m+1)}$, such that $u^{(m+1)} = u^{(m)} + \delta u^{(m+1)}$. For notational simplicity we will here suppress this iteration index notation.

Application of the D–G procedure to (3.83) in 2D yields

$$(\boldsymbol{I}+\boldsymbol{A}_x)\delta u^{(1)} = s^n - \boldsymbol{A}_y\delta u^n\,, \tag{3.84a}$$

$$(\boldsymbol{I}+\boldsymbol{A}_y)\delta u^{(2)} = s^n - \boldsymbol{A}_x\delta u^{(1)}\,. \tag{3.84b}$$

But we observe that if quasilinear iterations are carried to convergence so that $\|\delta u^n\| < \epsilon$, then the second term on the right-hand side of (3.84a) can be neglected.

Then construction of the D–G δ form for the δ-form quasilinear equations is carried out in the same manner described above where this resulted in Eqs. (3.81). This now yields

$$(\boldsymbol{I}+\boldsymbol{A}_x)\delta u^{(1)} = s^n\,, \tag{3.85a}$$

$$(\boldsymbol{I}+\boldsymbol{A}_y)\delta u^{(2)} = \delta u^{(1)}\,, \tag{3.85b}$$

as the reader may easily show. Hence, analogous to (3.82),

$$\delta u^{(m+1)} = \delta u = \delta u^{(2)}\,,$$

and the current iteration is completed in the usual way. We note that in Eq. (3.85a) s^n is the entire right-hand side of the quasilinearized equations such as (3.76), and as has previously been stressed, this is the complete discrete equation and thus approaches zero as $m \to \infty$. Hence, it is clear that $\delta u^{(1)} \to 0$, which implies $\delta u^{(2)} = \delta u^{(m+1)} \to 0$ as $m \to \infty$, just as is the case for the unsplit discrete equations.

Boundary Condition Treatment During Quasilinear Iterations. Implementation of boundary conditions during iteration of nonlinear differential equations raises a key question: "When, during the iterations, should boundary conditions be imposed—at the first iteration, at the last (converged) iteration, or at all iterations?" Each of these alternatives has been used with some degree of success, and as the reader might expect, the level of effectiveness of any of these choices will be influenced by other details of the overall iteration scheme. Here, we will treat only the case of quasilinear (Newton) iterations in δ form, consistent with our foregoing analyses.

We first note that use of δ form implies that at each iteration only an increment to the final (hopefully) converged solution at any given time level is computed, as indicated in Eqs. (3.76). We also must recall that the right-hand sides of these equations comprise the complete discrete equation to be solved, so for a converged solution these should be zero to within a prescribed error tolerance ϵ in an appropriate norm.

These two features of δ-form quasilinearization are important in selecting an effective boundary condition treatment.

The fact that the right-hand sides are the complete equations immediately implies that boundary conditions should be implemented and evaluated in the usual way. In particular, for generalized trapezoidal time integration used here, boundary conditions for both time levels must be implemented, and these must be imposed at every iteration. Recall from the treatment of elementary two-point boundary-value problems (see, *e.g.*, Keller [79]) that the effect of this is merely to alter matrix coefficients of the difference approximation at boundary points. This, in turn, implies corresponding modifications to the Jacobian matrices appearing on the left-hand sides of (3.76). This will automatically produce δ quantities on the boundaries that are consistent with those of interior points, thus maintaining correct coupling of boundary- and interior-point solutions during the iterative solution process. Of course, in the special case of Dirichlet conditions for which solution values are presrcibed, it must be the case that $\delta \boldsymbol{u} = \delta \boldsymbol{v} = 0$ at all such boundary points. But this will occur automatically if the right-hand sides of (3.76) have been correctly constructed and the correct Jacobian matrix elements have been used.

A final note regarding boundary condition implementation pertains to the case of nonlinear boundary conditions. These are treated in considerable detail in [44]. Here, we will simply note in summary that such conditions can be treated via δ-form quasilinearization just as the differential equations have been. The results are then linear, so they are implemented in the usual way and iterated simultaneously with the nonlinrearities of the PDEs. We remark that it is in this context that quasilinearization can sometimes present some definite advantages over the usual Newton's method in terms of ease of implementation. We leave investigation if this to the reader.

Solution Filtering

The preceding development has provided a complete construction of the discrete solution operator $S_h(k)$ appearing in Eq. (3.50). The next required operation of that overall procedure is application of the filter, denoted $F(\delta_h)$. As noted in Chap. 2, there are numerous ways in which the smoothing corresponding to this might be accomplished, and in fact, many of them would be implemented within $S_h(k)$, itself. In particular, the monotone schemes typically used for shock capturing, and now becoming widely used for "implicit" large-eddy simulation (ILES) of turbulent flows—see Fureby and Grinstein [80], are of this type. Moreover, "hyperviscosity" methods (see, *e.g.*, Lamorgese, *et al.*, [81]) are now sometimes used to implement techniques similar to classical artificial dissipation. But in the context of the incompressible N.–S. equations for which a divergence-free constraint must be satisfied, the approach proposed here exhibits some advantages.

Foremost among these is the close correspondence with the 'pure-math' treatment of non-smooth solutions of PDEs, as shown in Chap. 2. In particular, the filter $F(\delta_h)$ will always be some discrete mollifier. There are, of course, many possible choices; in Chap. 2 we emphasized one that can be derived as an approximation of a widely-used analytical mollifier, partly due to its simplicity, and partly due to its performance compared with other possibilities in numerical tests (see Yang and McDonough [53]). As is also noted in Chap. 2, this Shuman filter [25] requires only $\mathcal{O}(N)$ arithmetic per time step for its application via a straightforward extension of the 1-D formula to 2D, as given in Eq. (2.120).

The disadvantage of this approach is the same as that found in the analytical application of mollifiers, namely, the error induced by this procedure. But similar errors are produced by all other methods used to treat aliasing. They are unavoidable, but in the case of the Shuman filter they are relatively easy to control—both through discretization step size (as is true with essentially any smoothing technique) and via a user-selected parameter, denoted by β in Eq. (2.120) and analogous to the support of an analytical mollifier. We repeat (2.120) here specifically for the auxiliary velocity field produced by the 2-D Burgers'-

equation form of the N.–S. momentum equations treated here.

$$\widetilde{u}_{i,j} = \frac{\widehat{u}_{i-1,j} + \widehat{u}_{i,j-1} + \beta_u \widehat{u}_{i,j} + \widehat{u}_{i,j+1} + \widehat{u}_{i+1,j}}{4 + \beta_u}\,, \tag{3.86a}$$

$$\widetilde{v}_{i,j} = \frac{\widehat{v}_{i-1,j} + \widehat{v}_{i,j-1} + \beta_v \widehat{v}_{i,j} + \widehat{v}_{i,j+1} + \widehat{v}_{i+1,j}}{4 + \beta_v}\,, \tag{3.86b}$$

$\forall\ i,j = 1,2,\ldots,N_x,N_y$. Here, β_u and β_v are filter parameters that, in general, must be provided as input to any computational algorithm; they possess the properties described in Chap. 2, and in particular are such that as $\beta \to \infty$ effects of filtering become negligible. We also note that the meaning of the short-hand symbol " $\widetilde{\ }$ " for $F(\delta_h)$ is clearly different from that of the vertical simple linear averaging needed for construction of advective terms on staggered grids, that we have used throughout these lectures. It is believed that the interpretation of this symbol should be clear from the context in which it appears.

We shall not here give further details of the use of this filter, as it was described quite thoroughly in Chap. 2, except to again emphasize that it is applied immediately after use of the discrete solution operator $S_h(k)$. Hence, it smooths a not as yet divergence-free approximation to the solution. Indeed, as we have earlier pointed out, this is the appropriate time to apply mollification since the high-wavenumber-producing aspects of the solution process are contained in the nonlinear solution operator S_h, and these had best be removed as soon as possible. Once this is done, all that remains is the linear projection step which produces no new high-wavenumber content (but which can propagate any existing high wavenumbers).

Projection to Divergence-Free Velocity

At this point we have completed calculation of $F(\delta_h)S_h(k)\boldsymbol{u}^n$, that is, $\widetilde{\widehat{\boldsymbol{u}}}$; and this does not satisfy the divergence-free condition $\nabla\cdot\boldsymbol{u} = 0$. To achieve this we must construct and apply the Leray projector P_L. This process consists of three steps, all of which have been previously treated in some detail. Here, we will briefly review these, now, in the context of the complete solution process: construction of the divergence of $\widehat{\boldsymbol{u}}$, solution of the PPE for pseudo pressure ϕ, and construction and application of the Leray projector P_L.

Divergence of $\widehat{\boldsymbol{u}}$. Construction of divergence of the velocity field earlier provided motivation for assigning locations of the dependent variables on a staggered grid (recall Fig. 2.7 and Eqs. (2.41) and (2.60)). The latter of these is expressed in cell-index notation, and in the notation of the present chapter we use this to express the divergence of $\widehat{\boldsymbol{u}}$ as

$$\widehat{D}_{i,j} \equiv \frac{1}{h_x}\left(\widehat{u}_{i,j} - \widehat{u}_{i-1,j}\right) + \frac{1}{h_y}\left(\widehat{v}_{i,j} - \widehat{v}_{i,j-1}\right)\,. \tag{3.87}$$

Then, analogous to the formal projection method construction of Eqs. (3.43)–(3.48), we have a PPE of the form

$$\Delta\phi = \frac{1}{k}\widehat{D}\,. \tag{3.88}$$

Solution of the PPE. As already noted, this is an elliptic equation for the pseudo pressure, ϕ, which is actually a velocity potential. Solution technuiques for such equations are treated in detail in [44] and numerous references therein. Here, we merely point out that despite its simplicity, successive line overrelaxation (SLOR) is still widely used. It requires minimal arithmetic per iteration, and it is readily parallelized. It also forms the basis for various more modern techniques associated with domain decomposition (see [44]), and in one form or another is typically a solution option in most commercial CFD codes, both for the PPE and for the momentum equations. In such codes SLOR is often termed "alternating-direction implicit" (ADI), and while there are some similarities it should be noted that this is actually a misuse of terms. In fact, true ADI in this context can be constructed only for 2-D problems, but SLOR can be used for any number of dimensions.

Because Eq. (3.88) is elliptic, it must have boundary conditions prescribed on all boundaries. Details of this have been given earlier for staggered-grid formulations, and we will not repeat that material here. But we remark that Neumann conditions are always imposed on solid boundaries, wheras Dirichlet conditions may be employed at open boundaries if pressure is known there, and if we are willing to accept the approximation $\phi \sim p$ to permit assigning physical values of pressure to the boundary values for ϕ. We also note, however, that such assignments are not generally mathematically valid at inflows: the N.–S. problem is formally ill posed with assignment of pressure at an inlet.

It is clear that in cases where all problem domain boundaries are solid, the PPE problem is of pure Neumann type, and as we have discussed in earlier chapters this problem is mildly ill posed (solutions are not unique). Even within the context of modifications proposed in Chap. 2, numerical methods typically employed converge rather slowly. Indeed, it is well known that numerical methods for solution of elliptic problems generally converge much faster for Dirichlet problems than for those of other types. Here we briefly describe a technique that can sometimes be useful to improve convergence rates of Neumann problems.

This begins by recalling that on a staggered grid, cell-centered solution variables such as pressure, or pseudo pressure being treated here, require definition of image-point values independent of the boundary condition type. In particular, the same image point must be used for both Dirichlet and Neumann boundary conditions (but, of course, the two types of conditions must be implemented in slightly different ways). This allows us to use image-point values from the preceding time step to construct Dirichlet boundary conditions for use at a new time step, and thus improve the convergence rate of linear elliptic iterative solvers. This can be done as follows. Assume time step n has been completed. Then the grid function $\{\phi\}_{i,j}^{N_x,N_y}$ for pseudo pressure is known at all interior points. Now if Neumann conditions have been imposed on any particular boundary, say the left one for definiteness, then the image-point values at time level n can be computed as

$$\phi_{1,j}^n = \phi_{2,j}^n \,.$$

We now use this to construct the boundary value that must exist on the actual domain boundary via linear interpolation. In particular, we have

$$\phi_{b,j}^n = \frac{1}{2}\left(\phi_{1,j}^n + \phi_{2,j}^n\right) \,,$$

which in this case is just $\phi_{2,j}^n$, itself. We now assume that we can approximate the time level $n+1$ boundary value with the time level n one. Then the PPE for ϕ can be solved as a Dirichlet problem with much higher convergence rates at time level $n+1$. Once this has been done, the entire process is formall repeated, starting with calculating the Neumann condition image-point value. But, of course, with the low-order approximation treated herein, we will always have $\phi_{1,j}^{n+1} = \phi_{2,j}^n$ ultimately resulting in $\phi_{b,j}^{n+1} = \phi_{2,j}^n$, and hence, a Dirichlet boundary condition.

But it should also be clear that this cannot be done without incurring some error. For steady-state problems this ultimately vanishes since, in fact, $\phi_{b,j}^{n+1} = \phi_{b,j}^n$; so any Dirichlet condition constructed from the same data used to produce the actual boundary value will lead to a correct result. But for time-dependent solutions this is not the case. The basic approximation in which we set the $n+1$ time level boundary value to the computed n level one clearly introduces an $\mathcal{O}(k)$ error in boundary values, and this is propagated throughout the solutions for ϕ. Thus, we can immediately conclude that if this technique of employing Dirichlet conditions in place of actual Neumann ones is used, the pseudo-pressure values will be even farther from those of true physical pressure than is otherwise true. On the other hand, error in divergence-free velocities is not as severe. In particular, in place $\nabla\phi$, we must represent the Leray projector as

$$\boldsymbol{u}^{n+1} = \widehat{\boldsymbol{u}} - k\nabla(\phi + \mathcal{O}(k)) = \widehat{\boldsymbol{u}} - k\nabla\phi + \mathcal{O}(k^2) \,,$$

so the final mass-conserved velocity field is still globally first-order accurate in time.

Construction and application of Leray projector. As we have see earlier in Chap. 1, solution of equations of the form (3.88) provide the Leray projector, P_L, as given in Eq. (3.47). In particular, we have

$$P_L\widehat{\boldsymbol{u}} = \widehat{\boldsymbol{u}} - k\nabla\phi \,. \qquad (3.89)$$

The elements of $\nabla\phi$ are easily constructed via finite differencing, but there is a subtlety which if not observed will result in an incorrect form of P_L, and consequent failure to satisfy the required divergence-free condition for $\boldsymbol{u}$. Namely, it must be the case that $\nabla\phi$ is numerically constructed from the grid function $\{\phi_{i,j}\}_{i,j=1}^{N_x,N_y}$ in such a way that the discrete form of $\nabla\cdot\nabla\phi$ is identical to the discrete form of $\Delta\phi$. If this is not true, then $\nabla\cdot$
boldsymbolu$^{n+1} = 0$ will <u>not</u> hold for $\boldsymbol{u}^{n+1}$ constructed from (3.88). We remark that this can pose difficulties when generalized coordinates are used, and with discretizations necessary for unstructured-grid implementations. In our present context this problem is, for the most part, nonexistent.

Solution of True PPE for Pressure

We have emphasized that $\phi \neq p$, although $\phi \sim p$ is often sufficiently accurate. But there are flow situations in which very accurate predictions of physical pressure are needed. Examples include flows containing free surfaces and ones involving fluid-structure interactions, especially when the structures are very responsive to applied forces. In such cases it will be necessary to solve the true PPE given in Eq. (2.30), with boundary conditions Eqs. (2.35) and (2.36), which we repeat here:

$$\Delta p = -2(u_y v_x - u_x v_y)\,, \tag{3.90}$$

with

$$\frac{\partial p}{\partial x} = -\left(u_t + (u^2)_x + (uv)_y - \nu\Delta u\right)\,, \quad \text{on vertical boundaries}\,, \tag{3.91a}$$

and

$$\frac{\partial p}{\partial y} = -\left(v_t + (uv)_x + (v^2)_y - \nu\Delta v\right)\,, \quad \text{on horizontal boundaries}\,. \tag{3.91b}$$

We should recall that (3.90) is valid only if $\boldsymbol{u}$ is divergence free, but we have already guaranteed this via Leray projection. Thus, at any desired time step Eqs. (3.90)–(3.91) can be solved using standard techniques once $\boldsymbol{u}^{n+1}$ has been produced by the discrete solution operator $P_L F(\delta_h) S_h(k)$ acting on $\boldsymbol{u}^n$.

Preparation of Results for Post Processing

Here we observe that most commercial CFD post-processing tools are built to accept grid functions defined only at <u>natural</u> unstaggered finite-difference grid points unless "hooks" are provided in the software for specific commercial flow solvers. In the case of research codes it is necessary for the code developer to transfer all results not already defined at natural finite-difference grid points to these points during the process of creating output files to be read by the post processoring software.

Clearly, there are many ways by means of which this might be done, but simple multi-linear interpolation is widely used and is typically as accurate as the original calculations in any case. Furthermore, when staggered grids, such as emphasized herein, are used, the averaged velocities $\widetilde{u}$ and $\overline{v}$ (Eqs. (3.8c) and (3.8b), respectively) serve this purpose. On the other hand, the pressure p (or pseudo pressure, ϕ) must be computed from, *e.g.*,

$$\overline{p}_{i,j} = \frac{1}{4}\left(p_{i,j} + p_{i,j+1} + p_{i+1,j} + p_{i+1,j+1}\right)\,. \tag{3.92}$$

Any other scalars that might be computed and output would be handled in an analogous way.

Projection 1 Pseudo-Language Algorithm

We are now prepared to construct a pseudo-language algorithm summarizing the methods of the preceding discussions.

Algorithm 3.4 *(Projection 1) Suppose n time steps have been completed in solving the N.–S. equations. Then the $n+1^{st}$ step is computed as follows.*

1. *Construct averaged velocity components $\overline{u}^n$, $\overline{v}^n$, $\widetilde{u}^n$, $\widetilde{v}^n$ needed to calculate time level n portion of right-hand side of Eqs.* (3.76).

2. *Calculate time level n portion of these right-hand sides, and store results, say in RHS^n.*

3. *If nonlinear iteration counter $m > 1$*
   then
   - *(a) calculate $\widetilde{u}^{(m)}$ and $\overline{v}^{(m)}$ and construct time-level $n+1$ part of x-momentum equation,* (3.76a);
   - *(b) add these to appropriate x-momentum RHS^n result computed in 2;*
   - *(c) calculate elements of left-hand side Jacobian matrix for x-momentum equation (e.g., Eqs.* (3.73)*);*
   - *(d) apply D–G time splitting to compute $\delta u^{(m+1)}$, and update u-component auxiliary velocity iterate: $\widehat{u}^{(m+1)} = \widehat{u}^{(m)} + \delta u^{(m+1)}$;*
   - *(e) calculate $\overline{u}^{(m+1)}$ and $\widetilde{u}^{(m+1)}$ and construct time-level $n+1$ part of y-momentum equation,* (3.76b);
   - *(f) repeat steps (a) through (d) for y momentum with $u \to v$, ending with the update $\widehat{v}^{(m+1)} = \widehat{v}^{(m)} + \delta v^{(m+1)}$.*

   else
   carry out steps (a) through (f) above using time-level n values as initial guesses.

4. *Calculate complete right-hand side of* (3.76) *and test convergence of quasilinear iterations:*
   if $\|RHS\| > \epsilon$, then go to 3
   else if $m > maxitr$, print error message.

5. *Mollify $\widehat{u}^{(m+1)}$ and $\widehat{v}^{(m+1)}$ using the filter $F(\delta_h)$ given in Eqs.* (3.86).

6. *Calculate divergence of $\widehat{\boldsymbol{u}}^{n+1}$ (Eq.* (3.87)*, and solve PPE for pseudo pressure ϕ using Eq.* (3.88).

7. *Project velocity field to time-level $n+1$ divergence-free field $(u^{n+1}, v^{n+1})^T$ with* (3.89).

8. *Solve true PPE (Eqs.* (3.90) *and* (3.91)*) and for physical pressure p, if needed.*

9. *Prepare computed solutions for output to post processor.*

Several remarks are worthwhile with respect to this algorithm. First, as we have previously noted, although we have performed all constructions herein in 2D, all of the results associated with projection 1 extend directly to 3D in a straightforward way, coding difficulties not withstanding. Second, it is quite clear that this algorithm is considerably more elaborate than has been true for earlier ones. This is true, first because we have provided somewhat more detail because of its intrinsic importance, but also because it is, in fact, a more complex algorithm than any of preceding ones treated herein. Moreover, it is also more intricate than the SIMPLE algorithm to be treated in the next section, and it is probably at least in part due to this that it has seen very little application in commercial CFD software. Despite this, it is well known to be the method of choice for time-dependent research codes, especially ones intended for turbulence simulations.

3.4.4 A second-order projection method

There have been numerous attempts to produce second- (and higher-) order (in time) projection methods; typical among these are the works, *e.g.*, of Gresho [72], Bell *et al.* [74] and Shen [75, 76]. Probably the simplest of these not requiring additional Poisson equation solves is that due to Shen [75] which we will describe here.

There are two key modifications that must be made to the first-order projection method treated above. The first is retention of the time-level n pressure-gradient terms in the momentum equations. For x momentum, for example, the semi-discrete form written for trapezoidal integration is

$$\widehat{u}^{n+1} = u^n - k\phi_x^n + \frac{k}{2}\left[\nu\left(\Delta\widehat{u}^{n+1} + \Delta u^n\right) - \left(\widehat{u}^2\right)_x^{n+1} - \left(u^2\right)_x^n - \left(\widehat{u}\,\widehat{v}\right)_y^{n+1} - (uv)_y^n\right] . \qquad (3.93)$$

It is important to observe that it is the pseudo pressure from time level n that appears in this equation, and not the true physical pressure. It will be apparent below that the former must be used to achieve higher-order accuracy, but at the same time not employing true pressure avoids one Poisson equation solve. Furthermore, the temporal discretization of the pressure gradient term here corresponds to a forward-Euler method, so it is very simple.

The second key difference from the first-order projection method occurs in the PPE. In this equation we now employ a δ form by defining

$$\delta\phi \equiv \phi^{n+1} - \phi^n , \qquad (3.94)$$

and solving the pressure Poisson equation

$$\Delta\delta\phi = \frac{2}{k}\nabla\cdot\widehat{u}^{n+1} \qquad (3.95)$$

for the pseudo pressure increment $\delta\phi$. Observe that the factor 2 in the right-hand side of this equation is critical for achieving improved order of accuracy, and as we will now see, it does not influence construction of the Leray projector provided it is taken into account. In particular, Leray projection is now accomplished as follows, where we now employ the vector form of this ($\boldsymbol{u} = (u,v)^T$):

$$\begin{aligned}\boldsymbol{u}^{n+1} &= \widehat{\boldsymbol{u}}^{n+1} - \frac{k}{2}\nabla\delta\phi ,\\ &= \widehat{\boldsymbol{u}}^{n+1} - \frac{k}{2}\nabla\left(\phi^{n+1} - \phi^n\right) . \qquad (3.96)\end{aligned}$$

We next demonstrate that, indeed, this fairly minor modification leads to formal second-order formal accuracy in time. We first rearrange the preceding equation to obtain en expression for $\widehat{\boldsymbol{u}}^{n+1}$ in terms of the divergence-free velocity field and the Leray projector:

$$\widehat{u}^{n+1} = u^{n+1} + \frac{k}{2}\left(\phi^{n+1} - \phi^n\right)_x , \qquad (3.97)$$

where we have written this only for the x-component of velocity. Here, we will carry out analysis only for the x-momentum equation and leave treatment of y momentum as an exercise for the reader.

We substitute (3.97) into (3.93) to obtain

$$\begin{aligned}u^{n+1} + \frac{k}{2}\left(\phi^{n+1} - \phi^n\right)_x &= u^n - k\phi_x^n + \frac{k}{2}\Bigg[\nu\left(\Delta\big(u^{n+1} + \frac{k}{2}\left(\phi^{n+1} - \phi^n\right)_x\big) + \Delta u^n\right)\\ &\quad - \left(\big(u^{n+1} + \frac{k}{2}\left(\phi^{n+1} - \phi^n\right)_x\big)^2\right)_x - \left(u^2\right)_x^n\\ &\quad - \left(\big(u^{n+1} + \frac{k}{2}\left(\phi^{n+1} - \phi^n\right)_x\big)\big(v^{n+1} + \frac{k}{2}\left(\phi^{n+1} - \phi^n\right)_y\big)\right)_y - (uv)_y^n\Bigg] .\end{aligned}$$

This can be rearranged to the form

$$\begin{aligned} u^{n+1} = u^n \; &- \frac{k}{2}\left(\phi^{n+1}+\phi^n\right)_x + \frac{k}{2}\left\{\left[\nu\Delta u - \left(u^2\right)_x - (u\,v)_y\right]^{n+1} + \left[\nu\Delta u - \left(u^2\right)_x - (u\,v)_y\right]^{n}\right\} \\ &- \frac{k^2}{4}\left(\phi^{n+1}-\phi^n\right)_x - \frac{k^2}{4}\left[u^{n+1}\left(\phi^{n+1}-\phi^n\right)_x\right]_x \\ &- \frac{k^2}{4}\left[u^{n+1}\left(\phi^{n+1}-\phi^n\right)_y + v^{n+1}\left(\phi^{n+1}-\phi^n\right)_x\right] + \mathcal{O}(k^3)\,. \end{aligned} \tag{3.98}$$

We see that the pseudo pressure gradient is now being integrated via trapezoidal quadrature; but at this point the leading truncation error terms appear to be at second order, implying only first-order global accuracy. But from the mean value theorem we have

$$\phi^{n+1} - \phi^n - \frac{\partial\phi}{\partial t}(\xi)\,k \qquad \text{for some } \xi \in [t^n, t^{n+1}]\,.$$

Thus, if $\partial\phi/\partial t$ is bounded (which it will be because the velocity components used in the right-hand side of (3.95) have been mollified), we see that the dominant (local) truncation error is $\mathcal{O}(k^3)$, and the method is second-order accurate for velocity. It is easy to show that this will also be true for both pseudo pressure ϕ and true physical pressure p. We leave demonstration of this to the reader.

3.5 The SIMPLE Algorithms

The SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) algorithm devised by Patankar and Spalding, beginning with [82] and followed by Patankar [40], in its various forms probably is the most widely used of primitive-variable methods; it is typically the core algorithm in most commercial engineering CFD codes. The method is very robust on coarse grids, but it exhibits quite low asymptotic convergence rates. Hence, it is effective for calculating rough solutions to even very complex problems, but it rapidly loses its effectiveness as grids are refined to achieve more spatial accuracy. The method is constructed as a time-stepping procedure, as has been true of all our previously-discussed algorithms, but it is too inefficient to be used for time-dependent problems containing very many spatial grid points.

Through the years there have been numerous modifications to the basic SIMPLE procedure presented in [40]—some intended to remove some of the assumptions we will discuss in the sequel, and some specifically intended to improve performance for time-dependent problems. In the present notes we shall not attempt to provide an encyclopedic coverage of these, and instead focus on a fairly complete treatment of the earliest basic methods given by Patankar in [40], namely, SIMPLE and SIMPLER standing for SIMPLE-Revised.

In the following sections we will first present an analytical form of SIMPLE, as we have previously done for the projection method, thus providing some basic heuristics that are more difficult to perceive in a completely discrete treatment. We will follow this with such treatments, first for SIMPLE, and then for SIMPLER.

3.5.1 An analytical version of SIMPLE

SIMPLE is a finite-volume based scheme, utilizing a staggered grid (although there are now many cell-centered unstaggered implementations—especially in commercial flow codes), and as we will see below, it is derived, in part, in terms of discrete equations (especially with respect to the pressure-velocity "link") rather than from discretizations of a continuous formulation. The details of this are quite tedious—SIMPLE is really not very simple! So before embarking on this we will provide an heuristic treatment in terms of continuous equations following that given by Comini and Del Giudice [83].

The basic assumption employed in constructing the SIMPLE algorithm is that at any given time step the flow variables can be expressed as (in 2D)

$$u = u^* + u'\,, \quad v = v^* + v' \quad \text{and} \quad p = p^* + p'\,, \tag{3.99}$$

where " $*$ " denotes an initial estimate, and " $'$ " represents a correction. Formally, all parts of these decompositions are functions of both space and time. Thus, we can substitute these into the Navier–Stokes equations, say, of the form Eqs. (3.1) and (3.2) to obtain for x momentum, for example,

$$(u^* + u')_t + \left((u^* + u')^2\right)_x + \left((v^* + v')(u^* + u')\right)_y = -(p^* + p')_x + \nu\Delta(u^* + u') \,. \tag{3.100}$$

We next ignore effects of the corrections on advective and diffusive terms, and linearize about the previous time level n to arrive at

$$u_t^* + u_t' + (u^n\, u^*)_x + (v^n\, u^*)_y = -p_x^n - p_x' + \nu\Delta u^* \,, \tag{3.101}$$

where we have also set $p^* = p^n$. We now apply a Yanenko-like splitting in much the same way as was done earlier in deriving the projection method. This leads to

$$u_t^* + (u^n\, u^*)_x + (v^n\, u^*)_y = -p_x^n + \nu\Delta u^* \,, \tag{3.102a}$$

and

$$u_t' = -p_x' \,. \tag{3.102b}$$

A completely analogous set of equations is obtained for y momentum:

$$v_t^* + (u^n\, v^*)_x + (v^n\, v^*)_y = -p_y^n + \nu\Delta v^* \,, \tag{3.103a}$$

and

$$v_t' = -p_y' \,. \tag{3.103b}$$

Although Eqs. (3.102a) and (3.103a) can be easily solved using pressure from time level n, this will not lead to mass conservation at time level $n+1$. To obtain a divergence-free velocity field at the advanced time level we use Eqs. (3.102b), (3.103b) to develop a pressure Poisson equation as follows. Let

$$D' = u_x' + v_y' \,;$$

then differentiation with respect to time leads to

$$\frac{\partial D'}{\partial t} = \frac{\partial}{\partial t}\left(u_x' + v_y'\right) = u_{tx}' + v_{ty}' \,. \tag{3.104}$$

Now, similar to Eqs. (3.99), we have $D = D^* + D'$, and at the end of time step n we should have $D = 0$, implying that $D' = -D^*$. From the left-hand side of (3.104), using a simple forward-difference approximation, we have

$$\frac{\partial D'}{\partial t} = \frac{D'^{\,n+1} - D'^{\,n}}{\Delta t} \,.$$

For $n \to \infty$ we would expect $D^* \to D \to 0$, and it follows that $D'^{\,n} = 0$ should hold. Thus, we can write the above as

$$\frac{\partial D'}{\partial t} \simeq \frac{D'^{\,n+1}}{\Delta t} \simeq -\frac{D^{*\,n+1}}{\Delta t} \,. \tag{3.105}$$

We mention here that there is an alternative way in which this can be argued, based on the form of the solution algorithm to be presented below. Namely, we will see that this algorithm is constructed so that $D = 0$ to some specified tolerance at the end of each (pseudo) time step. But in addition, the iterations are performed such that $D^* \to D$ within each time step. This, in turn, implies that $D'^{\,n} = 0$.

We next use (3.102b), (3.103b) in the right-hand side of Eq. (3.104) to obtain

$$\frac{\partial D'}{\partial t} = -p_{xx}' - p_{yy}' \,,$$

and equating this to the right-hand side of (3.105) gives the desired result, a PPE for the correction pressure:

$$\Delta p' = \frac{D^*}{\Delta t}\,, \tag{3.106}$$

where we have suppressed the now superfluous time level notation.

The boundary conditions to be employed with this PPE are

$$p' = 0$$

at boundary points where pressure has been assigned, and

$$\frac{\partial p'}{\partial n} = 0$$

at all other boundary points. The similarity of these conditions, and that of the complete PPE problem, to those employed for the pseudo pressure ϕ in projection methods should be noted.

Once p' has been calculated we can then update the pressure and velocity components using

$$p^{n+1} = p^n + p'\,, \tag{3.107a}$$

$$u^{n+1} = u^* - \frac{\partial p'}{\partial x}\Delta t \equiv u^* + u'\,, \tag{3.107b}$$

$$v^{n+1} = v^* - \frac{\partial p'}{\partial y}\Delta t \equiv v^* + v'\,. \tag{3.107c}$$

The last two of these equations are derived by formally discretizing Eqs. (3.102b) and (3.103b). It is of interest to note that Eqs. (3.107) are quite similar to the SOLA updates developed earlier in Eqs. (3.24) and following. In fact, if (3.106) is discretized in the standard way and all off-diagonal terms then discarded, the result is the SOLA pressure update, Eq. (3.25). (Recall that this was derived as a "diagonal" Newton method during analysis of SOLA.)

We see from this that SIMPLE is structurally similar to both a projection method (which incorporates a PPE solve) and to SOLA (which only uses pressure and velocity updates—without a PPE solve). However, it is important to recognize that the analysis just carried out with the continuous equations does not translate into the discrete numerical algorithm known as SIMPLE in a completely direct way. Our next task will be to formulate a discrete version of the above equations and to produce a corresponding computational procedure.

3.5.2 The discrete SIMPLE algorithm

We will begin by discretizing Eqs. (3.1) in time with a backward-Euler method, and in space via centered differencing, and we linearize advective terms about the time-level n solution when computing time-level $n+1$ results. This leads to the system of discrete equations introduced earlier as Eqs. (3.39) in the context of the artificial compressibility method, which we repeat here for convenience:

$$\left\{1 - k\Big[\nu\left(D^2_{0,x} + D^2_{0,y}\right) - D_{+,x}\left(\overline{u}^n_{i,j}\,\cdot\right) - D_{-,y}\left(\overline{v}^n_{i,j}\,\cdot\right)\Big]\right\}u^{n+1}_{i,j} = u^n_{i,j} - kD_{+,x}p_{i,j}\,, \tag{3.108a}$$

$$\left\{1 - k\Big[\nu\left(D^2_{0,x} + D^2_{0,y}\right) - D_{+,x}\left(\widetilde{u}^n_{i,j}\,\cdot\right) - D_{-,y}\left(\widetilde{v}^n_{i,j}\,\cdot\right)\Big]\right\}v^{n+1}_{i,j} = v^n_{i,j} - kD_{+,y}p_{i,j}\,. \tag{3.108b}$$

As can be seen from the notation employed in these equations, we have chosen to retain the typical numerical-analytic (i,j) indexing rather than use the juvenile "EWNS" notation found in [40] and in most other engineering treatments of SIMPLE and similar algorithms. We remark that (i,j) indexing is more compatible with formal programming languages, it is easier to work with in terms of formal analysis (*e.g.*, truncation error), and it is readily generalized to 3D.

Momentum Equations

We will treat only the x-momentum equation in detail; expanding the difference operators in (3.108a) using uniform grid spacing h and time step k results in

$$u_{i,j} - \frac{k\nu}{h^2}\Big(u_{i-1,j} + u_{i,j-1} - 4u_{i,j} + u_{i,j+1} + u_{i+1,j}\Big) + \frac{k}{h}\Big(\overline{u}^n_{i+1,j}\overline{u}_{i+1,j} - \overline{u}^n_{i,j}\overline{u}_{i,j}\Big)$$
$$+ \frac{k}{h}\Big(\overline{v}^n_{i,j}\widetilde{u}_{i,j} - \overline{v}^n_{i,j-1}\widetilde{u}_{i,j-1}\Big) = u^n_{i,j} - \frac{k}{h}\Big(p_{i+1,j} - p_{i,j}\Big)\,, \qquad (3.109)$$

where variables without superscripts denote advanced time level results to be computed. We also recall that " $-$ " and " $\sim$ " correspond to horizontal and vertical averages, respectively, introduced earlier for treatment of advective terms on staggered grids. Application of formulas for these averages (Eqs. (3.8)), and collecting terms of like spatial subscripts, permits us to express this in the compact form

$$-A_1u_{i-1,j} - A_2u_{i,j-1} + A_3u_{i,j} - A_4u_{i,j+1} - A_5u_{i+1,j} = b_{i,j} - \frac{k}{h}\Big(p_{i+1,j} - p_{i,j}\Big)\,, \qquad (3.110)$$

with the various coefficients given as follows:

$$A_1 = \frac{k}{h}\left(\frac{\nu}{h} + \frac{1}{2}\overline{u}^n_{i,j}\right)\,, \qquad (3.111a)$$

$$A_2 = \frac{k}{h}\left(\frac{\nu}{h} + \frac{1}{2}\overline{v}^n_{i,j-1}\right)\,, \qquad (3.111b)$$

$$A_3 = 1 + 4\frac{k\nu}{h^2} + \frac{k}{2h}\left(\overline{u}^n_{i+1,j} - \overline{u}^n_{i,j} + \overline{v}^n_{i,j} - \overline{v}^n_{i,j-1}\right)\,, \qquad (3.111c)$$

$$A_4 = \frac{k}{h}\left(\frac{\nu}{h} - \frac{1}{2}\overline{v}^n_{i,j}\right)\,, \qquad (3.111d)$$

$$A_5 = \frac{k}{h}\left(\frac{\nu}{h} - \frac{1}{2}\overline{u}^n_{i+1,j}\right)\,, \qquad (3.111e)$$

and $b_{i,j} = u^n_{i,j}$. It is clear from these formulas that the A_ℓs generally are nonconstant and, thus, formally should contain i,j subscripts; and if time-dependent solutions are to be computed, they must be reevaluated at each iteration within a time step. In [40] (3.110) is solved for the diagonal term to obtain

$$A_3u_{i,j} = \sum_{\substack{\ell=1\\ \ell\neq3}}^{5} A_\ell u_{i(\ell),j(\ell)} + b_{i,j} - \frac{k}{h}\Big(p_{i+1,j} - p_{i,j}\Big)\,, \qquad (3.112)$$

where the values of $i(\ell)$ and $j(\ell)$ can be read from Eq. (3.110).

The next step in constructing the SIMPLE algorithm is to recall from Eq. (3.99) that we are decomposing the dependent variables as, for example, $u = u^* + u'$. From our treatment of the continuous form of SIMPLE we expect

$$A_3u^*_{i,j} = \sum_\ell A_\ell u^*_{i(\ell),j(\ell)} + b_{i,j} - \frac{k}{h}\Big(p^*_{i+1,j} - p^*_{i,j}\Big) \qquad (3.113)$$

should hold (because the equations for the " $*$ " quantities are of the same form as those for the total quantities), and subtracting this from (3.112) results in

$$A_3u'_{i,j} = \sum_\ell A_\ell u'_{i(\ell),j(\ell)} - \frac{k}{h}\Big(p'_{i+1,j} - p'_{i,j}\Big)\,.$$

Now also recall that in the continuous equations we ignored all effects of correction quantities in advective and diffusive terms. It is easy to see that this results in eliminating the summation in the above expression, and we arrive at the u-component velocity correction in the form

$$u'_{i,j} = \frac{k}{A_3\,h}\Big(p'_{i,j} - p'_{i+1,j}\Big)\,. \qquad (3.114)$$

We should note that although this is of the same general form as would be obtained via direct discretization of (3.102b)—with the assumption that u'^n, it is quite different in detail. In particular, use of Eq. (3.102b) would lead to simply k/h for the coefficient of the pressure difference, but here it contains the complicated factor A_3 multiplying h in the denominator. As can be seen from Eq. (3.111c) this additional factor is nontrivial, and its presence contributes to the numerical performance of SIMPLE being rather different from that of projection methods.

We now present analogous results for the y-momentum equation, leaving derivation of these as an exercise for the reader.

$$-B_1 u_{i-1,j} - B_2 u_{i,j-1} + B_3 u_{i,j} - B_4 u_{i,j+1} - B_5 u_{i+1,j} = c_{i,j} - \frac{k}{h}\Big(p_{i,j+1} - p_{i,j}\Big)\,, \tag{3.115}$$

with the various coefficients given as

$$B_1 = \frac{k}{h}\left(\frac{\nu}{h} + \frac{1}{2}\widetilde{u}^n_{i-1,j}\right)\,, \tag{3.116a}$$

$$B_2 = \frac{k}{h}\left(\frac{\nu}{h} + \frac{1}{2}\widetilde{v}^n_{i,j}\right)\,, \tag{3.116b}$$

$$B_3 = 1 + 4\,\frac{k\nu}{h^2} + \frac{k}{2h}\left(\widetilde{u}^n_{i,j} - \widetilde{u}^n_{i-1,j} + \widetilde{v}^n_{i,j+1} - \widetilde{v}^n_{i,j}\right)\,, \tag{3.116c}$$

$$B_4 = \frac{k}{h}\left(\frac{\nu}{h} - \frac{1}{2}\widetilde{v}^n_{i,j+1}\right)\,, \tag{3.116d}$$

$$B_5 = \frac{k}{h}\left(\frac{\nu}{h} - \frac{1}{2}\widetilde{u}^n_{i,j}\right)\,, \tag{3.116e}$$

and $c_{i,j} = v^n_{i,j}$. Then the equation for v-component velocity predictions is

$$B_3 v^*_{i,j} = \sum_\ell B_\ell v^*_{i(\ell),j(\ell)} + c_{i,j} - \frac{k}{h}\Big(p^*_{i,j+1} - p^*_{i,j}\Big)\,, \tag{3.117}$$

and that for the corrections is

$$v'_{i,j} = \frac{k}{B_3\,h}\Big(p'_{i,j} - p'_{i,j+1}\Big)\,. \tag{3.118}$$

Pressure Poisson Equation

The remaining equation needed to complete the SIMPLE algorithm is that for the pressure corrections, p'. This is constructed by substituting the velocity corrections, (3.114) and (3.118), into the discrete continuity equation for grid cell (i,j) given as

$$\frac{u_{i,j} - u_{i-1,j}}{h_x} + \frac{v_{i,j} - v_{i,j-1}}{h_y} = 0\,,$$

or after substituting the decomposed velocity components, Eqs. (3.99),

$$\frac{(u^* + u')_{i,j} - (u^* + u')_{i-1,j}}{h_x} + \frac{(v^* + v')_{i,j} - (v^* + v')_{i,j-1}}{h_y} = 0\,.$$

For simplicitiy we will set $h_x = h_y = h$, and rewrite this as

$$\frac{1}{h}\Big(u'_{i,j} - u'_{i-1,j} + v'_{i,j} - v'_{i,j-1}\Big) = -\frac{1}{h}\Big(u^*_{i,j} - u^*_{i-1,j} + v^*_{i,j} - v^*_{i,j-1}\Big) \equiv -D^*_{i,j}\,. \tag{3.119}$$

Then using Eqs. (3.114) and (3.118) in the left-hand side of (3.119) leads to

$$\frac{k}{A_{3,i,j}\,h^2}\Big(p'_{i,j} - p'_{i+1,j}\Big) - \frac{k}{A_{3,i-1,j}\,h^2}\Big(p'_{i-1,j} - p'_{i,j}\Big) + \frac{k}{B_{3,i,j}\,h^2}\Big(p'_{i,j} - p'_{i,j+1}\Big) - \frac{k}{B_{3,i,j-1}\,h^2}\Big(p'_{i,j-1} - p'_{i,j}\Big) = -D^*_{i,j}\,,$$

or

$$-\frac{1}{A_{3,i-1,j}}p'_{i-1,j} - \frac{1}{B_{3,i,j-1}}p'_{i,j-1} + \left(\frac{1}{A_{3,i,j}} + \frac{1}{A_{3,i-1,j}} + \frac{1}{B_{3,i,j}} + \frac{1}{B_{3,i,j-1}}\right)p'_{i,j}$$
$$- \frac{1}{B_{3,i,j}}p'_{i,j+1} - \frac{1}{A_{3,i,j}}p'_{i+1,j} = -\frac{h^2}{k}D^*_{i,j}\,, \tag{3.120}$$

where we have explicitly retained cell indexing for the various A and B coefficients to emphasize that these are not constant, and moreover, that they must be inserted at the correct locations with respect to indices on p' in (3.120) to produce accurate results.

We now express (3.120) in a more compact form similar to that used for the momentum equations:

$$C_1p'_{i-1,j} + C_2p'_{i,j-1} + C_3p'_{i,j} + C_4p'_{i,j+1} + C_5p'_{i+1,j} = d^*_{i,j}\,, \tag{3.121}$$

with the various coefficients defined as follows.

$$C_1 \equiv \frac{1}{A_{3,i-1,j}}\,, \qquad C_2 \equiv \frac{1}{B_{3,i,j-1}}\,, \qquad C_3 \equiv -\left(\frac{1}{A_{3,i,j}} + \frac{1}{A_{3,i-1,j}} + \frac{1}{B_{3,i,j}} + \frac{1}{B_{3,i,j-1}}\right)\,,$$
$$C_4 \equiv \frac{1}{B_{3,i,j}}\,, \qquad C_5 \equiv \frac{1}{A_{3,i,j}}\,, \qquad d^*_{i,j} \equiv \frac{h^2}{k}D^*_{i,j}\,. \tag{3.122}$$

It should be observed that Eq. (3.121) is of the general form of a 2-D discrete Laplacian, as would be expected from our treatment involving the continuous form of SIMPLE; but the coefficients are not constant, in general. Thus, the discrete pressure correction equation would correspond to a more general elliptic PDE if the continuous version were recovered. But it is worth noting that, as is true for the discrete Laplacian, the sum of the off-diagonal coefficients equals the diagonal coefficient, to within a sign, which would not necessarily be true for discretization of a general elliptic operator not in self-adjoint form.

It is evident that carrying out the derivation in terms of discrete equations, as done here, can produce results that are rather different from those obtained by merely discretizing the continuous PDEs. Clearly, whenever this is the case, consistency of the discrete approximation with the original continuous equation(s) must be a concern. In the present case, however, the goal is to satisfy the divergence-free constraint imposed on the velocity field, so consistency, *per se*, with a true pressure Poisson equation is less an issue, just as is the situation when using usual projection methods. Furthermore, it is clear from the derivation leading to Eqs. (3.113) and (3.117) that the discrete momentum equations are, in fact, consistent with the continuous Navier–Stokes equations up to possible errors in the pressure. But as is also clear, the pressure arising from Eq. (3.121) is not the true physical pressure, and it is not clear that estimates such as those given in Eq. (3.48) for the projection method can be demonstrated for SIMPLE. In particular, suppose p^n is an accurate physical pressure. The nature of the pressure update in SIMPLE is such that

$$p^{n+1} = p^n + \sum_m p'^{(m)}\,,$$

where none of the p's produced during the iteration are necessarily close to physical pressure. Among other difficulties, they do not satisfy boundary conditions for physical pressure. It would thus seem that a useful remedy would be to solve the true pressure Poisson equation (3.90) with correct boundary conditions at each time step for which accurate pressures are needed, just as we recommended in some cases for projection methods. (But note that this would require a divergence-free velocity field—something not usually achieved with SIMPLE until steady state is reached, as will be evident from the pseudo-language algorithm of the next section.)

At this juncture we have developed all of the equations needed to calculate the predictions u^* and v^* as well as the pressure correction p'. What remains is to formulate the updates needed to proceed from one iteration to the next during a (pseudo) time step. These are similar to what was presented earlier for the continuous form of SIMPLE, but they are not identical. In the discrete form considered here we have the following.

$$p_{i,j}^{(m+1)} = p_{i,j}^{(m)} + \alpha_p p'_{i,j} \,, \tag{3.123a}$$

$$u_{i,j}^{(m+1)} = u^*_{i,j} + \alpha_u \frac{k}{A_{3,i,j}\, h}\left(p'_{i,j} - p'_{i+1,j}\right) , \tag{3.123b}$$

$$v_{i,j}^{(m+1)} = v^*_{i,j} + \alpha_v \frac{k}{B_{3,i,j}\, h}\left(p'_{i,j} - p'_{i,j+1}\right) . \tag{3.123c}$$

In these equations the α s are iteration (under-relaxation) parameters with values in the range $0 < \alpha \leq 1$; but there is no theoretical underpinning to suggest specific values to be assigned. Values between 0.5 and 0.8 are widely used because they usually work, but smaller values are sometimes needed. It should also be noted that there is no reason to expect $\alpha_p = \alpha_u = \alpha_v$ to hold, and in fact, typically $\alpha_p < \alpha_u\,(\alpha_v)$ must be used.

There are several important aspects of these updates that should be noted. The first is that the pressure update is in exactly the same form as found in the continuous version of SIMPLE, except for the presence of the iteration parameter. Furthermore, it is rather similar (without the iteration parameter) to calculation of pseudo pressure in Shen's second-order projection method [75] described earlier. Presence of these parameters in the velocity updates makes it clear that a single iteration cannot, in general, produce a divergence-free velocity field because they do not appear in the discrete Laplacian being solved for p'—unless, of course, α_u and α_v both equal unity. But this does not usually lead to convergence of the overall SIMPLE algorithm—probably due to the somewhat naïve iteration method (Picard) applied to the nonlinear terms of the momentum equations (recall Eqs. (3.113) and (3.117)). On the other hand, taking the discrete divergence of Eqs. (3.123b) and (3.123c) will lead to Eq. (3.120) in a straightforward way if $\alpha_u = \alpha_v = 1$. Thus, in the absence of relaxation parameters, the velocity updates of Eqs. (3.123) would provide slightly complicated projectors in the sense of Leray. Indeed, the only aspect of SIMPLE leading to a divergence-free velocity field is derivation of (3.120) directly from the discrete continuity equation, rather than from the momentum equations as was done, *e.g.*, in the MAC method, and one should expect a close correspondence with Leray projection.

SIMPLE Pseudo-Language Algorithm

We now have on hand all of the discrete equations needed to construct the complete SIMPLE pseudo-language algorithm.

Algorithm 3.5 *(SIMPLE) Suppose n pseudo-time steps have been completed. To calculate step $n+1$ level results, carry out the following steps.*

```
1. With linearization about current velocity iterate, load x-momentum equation coefficients using Eqs. (3.111).

2. Using the current iterate for pressure (or, $p^n$ for first iteration), solve discrete x-momentum equations (3.112) for $u^*$.

3. Repeat steps 1 and 2 for y momentum, using, respectively, Eqs. (3.116) and (3.117).

4. Evaluate $D^*$ from RHS of Eq. (3.119).

5. Test convergence:

      If $\|D^*\| < \epsilon\|D^n\|$, then stop.

            else if $\|D^*\| < \delta\|D^n\|$, start new pseudo-time step.
```

6. *Solve Poisson equation* (3.121) *for pressure correction* p'.

7. *Calculate updated velocity components and pressure from Eqs.* (3.123)*, and return to step 1.*

There are several items that should be noted with regard to this algorithm. The first is that, as we have emphasized throughout, SIMPLE should only be used for steady-state calculations, so we have posed this algorithm in terms of pseudo-time marching rather than as a time-accurate procedure. Second, we have placed little emphasis on solution techniques to be used in solving Eqs. (3.112), (3.117) and (3.121). It is easily seen that these are all in the form of discrete Laplacians, and in the context of the SIMPLE algorithm construction typical Poisson solvers would appear to be appropriate. When SIMPLE was initially introduced, Patankar typically recommended use of successive line overrelaxation for all equations, and this still is not an unreasonable approach to employ. But there are many other modern alternatives that may, in some cases, be more effective—*e.g.*, multigrid and domain decomposition techniques. Moreover, it is quite possible that the overall algorithm could be accelerated significantly if Douglas and Gunn time splitting were used for the momentum equation solves. The third item is associated with the form of convergence test employed. This again reflects the pseudo-time nature of the algorithm. In particular, during each pseudo-time step it typically is not effective to perform iterations until divergence is reduced to a small absolute value. Rather, in the context of SIMPLE, a relative convergence criterion is used within individual pseudo-time steps, along with a smaller overall tolerance. But even this smaller tolerance is typically relative, and moreover, in practice often fairly large—say, $\mathcal{O}(10^{-3})$ or larger. This is in contrast to projection methods which produce divergence-free velocity fields to high precision at every time step; often machine precision is employed, although this seems rather difficult to justify.

3.5.3 The SIMPLER algorithm

In this section we will treat the first, and still one of the more widely-used, modifications to the basic SIMPLE algorithm. SIMPLER, as we have earlier noted, stands for SIMPLE-Revised; it was available shortly after the introduction of SIMPLE and is discussed in fair detail in [40]. It had been observed very early that the pressure corrections used in the updates (3.123) of SIMPLE appeared to provide adequate corrections for velocity components, but were often quite inaccurate as updates for pressure. This led Patankar to seek an algorithmic structure that would retain the part of SIMPLE associated with velocity calculations, but replace the method used to obtain pressure. It will be apparent that the resulting procedure begins as a projection 1 method, which is used to produce pressure; then the algorithm reverts to SIMPLE to complete the velocity calculations.

Here, we will begin with a brief treatment of the analytical form presented by Comini and De Giudice [83], as we have previously done for the basic SIMPLE procedure. We follow this with a short analysis of the additional equations that must be used in the discrete form of this technique, and we conclude with a pseudo-language algorithm for SIMPLER.

Analytical Representation of SIMPLER

The first step in SIMPLER is calculation of an auxiliary velocity field, just as is done in typical projection algorithms. The momentum equations are expressed in the same form used earlier for projection 1. Thus, we have

$$\widehat{u}_t + (u^n\widehat{u})_x + (v^n\widehat{u})_y = \nu\Delta\widehat{u}\,, \tag{3.124a}$$

$$\widehat{v}_t + (u^n\widehat{v})_x + (v^n\widehat{v})_y = \nu\Delta\widehat{v}\,. \tag{3.124b}$$

We note, as already suggested, that no pressure-gradient terms appear in these equations, but in the context of SIMPLER, no quasilinear iterations are performed.

A Yanenko-like splitting is still employed—now identical to that used in projection 1—leading to

$$u'_t = -p_x\,, \qquad \text{and} \qquad v'_t = -p_y\,, \tag{3.125}$$

which can be obtained by setting

$$u = \widehat{u} + u' \qquad \text{and} \qquad v = \widehat{v} + v' \,, \tag{3.126}$$

and linearizing about the current time level in the original momentum equations (3.1). Then, similar to the approach employed to construct SIMPLE, we take the divergence of Eqs. (3.125) and from (3.126) deduce that $D' = -\widehat{D}$ to arrive at the pressure Poisson equation

$$\Delta p = \frac{\widehat{D}}{k} \,. \tag{3.127}$$

The boundary conditions used in this PPE are not the same as used for the pressure correction in SIMPLE; in fact, they are identical to those typically employed in a projection 1 procedure, namely, at boundary points where (physical) pressure is known, it is assigned; and elsewhere $\partial p/\partial n = 0$ is used. The pressure obtained from this problem will then be set to p^* in SIMPLE, but it will not be corrected. We see from this that pressures predicted by SIMPLER are exactly the same as those obtained from projection 1, which we previously noted are often sufficiently accurate, but are not the exact, physical pressures.

The remaining steps of SIMPLER consist of following the basic SIMPLE algorithm, except for correcting pressure. In particular, momentum equations are solved for u^* and v^*, but now using pressure obtained as described above. Then p' is computed and used to correct the estimated velocities, as in SIMPLE. We remark that projecting $\widehat{u}$ and $\widehat{v}$ onto a divergence-free subspace would provide far better results than do the iterations involving u^* and v^*. In fact, if this were done—and it requires very little additional arithmetic—then one could estimate u^* with $\widehat{u}$, and similarly for v^* in the nonlinear terms of the momentum equations, resulting in new computed u^* and v^* that would be nearly divergence free. This would significantly decrease the required iterations, and conceivably might permit use of relaxation parameters much closer to unity. But clearly, SIMPLER would still lack the efficiency (and probably accuracy) of a true projection method.

Discrete Form of SIMPLER

As noted above in the analytical description of SIMPLER, the first step is to solve discrete forms of the momentum equations (3.124) written without the pressure-gradient terms. These can be expressed in a form analogous to the equations solved for u^* and v^* in the basic SIMPLE algorithm:

$$\widehat{u}_{i,j} = \frac{1}{A_3}\left[\sum_\ell A_\ell\, \widehat{u}_{i(\ell),j(\ell)} + b_{i,j}\right] , \tag{3.128a}$$

$$\widehat{v}_{i,j} = \frac{1}{B_3}\left[\sum_\ell B_\ell\, \widehat{v}_{i(\ell),j(\ell)} + c_{i,j}\right] , \tag{3.128b}$$

with coefficients identical to those given for SIMPLE in Eqs. (3.111) and (3.116). We also can use approximations analogous to those of SIMPLE to produce the velocity update formulas

$$u_{i,j} = \widehat{u} + \frac{k}{A_3\, h}\big(p_{i,j} - p_{i+1,j}\big) \,, \tag{3.129a}$$

$$v_{i,j} = \widehat{v} + \frac{k}{B_3\, h}\big(p_{i,j} - p_{i,j+1}\big) \,. \tag{3.129b}$$

These can now be substituted into the discrete continuity equation, as was done for the SIMPLE algorithm, to produce the pressure Poisson equation for the "physical" pressure:

$$C_1 p_{i-1,j} + C_2 p_{i,j-1} + C_3 p_{i,j} + C_4 p_{i,j+1} + C_5 p_{i+1,j} = \frac{h^2}{k}\widehat{D}_{i,j} \,, \tag{3.130}$$

with

$$\widehat{D}_{i,j} \equiv \frac{1}{h}\big(\widehat{u}_{i,j} - \widehat{u}_{i-1,j} + \widehat{v}_{i,j} - \widehat{v}_{i,j-1}\big) \,, \tag{3.131}$$

and all C_ℓs the same as those used in the pressure-correction equation, Eqs. (3.122).

As we have indicated in discussing the analytical form of SIMPLER, the next step will be to solve the SIMPLE momentum equations (3.112) and (3.117) for the velocity predictions u^* and v^*, respectively, but now with $p^* = p$ in the pressure-gradient terms. This, in turn, leads to construction of the divergence of this predicted velocity field which is then used as the right-hand side of the pressure-correction equation, (3.121) of SIMPLE. The pressure correction obtained from this equation is used to update the velocity field using Eqs. (3.123b) and (3.123c), exactly is done in SIMPLE; but now in the case of SIMPLER, no further update of the pressure field is done.

Pseudo-Language Algorithm for SIMPLER

We have now completed presentation of all the discrete equations needed to solve the N.–S. equations using SIMPLER, and we here provide a pseudo-language algorithm for its implementation.

Algorithm 3.6 *(SIMPLER) Suppose n pseudo-time steps have been completed. To calculate the solution at step $n+1$ carry out the following.*

1. *Calculate coefficients for the momentum equations (Eqs.* (3.111) *and* (3.116)*) using current iterate (or time level n results for first iteration of new time step).*

2. *Sequentially solve momentum equations without pressure-gradient terms (Eqs.* (3.128)*) for auxiliary velocities $\{\widehat{u}_{i,j}\}$ and $\{\widehat{v}_{i,j}\}$.*

3. *Calculate divergence $\{\widehat{D}_{i,j}\}$ of auxiliary velocity field using* (3.131) *and coefficients C_ℓ (Eqs.* (3.122)*) of the PPE, and solve for pressure $\{p_{i,j}\}$.*

4. *Set $\{p^*_{i,j}\} = \{p_{i,j}\}$ in momentum equations used in SIMPLE (Eqs.* (3.112) *and* (3.117)*) and solve for $\{u^*_{i,j}\}$ and $\{v^*_{i,j}\}$.*

5. *Calculate $D^*_{i,j}$, and test convergence:*

   If $\|D^*\| < \epsilon \|D^n\|$ stop.

6. *Solve PPE for pressure correction (Eq.* (3.121)*) to obtain $\{p'_{i,j}\}$.*

7. *Correct velocity components as in SIMPLE, using Eqs.* (3.123b) *and* (3.123c)*; but do <u>not</u> correct pressure. Return to step 1 to start new pseudo-time step.*

There are several important items to note regarding SIMPLER. First, it is fairly obvious that the required arithmetic per iteration is approximately double that of basic SIMPLE. In turn, this implies that unless the convergence rate of SIMPLER is at least double that of SIMPLE, there is little justification for its use. In fact, in some cases this has been observed, but not in general. Second, as we have previously noted, the first part of SIMPLER is precisely projection 1, but the algorithm continues on with the usual steps of SIMPLE. An obvious consequence of this, as noted in [83] is that SIMPLER is the least efficient of all possible projection methods. In particular, it requires nearly double the arithmetic per pass through the algorithm when compared with any other typical projection method, but then introduction of the SIMPLE velocity updates forces additional complete iterations (*i.e.*, passes through the algorithm) per time step in an attempt to recover the divergence-free condition which would have held automatically had not the basic part of SIMPLE been appended. Finally, as has been true with all projection-related algorithms, the computed pressures do not necessarily correspond to physical ones. Unlike basic SIMPLE, the pressure calculated in SIMPLER does bear the error relationship with *Re* found for other projection methods. But just has been true for them, the PPE used to produce this pressure does not satisfy the correct boundary conditions.

3.6 Summary

In this chapter we have presented, in more or less chronological order, many of the main algorithms widely used for solving the incompressible Navier–Stokes equations in primitive-variable form. These fall into two main classes: those that require solution of a pressure Poisson equation (MAC, projection methods, various forms of SIMPLE), and those which do not (SOLA and artificial compressibility methods). We have placed considerable emphasis on projection and SIMPLE methods because these are almost certainly the most widely used. In particular, most research codes requiring time accuracy are based on projection methods, and most commercial CFD codes used in engineering practice have as their core algorithm some form of SIMPLE. In each case we have provided a pseudo-language algorithm—admittedly, rather sketchy in some cases—usually in a coarse-grained form. But use of these in conjunction with detailed discrete equations contained in the notes should be sufficient to permit construction of computer codes for any of the methods treated.

We have restricted the analyses and resulting algorithms to Cartesian coordinates in two space dimensions, and to staggered, structured grids. We have chosen the former because it results in a much simpler presentation, and at the same time, there are no fundamental changes needed for any of the methods treated here when upgrading to three space dimensions and/or generalized coordinates. The choice of staggered grids was made, as discussed in fair detail above, because there is a basic theorem, originating in finite-element analysis, which shows that for typical primitive-variable formulations discretized via finite-difference or finite-volume methods, staggered grids are generally necessary to guarantee convergence of discretely divergence-free solutions to divergence-free continuous ones. As we have previously noted, there are various ways in which this requirement can be circumvented, but all involve more arithmetic and/or result in less accurate solutions than can be easily obtained using staggered grids. It thus seems somewhat unreasonable to employ such so-called "co-located" grids, but this is now widely done in commercial CFD codes. Moreover, it is worth noting that this idea was given serious consideration already as early as the late 1960s, and was found to be deficient. It is mainly the tremendous advances in computing hardware that permit its use today.

Finally, treatment of only structured grids to some extent represents a bias on the part of the author. In recent years most commercial CFD codes have been constructed with unstructured gridding. This carries the advantage of very easy grid generation—often the most time-consuming part of a CFD problem, at least in human time, when a structured grid is employed. But use of unstructured gridding carries with it many disadvantages. It is, in general, more difficult to construct and analyze and less accurate for a given formal accuracy of discretization; it does not allow use of highly-efficient solution techniques widely-employed for structured-grid algorithms; it is less easily parallelized, and it presents often severe difficulties in preparing computed results for visualization. If only one, or maybe two (depending on which ones) of these, were true, it might be more difficult to recommend against unstructured grids; but it seems unreasonable to ignore so many difficulties, especially in the context of research codes where speed, accuracy and ease of parallelization are key requirements.

References

[1] P. J. Roache. *Computational Fluid Dynamics.* Hermosa Publishers, Albuquerque, NM, 1972.

[2] J. Leray. Etude de diverses équations intégrals non linéaires et de quelques problèmes que pose l'hydrodynamique. *J. Math. Pures Appl.* **12**, 1–82, 1933.

[3] J. Leray. Essai sur les mouvemnts d' un liquide visqueaux que limitent des parois. *J. Math. Pures Appl.* **13**, 331–418, 1934.

[4] O. Ladyzhenskaya. *The Mathematical Theory of Viscous Incompressible Flow*, revised English edition (translated from the Russian by Richard A. Silverman). Gordon & Breach, New York, 1963.

[5] D. Ruelle and F. Takens. On the nature of turbulence. *Comm. Math. Phys.* **20**, 167–192, 1971.

[6] E. N. Lorenz. Deterministic nonperiodic flow. *J. Atmos. Sci.* **20**, 130–141, 1963.

[7] S. A. Orszag and G. S. Patterson. Numerical simulation of turbulence: statistical models and turbulence, *Lecture Notes in Physics* **12**, 127–147, Springer-Verlag, Berlin, 1972.

[8] O. Ladyzhenskaya. A dynamical system generated by the Navier–Stokes equations. *J. Soviet Math.* **3**, 458–479, 1973.

[9] O. Ladyzhenskaya. *Attractors for Semigroups and Evolution Equations*, Cambridge University Press, Cambridge, 1991.

[10] R. Temam. *Navier–Stokes Equations: Theory and Numerical Analysis*, North-Holland Pub. Co., Amsterdam, 1979. (new edition published by Amer. Math. Soc., Providence, RI, 2001)

[11] R. Temam. *Navier–Stokes Equations and Nonlinear Functional Analysis*, Soc. Indust. Appl. Math., Philadelphia, 1983. (2^{nd} edition published by SIAM, 1995)

[12] R. Temam. *Infinite Dimensional Dynamical Systems in Mechanics and Physics*, Springer-Verlag, New York, 1988.

[13] P. Constantin and C. Foias. *Navier–Stokes Equations*, University of Chicago Press, Chicago, 1988.

[14] C. R. Doering and J. D. Gibbon. *Applied Analysis of the Navier–Stokes Equations*, Cambridge University Press, Cambridge, 1995.

[15] C. Foias, O. Manley, R. Rosa and R. Temam. *Navier–Stokes Equations and Turbulence*, Cambridge University Press, Cambridge, 2001.

[16] K. E. Gustafson *Introduction to Partial Differential Equations and Hilbert Space Methods*, John Wiley & Sons, New York, 1980.

[17] I. Stakgold. *Boundary Value Problems of Mathematical Physics I, II.* SIAM, Philadelphia, 2000. (Originally published by Macmillan Co., New York, 1967)

[18] P. W. Berg and J. L. McGregor. *Elementary Partial Differential Equations.* Holden-Day, San Francisco, 1966.

[19] L. Schwartz. *Théorie des Distributions I, II.* Hermann, Paris, 1950, 1951.

[20] A. R. Mitchell and D. F. Griffiths. *The Finite Difference Method in Partial Differential Equations*, John Wiley & Sons, Inc., Chichester, 1980.

[21] F. Treves. *Basic Linear Partial Differential Equations*, Academic Press, New York, 1975.

[22] H. L. Royden. *Real Analysis, 2^{nd} ed.* Macmillan, New York, 1971.

[23] J.-P. Aubin. *Applied Functional Analysis*, John Wiley & Sons, New York, 1979.

[24] J. A. Goldstein. *Semigroups of Linear Operators and Applications*, Oxford University Press, New York, 1985.

[25] F. G. Shuman. Numerical method in weather prediction: smoothing and filtering, *Mon. Weath. Rev.* **85**, 357–361, 1957.

[26] W. V. D. Hodge. *The Theory and Application of Harmonic Integrals*, Cambridge University Press: Cambridge, 1952.

[27] P. R. Garabedian. *Partial Differential Equations*, John Wiley & Sons, New York, 1964.

[28] C. Canuto, M. Y. Hussaini, A. Quarteroni and T. A. Tang. *Spectral Methods in Fluid Dynamics*, Springer-Verlag, New York, 1988.

[29] R. D. Richtmyer and K. W. Morton. *Difference Methods for Initial-Value Problems*, John Wiley & Sons, New York, 1967.

[30] P. M. Gresho and R. L. Sani. On pressure boundary conditions for the incompressible Navier–Stokes equations, *Int. J. Numer. Methods Fluids* **7**, 1111–1145, 1987.

[31] C. A. J. Fletcher. *Computational Techniques for Fluid Dynamics, Volume II*, Springer-Verlag, Berlin, 1988.

[32] P. M. Gresho. Incompressible fluid dynamics: some fundamental formulation issues, *Annu. Rev. Fluid Mech.* **23**, 413–453, 1991.

[33] H.-O. Kreiss and J. Lorenz. *Initial-Boundary Value Problems and the Navier–Stokes Equations*, Academic Press, Boston, 1989.

[34] T. B. Gatski, C. E. Grosch and M. E. Rose. A numerical study of the two-dimensional Navier–Stokes equations in vorticity–velocity variables, *J. Comput. Phys.* **48**, 1–22, 1982.

[35] A. N. Ziaei, J. M. McDonough, H. Emdad and A. R. Keshavarzi. Using vorticity to define conditions at multiple open boundaries for simulating flow in a simplified vortex settling basin, *Int. J. Numer. Meth. Fluids* **54**, 1–28, 2007.

[36] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, *Phys. Fluids* **8**, 2182–2189, 1965.

[37] C. W. Hirt, A. A. Amsden and J. L. Cook. An arbitrary Lagrangean-Eulerian computing method for all flow speeds, *J. Comput. Phys.* **14**, 227, 1774.

[38] C. M. Rhie and W. L. Chow. Numerical study of the turbulent flow past an airfoil with trailing edge separation, *AIAA J.* **11**, 1525–1532, 1983.

[39] C. Hirsch. *Numerical Computation of Internal and External Flows, Volume 2: Computational methods for inviscid and viscous flows*, John Wiley & Sons, Chichester, 1990.

[40] S. V. Patankar. *Numerical Heat Transfer and Fluid Flow*, McGraw-Hill Book Co., New York, 1980.

[41] Y. Zang, R. L. Street and J. R. Koseff. A non-staggered grid, fractional step method for time-dependent incompressible Navier–Stokes equations in curvilinear coordinates, *J. Comput. Phys.* **114**, 18–33, 1994.

[42] J. M. McDonough. *Lectures in Basic Computational Numerical Analysis*, available in PDF format at the following URL: http://www.engr.uky.edu/∼egr537

[43] J. Douglas, Jr. and J. E. Gunn. A general formulation of alternating direction methods, part I. parabolic and hyperbolic problems, *Numer. Math.* bf 6 428–453, 1964.

[44] J. M. McDonough. *Lectures in Computational Numerical Analysis of Partial Differential Equations*, available in PDF format at the following URL: http://www.engr.uky.edu/∼acfd

[45] M. D. Gunzburger. *Finite Element Methods for Viscous Incompressible Flows*, Academic Press, Boston, 1989.

[46] J. Kim and P. Moin. Application of a fractional step method to incompressible Navier–Stokes equations, *J. Comput. Phys.* **59**, 308–323, 1985.

[47] D. Tafti. Alternate formulations for the pressure equation Laplacian on a collocated grid for solving the unsteady incompressible Navier–Stokes equations, *J. Comput. Phys.* **116**, 143–153, 1995.

[48] B. P. Leonard. A stable and accurate convection modelling procedure based on quadratic upstream interpolation, *Comput. Meth. Appl. Mech. Engrg.* **19**, 59–98, 1979.

[49] R. E. Mickens. *Difference Equations*, Van Nostrand Rheinhold Co., New York, 1987.

[50] G. de Vahl Davis and G. D. Mallinson. An evaluation of upwind and central difference approximations by a study of recirculating flow, *Computers and Fluids* **4**, 29–43, 1976.

[51] D. B. Spalding. A novel finite-difference formulation for differential expressions involving both first and second derivatives, *Int. J. Num. Methods Eng.* **4**, 551, 1972.

[52] W. Shyy. A study of finite difference approximations to steady-state convection-dominated flow problems, *J. Comput. Phys.* **57**, 415–438, 1985.

[53] T. Yang and J. M. McDonough. Solution filtering technique for solving Burgers' equation, accepted for special issue of *Discrete and Continuous Dynamical Systems*, 2003.

[54] J. M. McDonough and T. Yang. Solution filtering technique for solving turbulence problems, to be submitted to *J. Comput. Phys.*, 2003.

[55] J. M. McDonough, Y. Yang and E. C. Hylin. Modeling Time-dependent turbulent flow over a backward-facing step via additive turbulent decomposition and chaotic algebraic maps, presented at *First Asian Computational Fluid Dynamics Conference*, Hong Kong, Jan. 16–19, 1995.

[56] J. M. McDonough, V. E. Garzon and D. Schulte. Effect of film-cooling hole location on turbulator heat transfer enhancement in turbine blade internal air-cooling circuits, presented at *ASME TURBO EXPO 99*, Indianapolis, IN, June 7–10, 1999.

[57] J. M. McDonough, T. Yang and M. Sheetz. Parallelization of a modern CFD incompressible turbulent flow code, presented at *Parallel CFD 2003*, Moscow, May 13–15, 2003.

[58] M. R. Visbal and D. V. Gaitonde. On the Use of Higher-Order Finite-Difference Schemes on Curvilinear and Deforming Meshes, *J. Comput. Phys.* bf 181, 155–185, 2002.

[59] W. F. Ames. *Numerical Methods for Partial Differential Equations* Second Edition, Academic Press, New York, 1977.

[60] R. Shapiro. Smoothing, filtering and boundary effects, *Rev. Geophys. and Space Phys.* **8**, 359–387, 1970.

[61] A. Majda, J. McDonough, and S. Osher. The Fourier method for nonsmooth initial data, *Math. Comput.* **32**, 1041–1081, 1978.

[62] R. Peyret and T. D. Taylor. *Computational Methods for Fluid Flow*, Springer-Verlag, New York, 1983.

[63] C. W. Hirt, B. D. Nichols and N. C. Romero. SOLA—A Numerical Solution ALgorithm for Transient Fluid Flows, Report LA-5852, Los Alamos Scientific Laboratories, Los Alamos, NM, 1975.

[64] A. J. Chorin. Numerical Study of Thermal Convection in a Fluid Heated from Below, Ph.D. Dissertation, Department of Mathematics, New York University, 1966.

[65] A. J. Chorin. A Numerical Method for Solving Incompressible Viscous Flow Problems, *J. Comput. Phys.* **2**, 12–26, 1967.

[66] D. Kwak, J. L. C. Chang, S. P. Shanks and S. R. Chakravarthy. A Three-Dimensional Incompressible Navier–Stokes Flow Solver Using Primitive Variables, *AIAA J.* **24**, 390–396, 1986.

[67] S. E. Rogers, D. Kwak and J. L. C. Chang. INS3D–An Incompressible Navier–Stokes Code in Generalized Three-Dimensional Coordinates, *NASA Tech. Memo.* 100012, NASA Ames Research Center, 1987.

[68] W. Y. Soh. Time-Marching Solution of Incompressible Navier–Stokes Equations for Internal Flow, *J. Comput. Phys.* **70**, 232–252, 1987.

[69] A. J. Chorin. On Convergence of Discrete Approximations to the Navier–Stokes Equations, *Math. Comput.* **23**, 341–353, 1969.

[70] A. Krzywicki and O. A. Ladyzhenskaya. A grid method for the Navier–Stokes equations, *Soviet Phys. Dokl.* **11**, 212, 1966.

[71] H. Fujita and T. Kato. On the Navier–Stokes initial value problem, *Arch. Rational Mech. Anal.* **16**, 269–316, 1964.

[72] P. M. Gresho. On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. Part 1: Theory, *Int. J. Numer. Meth. Fluids* **11**, 587–620, 1990.

[73] M. Fortin, R. Peyret and R. Temam. Résolution numérique des équations de Navier–Stokes pour un fluide incompressible, *J. de Mécanique* **10**, 357–390, 1971.

[74] J. B. Bell, P. Colella and H. M. Glaz. A Second-Order Projection Method for the Incompressible Navier–Stokes Equations, *J. Comput. Phys.* **85**, 257–283, 1989.

[75] J. Shen. On error estimates of some higher order projection and penalty-projection methods for Navier–Stokes equations, *Numer. Math.* **62**, 49–73, 1992.

[76] J. Shen. A remark on the projection-3 method, *Int. J. Numer. Meth. Fluids* **16**, 249–253, 1993.

[77] V. Girault and P.-A. Raviart. *Finite Element Methods for the Navier–Stokes Equations*, Springer, Berlin, 1986.

[78] N. N. Yanenko. *The Method of Fractional Steps*, Springer-Verlag, Berlin, 1971.

[79] H. B. Keller. *Numerical Methods for Two-Point Boundary-Value Problems.* Dover Pub., Inc., New York, NY, 1992.

[80] C. Fureby and Fernando F. Grinstein. Large Eddy Simulation of High-Reynolds-Number Free and Wall-Bounded Flows, *J. Comput. Phys.* **181**, 68–97, 2002.

[81] A. G. Lamorgese, D. A. Caughey and S. B. Pope. Direct numerical simulation of homogeneous turbulence with hyperviscosity, *Phys. Fluids* **17**, 1–10, 2005.

[82] D. B. Spalding. *GENMIX–A General Computer Program for Two-Dimensional Parabolic Phenomena*, Pergamon, New York, 1977.

[83] G. Comini and S. Del Guidice. Pressure-Velocity Coupling in Incompressible Fluid Flow, *Ann. Rev. Numer. Fluid Mech. and Heat Trans.* **1**, Hemisphere Pub. Corp., Washington, DC, 1987.