P. Wesseling

ELEMENTS OF COMPUTATIONAL FLUID DYNAMICS

Lecture notes WI 4011 Numerieke Stromingsleer

TU Delft

Faculty ITS
Applied Mathematics

Preface

The technological value of computational fluid dynamics has become undisputed. A capability has been established to compute flows that can be investigated experimentally only at reduced Reynolds numbers, or at greater cost, or not at all, such as the flow around a space vehicle at re-entry, or a loss-of-coolant accident in a nuclear reactor. Large commercial computational fluid dynamics computer codes have arisen, and found widespread use in industry. Users of these codes need to be familiar with the basic principles. It has been observed on numerous occasions, that even simple flows are not correctly predicted by advanced computational fluid dynamics codes, if used without sufficient insight in both the numerics and the physics involved. This course aims to elucidate some basic principles of computational fluid dynamics.

Because the subject is vast we have to confine ourselves here to just a few aspects. A more complete introduction is given in Wesseling (2001), and other sources quoted there. Occasionally, we will refer to the literature for further information. But the student will be examined only about material presented in these lecture notes.

Fluid dynamics is governed by partial differential equations. These may be solved numerically by finite difference, finite volume, finite element and spectral methods. In engineering applications, finite difference and finite volume methods are predominant. We will confine ourselves here to finite difference and finite volume methods.

Although most practical flows are turbulent, we restrict ourselves here to laminar flow, because this book is on numerics only. The numerical principles uncovered for the laminar case carry over to the turbulent case. Furthermore, we will discuss only incompressible flow. Considerable attention is given to the convection-diffusion equation, because much can be learned from this simple model about numerical aspects of the Navier-Stokes equations. One chapter is devoted to direct and iterative solution methods.

Errata and MATLAB software related to a number of examples discussed in these course notes may be obtained via the author's website, to be found at `ta.twi.tudelft.nl/nw/users/wesseling`
(see under "Information for students' / "College WI4 011 Numerieke Stromingsleer")

Delft, September 2001 P. Wesseling

Table of Contents

1. The basic equations of fluid dynamics

1.1 Introduction

Fluid dynamics is a classic discipline. The physical principles governing the flow of simple fluids and gases, such as water and air, have been understood since the times of Newton. Since about 1950 classic fluid dynamics finds itself in the company of computational fluid dynamics. This newer discipline still lacks the elegance and unification of its classic counterpart, and is in a state of rapid development.

Good starting points for exploration of the Internet for material related to computational fluid dynamics are the following websites:

`www.cfd-online.com/`

`www.princeton.edu/~gasdyn/fluids.html`

and the ERCOFTAC (European Research Community on Flow, Turbulence and Combustion) site:

`imhefwww.epfl.ch/ERCOFTAC/`

The author's website `ta.twi.tudelft.nl/users/wesseling` also has some links to relevant websites.

Readers well-versed in theoretical fluid dynamics may skip the remainder of this chapter, perhaps after taking note of the notation introduced in the next section. But those less familiar with this discipline will find it useful to continue with the present chapter.

The purpose of this chapter is:

- To introduce some notation that will be useful later;
- To recall some basic facts of vector analysis;
- To introduce the governing equations of laminar incompressible fluid dynamics;
- To explain that the Reynolds number is usually very large. In later chapters this will be seen to have a large impact on numerical methods.

1.2 Vector analysis

Cartesian tensor notation

We assume a right-handed Cartesian coordinate system $(x_1, x_2, ..., x_d)$ with d the number of space dimensions. Bold-faced lower case Latin letters denote vectors, for example, $\mathbf{x} = (x_1, x_2, ..., x_d)$. Greek letters denote scalars. In *Cartesian tensor notation*, which we shall often use, differentiation is denoted as follows:

$$\phi_{,\alpha} = \partial\phi/\partial x_\alpha \ .$$

Greek subscripts refer to coordinate directions, and the *summation convention* is used: summation takes place over Greek indices that occur twice in a term or product.

Examples

Inner product: $\mathbf{u}\cdot\mathbf{v} = u_\alpha v_\alpha = \sum\limits_{\alpha=1}^{d} u_\alpha v_\alpha$

Laplace operator: $\nabla^2\phi = \phi_{,\alpha\alpha} = \sum\limits_{\alpha=1}^{d} \partial^2\phi/\partial x_\alpha^2$

Note that $u_\alpha + v_\alpha$ does not mean $\sum\limits_{\alpha=1}^{d} (u_\alpha + v_\alpha)$ (why?) □

We will also use *vector notation*, instead of the *subscript notation* just explained, and may write div$\mathbf{u}$, if this is more elegant or convenient than the tensor equivalent $u_{\alpha,\alpha}$; and sometimes we write grad ϕ or $\nabla\phi$ for the vector $(\phi_{,1},\ \phi_{,2},\ \phi_{,3})$.

The Kronecker delta $\delta_{\alpha\beta}$ is defined by:

$$\delta_{11} = \delta_{22} = \cdots = \delta_{dd} = 1, \quad \delta_{\alpha\beta} = 0,\ \alpha \neq \beta \ ,$$

where d is the number of space dimensions.

Divergence theorem

We will need the following fundamental theorem:

Theorem 1.2.1. *For any volume $V \subset \mathbb{R}^d$ with piecewise smooth closed surface S and any differentiable scalar field ϕ we have*

$$\int_V \phi_{,\alpha} dV = \int_S \phi n_\alpha dS \ ,$$

where **n** *is the outward unit normal on* S.

For a proof, see for example Aris (1962).

A direct consequence of this theorem is:

Theorem 1.2.2. *(Divergence theorem).*
For any volume $V \subset \mathbb{R}^d$ *with piecewise smooth closed surface* S *and any differentiable vector field* **u** *we have*

$$\int_V \text{div}\mathbf{u} dV = \int_S \mathbf{u} \cdot \mathbf{n} dS \ ,$$

where **n** *is the outward unit normal on* S.

Proof. Apply Theorem 1.2.1 with $\phi_{,\alpha} = u_\alpha, \ \alpha = 1, 2, ..., d$ successively and add. □

A vector field satisfying div$\mathbf{u} = 0$ is called *solenoidal*.

The streamfunction

In two dimensions, if for a given velocity field **u** there exists a function ψ such that

$$\psi_{,1} = -u_2, \quad \psi_{,2} = u_1,$$

then such a function is called the *streamfunction*. For the streamfunction to exist it is obviously necessary that $\psi_{,12} = \psi_{,21}$; therefore we must have $u_{1,1} = -u_{2,2}$, or div $\mathbf{u} = 0$. Hence, *two-dimensional solenoidal vector fields have a streamfunction*. The normal to an isoline $\psi(\mathbf{x}) =$ constant is parallel to $\nabla\psi = (\psi_{,1}, \psi_{,2})$; therefore the vector $\mathbf{u} = (\psi_{,2}, -\psi_{,1})$ is tangential to this isoline. Streamlines are curves that are everywhere tangential to **u**. We see that in two dimensions the streamfunction is constant along streamlines. Later this fact will provide us with a convenient way to compute streamline patterns numerically.

Potential flow

The *curl* of a vector field is defined by

$$\mathrm{curl}\mathbf{u} = \begin{pmatrix} u_{3,2} - u_{2,3} \\ u_{1,3} - u_{3,1} \\ u_{2,1} - u_{1,2} \end{pmatrix} \ .$$

That is, the x_1-component of the vector $\mathrm{curl}\mathbf{u}$ is $u_{3,2} - u_{2,3}$, etc. Often, the curl is called rotation, and a vector field satisfying $\mathrm{curl}\mathbf{u} = 0$ is called *irrotational.* In two dimensions, the curl is obtained by putting the third component and $\partial/\partial x_3$ equal to zero. This gives

$$\mathrm{curl}\mathbf{u} = u_{2,1} - u_{1,2} \ .$$

It can be shown (cf. Aris (1962)) that if a vector field $\mathbf{u}$ satisfies $\mathrm{curl}\mathbf{u} = 0$ there exists a scalar field φ such that

$$\mathbf{u} = \mathrm{grad}\varphi \tag{1.1}$$

(or $u_\alpha = \varphi_{,\alpha}$). The scalar φ is called the *potential*, and flows with velocity field $\mathbf{u}$ satisfying (1.1) are called potential flows or *irrotational* flows (since $\mathrm{curl}\,\mathrm{grad}\varphi = 0$, cf. Exercise 1.2.3).

Exercise 1.2.1. Prove Theorem 1.2.1 for the special case that V is the unit cube.

Exercise 1.2.2. Show that $\mathrm{curl}\mathbf{u}$ is solenoidal.

Exercise 1.2.3. Show that $\mathrm{curl}\,\mathrm{grad}\varphi = 0$.

Exercise 1.2.4. Show that $\delta_{\alpha\alpha} = d$.

1.3 The total derivative and the transport theorem

Streamlines

We repeat: a *streamline* is a curve that is everywhere tangent to the velocity vector $\mathbf{u}(t, \mathbf{x})$ at a given time t. Hence, a streamline may be parametrized with a parameter s such that a streamline is a curve $\mathbf{x} = \mathbf{x}(s)$ defined by

$$d\mathbf{x}/ds = \mathbf{u}(t, \mathbf{x}) \ .$$

The total derivative

Let $\mathbf{x}(t, \mathbf{y})$ be the position of a material particle at time $t > 0$, that at time $t = 0$ had initial position $\mathbf{y}$. Obviously, the velocity field $\mathbf{u}(t, \mathbf{x})$ of the flow satisfies

$$\mathbf{u}(t, \mathbf{x}) = \frac{\partial \mathbf{x}(t, \mathbf{y})}{\partial t} \ . \tag{1.2}$$

The time-derivative of a property ϕ of a material particle, called a *material property* (for example its temperature), is denoted by $D\phi/Dt$. This is called the *total derivative.* All material particles have some ϕ, so ϕ is defined everywhere in the flow, and is a scalar field $\phi(t, \mathbf{x})$. We have

$$\frac{D\phi}{Dt} \equiv \frac{\partial}{\partial t}\phi[t, \mathbf{x}(t, \mathbf{y})] \ , \tag{1.3}$$

where the partial derivative has to be taken with $\mathbf{y}$ constant, since the total derivative tracks variation for a particular material particle. We obtain

$$\frac{D\phi}{Dt} = \frac{\partial \phi}{\partial t} + \frac{\partial x_\alpha(t, \mathbf{y})}{\partial t}\frac{\partial \phi}{\partial x_\alpha} \ .$$

By using (1.2) we get

$$\frac{D\phi}{Dt} = \frac{\partial \phi}{\partial t} + u_\alpha \phi_{,\alpha} \ .$$

The transport theorem

A *material volume* $V(t)$ is a volume of fluid that moves with the flow and consists permanently of the same material particles.

Theorem 1.3.1. *(Reynolds's transport theorem)*
For any material volume $V(t)$ and differentiable scalar field ϕ we have

$$\frac{d}{dt}\int\limits_{V(t)} \phi dV = \int\limits_{V(t)} (\frac{\partial \phi}{\partial t} + \text{div } \phi\mathbf{u}) dV \ . \tag{1.4}$$

For a proof, see Sect. 1.3 of Wesseling (2001).

We are now ready to formulate the governing equations of fluid dynamics, which consist of the conservation laws for mass, momentum and energy.

1.4 Conservation of mass

Continuum hypothesis

The dynamics of fluids is governed by the conservation laws of classical physics, namely conservation of mass, momentum and energy. From these laws partial differential equations are derived and, under appropriate circumstances, simplified. It is customary to formulate the conservation laws under the assumption that the fluid is a continuous medium (*continuum hypothesis*). Physical properties of the flow, such as density and velocity can then be described as time-dependent scalar or vector fields on $\mathbb{R}^2$ or $\mathbb{R}^3$, for example $\rho(t, \mathbf{x})$ and $\mathbf{u}(t, \mathbf{x})$.

The mass conservation equation

The mass conservation law says that the rate of change of mass in an arbitrary material volume $V(t)$ equals the rate of mass production in $V(t)$. This can be expressed as

$$\frac{d}{dt} \int\limits_{V(t)} \rho dV = \int\limits_{V(t)} \sigma dV \ , \tag{1.5}$$

where $\rho(t, \mathbf{x})$ is the density of the material particle at time t and position $\mathbf{x}$, and $\sigma(t, \mathbf{x})$ is the rate of mass production per volume. In practice, $\sigma \neq 0$ only in multiphase flows, in which case (1.5) holds for each phase separately. We take $\sigma = 0$, and use the transport theorem to obtain

$$\int\limits_{V(t)} (\frac{\partial \rho}{\partial t} + \text{div}\rho\mathbf{u}) dV = 0 \ .$$

Since this holds for every $V(t)$ the integrand must be zero:

$$\frac{\partial \rho}{\partial t} + \text{div}\rho\mathbf{u} = 0 \ . \tag{1.6}$$

This is the *mass conservation law*, also called the *continuity equation.*

Incompressible flow

An incompressible flow is a flow in which the density of each material particle remains the same during the motion:

$$\rho[t, \mathbf{x}(t, \mathbf{y})] = \rho(0, \mathbf{y}) \ . \tag{1.7}$$

Hence

$$\frac{D\rho}{Dt} = 0 \ .$$

Because

$$\text{div}\rho\mathbf{u} = \rho \text{div}\mathbf{u} + u_\alpha \rho_{,\alpha} \ ,$$

it follows from the mass conservation law (1.6) that

$$\mathrm{div}\mathbf{u} = 0 \; . \tag{1.8}$$

This is the form that the mass conservation law takes for incompressible flow.

Sometimes incompressibility is erroneously taken to be a property of the fluid rather than of the flow. But it may be shown that compressibility depends only on the speed of the flow, see Sect. 1.12 of Wesseling (2001). If the magnitude of the velocity of the flow is of the order of the speed of sound in the fluid ($\sim$ 340 m/s in air at sea level at 15°C, $\sim$ 1.4 km/s in water at 15°C, depending on the amount of dissolved air) the flow is compressible; if the velocity is much smaller than the speed of sound, incompressibility is a good approximation. In liquids, flow velocities anywhere near the speed of sound cannot normally be reached, due to the enormous pressures involved and the phenomenon of *cavitation*.

1.5 Conservation of momentum

Body forces and surface forces

Newton's law of conservation of momentum implies that the rate of change of momentum of a material volume equals the total force on the volume. There are *body forces* and *surface forces*. A body force acts on a material particle, and is proportional to its mass. Let the volume of the material particle be $dV(t)$ and let its density be ρ. Then we can write

$$\text{body force} = \mathbf{f}^b \rho dV(t) \; . \tag{1.9}$$

A surface force works on the surface of $V(t)$ and is proportional to area. The surface force working on a surface element $dS(t)$ of $V(t)$ can be written as

$$\text{surface force} = \mathbf{f}^s dS(t) \; . \tag{1.10}$$

Conservation of momentum

The law of conservation of momentum applied to a material volume gives

$$\frac{d}{dt}\int\limits_{V(t)} \rho u_\alpha dV = \int\limits_{V(t)} f_\alpha^b dV + \int\limits_{S(t)} f_\alpha^s dS \; . \tag{1.11}$$

By substituting $\phi = \rho u_\alpha$ in the transport theorem (1.4), this can be written as

$$\int\limits_{V(t)} \left[\frac{\partial \rho u_\alpha}{\partial t} + (\rho u_\alpha u_\beta)_{,\beta}\right] dV = \int\limits_{V(t)} \rho f_\alpha^b dV + \int\limits_{S(t)} f_\alpha^s dS \; . \tag{1.12}$$

It may be shown (see Aris (1962)) there exist nine quantities $\tau_{\alpha\beta}$ such that

$$f^s_\alpha = \tau_{\alpha\beta} n_\beta \ , \tag{1.13}$$

where $\tau_{\alpha\beta}$ is the *stress tensor* and $\mathbf{n}$ is the outward unit normal on dS. By applying Theorem 1.2.1 with ϕ replaced by $\tau_{\alpha\beta}$ and n_α by n_β, equation (1.12) can be rewritten as

$$\int\limits_{V(t)} \Big[\frac{\partial \rho u_\alpha}{\partial t} + (\rho u_\alpha u_\beta)_{,\beta}\Big] dV = \int\limits_{V(t)} (\rho f^b_\alpha + \tau_{\alpha\beta,\beta}) dV \ .$$

Since this holds for every $V(t)$, we must have

$$\frac{\partial \rho u_\alpha}{\partial t} + (\rho u_\alpha u_\beta)_{,\beta} = \tau_{\alpha\beta,\beta} + \rho f^b_\alpha \ , \tag{1.14}$$

which is the momentum conservation law . The left-hand side is called the *inertia term*, because it comes from the inertia of the mass of fluid contained in $V(t)$ in equation (1.11).

An example where $\mathbf{f}^b \neq 0$ is stratified flow under the influence of gravity.

Constitutive relation

In order to complete the system of equations it is necessary to relate the rate of strain tensor to the motion of the fluid. Such a relation is called a *constitutive relation.* A full discussion of constitutive relations would lead us too far. The simplest constitutive relation is (see Batchelor (1967))

$$\tau_{\alpha\beta} = -p\delta_{\alpha\beta} + 2\mu(e_{\alpha\beta} - \frac{1}{3}\varDelta\delta_{\alpha\beta}) \ , \tag{1.15}$$

where p is the pressure, $\delta_{\alpha\beta}$ is the Kronecker delta, μ is the dynamic viscosity, $e_{\alpha\beta}$ is the *rate of strain tensor*, defined by

$$e_{\alpha\beta} = \frac{1}{2}(u_{\alpha,\beta} + u_{\beta,\alpha}) \ ,$$

and

$$\varDelta = e_{\alpha\alpha} = \mathrm{div}\mathbf{u} \ .$$

The quantity $\nu = \mu/\rho$ is called the *kinematic viscosity.* In many fluids and gases μ depends on temperature, but not on pressure. Fluids satisfying (1.15) are called *Newtonian fluids.* Examples are gases and liquids such as water and mercury. Examples of non-Newtonian fluids are polymers and blood.

The Navier-Stokes equations

Substitution of (1.15) in (1.14) gives

$$\frac{\partial \rho u_\alpha}{\partial t} + (\rho u_\alpha u_\beta)_{,\beta} = -p_{,\alpha} + 2[\mu(e_{\alpha\beta} - \frac{1}{3}\Delta\delta_{\alpha\beta})]_{,\beta} + \rho f^b_\alpha \ . \tag{1.16}$$

These are the *Navier-Stokes equations.* The terms in the left-hand side are due to the inertia of the fluid particles, and are called the *inertia terms.* The first term on the right represents the pressure force that works on the fluid particles, and is called the *pressure term.* The second term on the right represents the friction force, and is called the *viscous term.* The third term on the right is the *body force.*

Because of the continuity equation (1.6), one may also write

$$\rho\frac{Du_\alpha}{Dt} = -p_{,\alpha} + 2[\mu(e_{\alpha\beta} - \frac{1}{3}\Delta\delta_{\alpha\beta})]_{,\beta} + \rho f^b_\alpha \ . \tag{1.17}$$

In incompressible flows $\Delta = 0$, and we get

$$\rho\frac{Du_\alpha}{Dt} = -p_{,\alpha} + 2(\mu e_{\alpha\beta})_{,\beta} + \rho f^b_\alpha \ . \tag{1.18}$$

These are the *incompressible Navier-Stokes equations.* If, furthermore, $\mu =$ constant then we can use $u_{\beta,\alpha\beta} = u_{\beta,\beta\alpha} = 0$ to obtain

$$\rho\frac{Du_\alpha}{Dt} = -p_{,\alpha} + \mu u_{\alpha,\beta\beta} + \rho f^b_\alpha \ . \tag{1.19}$$

This equation was first derived by Navier (1823), Poisson (1831), de Saint-Venant (1843) and Stokes (1845). Its vector form is

$$\rho\frac{D\mathbf{u}}{Dt} = -\nabla p + \mu\nabla^2\mathbf{u} + \rho\mathbf{f}^b \ ,$$

where ∇^2 is the Laplace operator. The quantity

$$\frac{D\mathbf{u}}{Dt} = \frac{\partial\mathbf{u}}{\partial t} + u_\alpha\mathbf{u}_{,\alpha}$$

is sometimes written as

$$\frac{D\mathbf{u}}{Dt} = \frac{\partial\mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} \ .$$

Making the equations dimensionless

In fluid dynamics there are exactly four independent physical units: those of length, velocity, mass and temperature, to be denoted by L, U, M and

T_r, respectively. From these all other units can be and should be derived in order to avoid the introduction of superfluous coefficients in the equations. For instance, the appropriate unit of time is L/U; the unit of force F follows from Newton's law as MU^2/L. Often it is useful not to choose these units arbitrarily, but to derive them from the problem at hand, and to make the equations dimensionless. This leads to the identification of the dimensionless parameters that govern a flow problem. An example follows.

The Reynolds number

Let L and U be typical length and velocity scales for a given flow problem, and take these as units of length and velocity. The unit of mass is chosen as $M = \rho_r L^3$ with ρ_r a suitable value for the density, for example the density in the flow at upstream infinity, or the density of the fluid at rest. Dimensionless variables are denoted by a prime:

$$\mathbf{x}' = \mathbf{x}/L, \quad \mathbf{u}' = \mathbf{u}/U, \quad \rho' = \rho/\rho_r \ . \tag{1.20}$$

In dimensionless variables, equation (1.16) takes the following form:

$$\frac{L}{U}\frac{\partial \rho' u'_\alpha}{\partial t} + (\rho' u'_\alpha u'_\beta)_{,\beta} = -\frac{1}{\rho_r U^2} p_{,\alpha} + \frac{2}{\rho_r U L}\{\mu(e'_{\alpha\beta} - \frac{1}{3}\Delta' \delta_{\alpha\beta})\}_{,\beta} + \frac{L}{U^2}\rho' f^b_\alpha \ , \tag{1.21}$$

where now the subscript $,\alpha$ stands for $\partial/\partial x'_\alpha$, and $e'_{\alpha\beta} = \frac{1}{2}(u'_{\alpha,\beta} + u'_{\beta,\alpha})$, $\Delta' = e'_{\alpha\alpha}$. We introduce further dimensionless quantities as follows:

$$t' = Ut/L, \quad p' = p/\rho_r U^2, \quad (\mathbf{f}^b)' = \frac{L}{U^2}\mathbf{f}^b. \tag{1.22}$$

By substitution in (1.21) we obtain the following *dimensionless form* of the Navier-Stokes equations, deleting the primes:

$$\frac{\partial \rho u_\alpha}{\partial t} + (\rho u_\alpha u_\beta)_{,\beta} = -p_{,\alpha} + 2\{\mathrm{Re}^{-1}(e_{\alpha\beta} - \frac{1}{3}\Delta\delta_{\alpha\beta})\}_{,\beta} + f^b_\alpha \ ,$$

where the *Reynolds number* Re is defined by

$$\mathrm{Re} = \frac{\rho_r U L}{\mu} \ .$$

The dimensionless form of (1.19) is, if $\rho = \text{constant} = \rho_r$,

$$\frac{Du_\alpha}{Dt} = -p_{,\alpha} + \mathrm{Re}^{-1} u_{\alpha,\beta\beta} + f^b_\alpha \ . \tag{1.23}$$

The transformation (1.20) shows that the inertia term is of order $\rho_r U^2/L$ and the viscous term is of order $\mu U/L^2$. Hence, Re is a measure of the ratio of inertial and viscous forces in the flow. This can also be seen immediately from equation (1.23). For $\mathrm{Re} \gg 1$ inertia dominates, for $\mathrm{Re} \ll 1$ friction (the viscous term) dominates. Both are balanced by the pressure gradient.

In the case of constant density, equations (1.23) and (1.8) form a complete system of four equations with four unknowns. The solution depends on the single dimensionless parameter Re only. What values does Re have in nature? At a temperature of 15^0C and atmospheric pressure, for air we have for the kinematic viscosity $\mu/\rho = 1.5 * 10^{-5}$ m^2/s, whereas for water $\mu/\rho = 1.1 * 10^{-6}$ m^2/s. In the International Civil Aviation Organization Standard Atmosphere, $\mu/\rho = 4.9 * 10^{-5}$ m^2/s at an altitude of 12.5 km. This gives for the flow over an aircraft wing in cruise condition at 12.5 km altitude with wing cord $L = 3$ m and $U = 900$ km/h: $\mathrm{Re} = 1.5 * 10^7$. In a windtunnel experiment at sea-level with $L = 0.5$ m and $U = 25$ m/s we obtain $\mathrm{Re} = 8.3 * 10^5$. For landing aircraft at sea-level with $L = 3$ m and $U = 220$ km/h we obtain $\mathrm{Re} = 1.2 * 10^7$. For a house in a light wind with $L = 10$ m and $U = 0.5$ m/s we have $\mathrm{Re} = 3.3 * 10^5$. Air circulation in a room with $L = 4$ m and $U = 0.1$ m/s gives $\mathrm{Re} = 2.7 * 10^4$. A large ship with $L = 200$ m and $U = 7$ m/s gives $\mathrm{Re} = 1.3 * 10^8$, whereas a yacht with $L = 7$ m and $U = 3$ m/s has $\mathrm{Re} = 1.9 * 10^7$. A small fish with $L = 0.1$ m and $U = 0.2$ m/s has $\mathrm{Re} = 1.8 * 10^4$.

All these very different examples have in common that $\mathrm{Re} \gg 1$, which is indeed almost the rule in flows of industrial and environmental interest. One might think that flows around a given shape will be quite similar for different values of Re, as long as $\mathrm{Re} \gg 1$, but nothing is farther from the truth. At $\mathrm{Re} = 10^7$ a flow may be significantly different from the flow at $\mathrm{Re} = 10^5$, in the same geometry. This strong dependence on Re complicates predictions based on scaled down experiments. Therefore computational fluid dynamics plays an important role in extrapolation to full scale. The rich variety of solutions of (1.23) that evolves as $\mathrm{Re} \to \infty$ is one of the most surprising and interesting features of fluid dynamics, with important consequences for technological applications. A 'route to chaos' develops as $\mathrm{Re} \to \infty$, resulting in *turbulence*. Intricate and intriguing flow patterns occur, accurately rendered in masterful drawings by Leonardo da Vinci, and photographically recorded in Hinze (1975), Nakayama and Woods (1988), Van Dyke (1982) and Hirsch (1988).

Turbulent flows are characterized by small rapid fluctuations of a seemingly random nature. Smooth flows are called *laminar*. The transition form laminar to turbulent flow depends on the Reynolds number and the flow geometry. Very roughly speaking (!), for $\mathrm{Re} > 10000$ flows may be assumed to be turbulent.

The complexity of flows used to be thought surprising, since the physics underlying the governing equations is simply conservation of mass and momentum. Since about 1960, however, it is known that the sweeping generalizations about determinism of Newtonian mechanics made by many scientists (notably Laplace) in the nineteenth century were wrong. Even simple classic nonlinear dynamical systems often exibit a complicated seemingly random behavior, with such a sensitivity to initial conditions, that their long-term behavior cannot be predicted in detail. For a discussion of the modern view on (un-)predictability in Newtonian mechanics, see Lighthill (1986).

The Stokes equations

Very viscous flows are flows with $\mathrm{Re} \ll 1$. For $\mathrm{Re} \downarrow 0$ the system (1.23) simplifies to the *Stokes equations.* If we multiply (1.23) by Re and let $\mathrm{Re} \downarrow 0$, the pressure drops out, which cannot be correct, since we would have four equations (Stokes and mass conservation) for three unknowns u_α. It follows that $p = \mathcal{O}(\mathrm{Re}^{-1})$. We therefore substitute

$$p = \mathrm{Re}^{-1} p^{'} \ . \tag{1.24}$$

From (1.24) and (1.22) it follows that the dimensional (physical) pressure is $\mu U p^{'}/L$. Substitution of (1.24) in (1.23), multiplying by Re and letting $\mathrm{Re} \downarrow 0$ gives the Stokes equations:

$$u_{\alpha,\beta\beta} - p_{,\alpha} = 0 \ . \tag{1.25}$$

These linear equations together with (1.8) were solved by Stokes (1851) for flow around a sphere. Surprisingly, the Stokes equations do *not* describe low Reynolds flow *in two dimensions.* This is called the *Stokes paradox.* See Sect. 1.6 of Wesseling (2001) for the equations that govern low Reynolds flows in two dimensions.

The governing equations of incompressible fluid dynamics are given by, if the density is constant, equations (1.23) and (1.8). This is the only situation to be considerd in these lecture notes.

Exercise 1.5.1. Derive equation (1.17) from equation (1.16).

Exercise 1.5.2. What is the speed of sound and the kinematic viscosity in the air around you? You ride your bike at 18 km/h. Compute your Reynolds number based on a characteristic length (average of body length and width, say) of 1 m. Do you think the flow around you will be laminar or turbulent?

1.6 The convection-diffusion equation

Conservation law for material properties

Let φ be a material property, i.e. a scalar that corresponds to a physical property of material particles, such as heat or concentration of a solute in a fluid, for example salt in water. Assume that φ is conserved and can change only through exchange between material particles or through external sources. Let φ be defined per unit of mass. Then the conservation law for φ is:

$$\frac{d}{dt}\int\limits_{V(t)} \rho\varphi dV = \int\limits_{S(t)} \mathbf{f}\cdot\mathbf{n}\, dS + \int\limits_{V(t)} q dV \ .$$

Here $\mathbf{f}$ is the *flux vector*, governing the rate of transfer through the surface, and q is the source term. For $\mathbf{f}$ we assume *Fick's law* (called Fourier's law if φ is temperature):

$$\mathbf{f} = k \ \mathrm{grad}\ \varphi \ ,$$

with k the diffusion coefficient. By arguments that are now familiar it follows that

$$\frac{\partial\rho\varphi}{\partial t} + \mathrm{div}(\rho\varphi\mathbf{u}) = (k\varphi_{,\alpha})_{,\alpha} + q \ . \tag{1.26}$$

This is the *convection-diffusion equation.* The left-hand side represents transport of φ by convection with the flow, the first term at the right represents transport by diffusion.

By using the mass conservation law, equation (1.26) can be written as

$$\rho\frac{D\varphi}{Dt} = (k\varphi_{,\alpha})_{,\alpha} + q \ . \tag{1.27}$$

If we add a term $r\varphi$ to the left-hand side of (1.26) we obtain the *convection-diffusion-reaction equation*:

$$\frac{\partial\rho\varphi}{\partial t} + \mathrm{div}(\rho\varphi\mathbf{u}) + r\varphi = (k\varphi_{,\alpha})_{,\alpha} + q \ .$$

This equation occurs in flows in which chemical reactions take place. The *Black-Scholes equation*, famous for modeling option prices in mathematical finance, is also a convection-diffusion-reaction equation:

$$\frac{\partial\varphi}{\partial t} + (1-k)\frac{\partial\varphi}{\partial x} + k\varphi = \frac{\partial^2\varphi}{\partial x^2} \,.$$

We will not discuss the convection-diffusion-reaction equation, but only the convection-diffusion equation.

Note that the momentum equation (1.19) comes close to being a convection-diffusion equation. Many aspects of numerical approximation in computational fluid dynamics already show up in the numerical analysis of the relatively simple convection-diffusion equation, which is why we will devote two special chapters to this equation.

Dimensionless form

We can make the convection-diffusion equation dimensionless in the same way as the Navier-Stokes equations. The unit for φ may be called φ_r. It is left as an exercise to derive the following dimensionless form for the convection-diffusion equation (1.26):

$$\frac{\partial \rho\varphi}{\partial t} + \operatorname{div}(\rho\varphi\mathbf{u}) = (\mathrm{Pe}^{-1}\varphi_{,\alpha})_{,\alpha} + q\ , \tag{1.28}$$

where the *Péclet number* Pe is defined as

$$\mathrm{Pe} = \rho_0 UL/k_0\ .$$

We see that the Péclet number characterizes the balance between convection and diffusion. For $\mathrm{Pe} \gg 1$ we have dominating convection, for $\mathrm{Pe} \ll 1$ diffusion dominates. If equation (1.28) stands for the heat transfer equation with φ the temperature, then for air we have $k \approx \mu/0.73$. Therefore, for the same reasons as put forward in Sect. 1.5 for the Reynolds number, in computational fluid dynamics $\mathrm{Pe} \gg 1$ is the rule rather than the exception.

Exercise 1.6.1. Derive equation (1.28).

1.7 Summary of this chapter

We have introduced Cartesian tensor notation, and have recalled some basic facts from vector analysis. The transport theorem helps to express the conservation laws for mass and momentum of a fluid particle in terms of partial differential equations. This leads to the incompressible Navier-Stokes equations. Nondimensionalization leads to the identification of the dimensionless parameter governing incompressible viscous flows, called the Reynolds number. We have seen that the value of the Reynolds number is usually very high in flows of industrial and environmental interest. We have briefly touched upon the phenomenon of turbulence, which occurs if the Reynolds number is large enough. The convection-diffusion equation, which is the conservation law for material properties that are transported by convection and diffusion, has been derived. Its dimensionless form gives rise to the dimensionless Péclet number.

Some self-test questions

Write down the divergence theorem.

What is the total derivative?

Write down the transport theorem.

Write down the governing equations of incompressible viscous flow.

Define the Reynolds number.

Write down the convection-diffusion-reaction equation.

2. The stationary convection-diffusion equation in one dimension

2.1 Introduction

Although the one-dimensional case is of no practical use, we will devote a special chapter to it, because important general principles of CFD can be easily analyzed and explained thoroughly in one dimension. We will pay special attention to difficulties caused by a large Péclet number Pe, which is generally the case in CFD, as noted in Sect. 1.6.

In this chapter we consider the one-dimensional stationary version of the dimensionless convection-diffusion equation (1.28) with $\rho = 1$:

$$\frac{du\varphi}{dx} = \frac{d}{dx}\left(\varepsilon\frac{d\varphi}{dx}\right) + q(x)\ , \quad x \in \Omega \equiv (0,1)\ , \tag{2.1}$$

where the domain has been chosen to be the unit interval, and $\varepsilon = 1/\text{Pe}$. For the physical meaning of this equation, see Sect. 1.6

The purpose of this chapter is:

• To explain that a boundary value problem can be *well-posed* or *ill-posed*, and to identify boundary conditions that give a well-posed problem for $\text{Pe} \gg 1$;

• To discuss the choice of outflow boundary conditions;

• To explain how the *maximum principle* can tell us whether the exact solution is *monotone*;

• To explain the *finite volume* discretization method;

• To explain the *discrete maximum principle* that may be satisfied by the numerical scheme;

• To study the *local truncation error* on nonuniform grids;

• To show by means of the discrete maximum principle that although the local truncation error is relatively large at the boundaries and in the interior of a nonuniform grid, nevertheless the *global truncation error* can be about as small as on a uniform grid;

• To show how by means of local grid refinement accuracy and computing work can be made independent of the Péclet number;

• To illustrate the above points by numerical experiments;

• To give a few hints about programming in MATLAB.

2.2 Analytic aspects

Conservation form

The time-dependent version of (2.1) can be written as

$$\frac{\partial \varphi}{\partial t} = L\varphi + q\,, \quad L\varphi \equiv \frac{\partial u\varphi}{\partial x} - \frac{\partial}{\partial x}\left(\varepsilon \frac{\partial \varphi}{\partial x}\right).$$

Let us integrate over Ω:

$$\frac{d}{dt}\int_\Omega \varphi d\Omega = \int_\Omega L\varphi d\Omega + \int_\Omega q d\Omega.$$

Since

$$\int_\Omega L\varphi d\Omega = \left(u\varphi - \varepsilon \frac{\partial \varphi}{\partial x}\right)\Big|_0^1, \tag{2.2}$$

(where we define $f(x)|_a^b \equiv f(b) - f(a)$), we see that

$$\frac{d}{dt}\int_\Omega \varphi d\Omega = \left(u\varphi - \varepsilon \frac{\partial \varphi}{\partial x}\right)\Big|_0^1 + \int_\Omega q d\Omega.$$

Hence, if there is no transport through the boundaries $x = 0, 1$, and if the source term $q = 0$, then

$$\frac{d}{dt}\int_\Omega \varphi d\Omega = 0\,.$$

Therefore $\int_\Omega \varphi d\Omega$ is *conserved*. The total amount of φ, *i.e.* $\int_\Omega \varphi d\Omega$, can change only in time by transport through the boundaries $x = 0, 1$, and by the action of a source term q. Therefore a differential operator such as L, whose integral over the domain Ω reduces to an integral over the boundary, is said to be in *conservation form*.

A famous example of a *nonlinear* convection equation (no diffusion) is the *Burgers equation* (named after the TUD professor J.M. Burgers, 1895–1981):

$$\frac{\partial \varphi}{\partial t} + \frac{1}{2}\frac{\partial \varphi^2}{\partial x} = 0\,. \tag{2.3}$$

This equation is in conservation form. But the following version is not in conservation form:

$$\frac{\partial \varphi}{\partial t} + \varphi\frac{\partial \varphi}{\partial x} = 0\,. \tag{2.4}$$

An exact solution

Let $u \equiv 1$, $\varepsilon =$ constant and $q = 0$. Then equation (2.1) becomes

$$\frac{d\varphi}{dx} = \varepsilon\frac{d^2\varphi}{dx^2}\ , \quad x \in \varOmega \equiv (0,1)\ , \tag{2.5}$$

which can be solved analytically by postulating $\varphi = \mathrm{e}^{\lambda x}$. Substitution in (2.5) shows this is a solution if

$$\lambda - \varepsilon\lambda^2 = 0\ ,$$

hence $\lambda = 0$ or $\lambda = 1/\varepsilon$. Therefore the general solution is

$$\varphi(x) = A + B\mathrm{e}^{x/\varepsilon}\ , \tag{2.6}$$

with A and B free constants, that must follow from the boundary conditions, in order to determine a unique solution. We see that precisely two boundary conditions are needed.

Boundary conditions

For a second order differential equation, such as (2.1), two *boundary conditions* are required, to make the solution unique. A differential equation together with its boundary conditions is called a *boundary value problem.* We start with the following two boundary conditions, both at $x = 0$:

$$\varphi(0) = a, \quad \frac{d\varphi(0)}{dx} = b. \tag{2.7}$$

The first condition, which prescribes a value for φ, is called a *Dirichlet condition*; the second, which prescribes a value for the derivative of φ, is called a *Neumann condition.* The boundary conditions (2.7) are satisfied if the constants in (2.6) are given by $A = a - \varepsilon b, \quad B = \varepsilon b$, so that the exact solution is given by

$$\varphi(x) = a - \varepsilon b + \varepsilon b\mathrm{e}^{x/\varepsilon}\ . \tag{2.8}$$

Ill-posed and well-posed

We now show *there is something wrong* with boundary conditions (2.7) if $\varepsilon \ll 1$. Suppose b is perturbed by an amount δb. The resulting perturbation in $\varphi(1)$ is

$$\delta\varphi(1) = \varepsilon \delta b \mathrm{e}^{1/\varepsilon} .$$

We see that

$$\frac{|\delta\varphi(1)|}{|\delta b|} \gg 1 \quad \text{if} \quad \varepsilon \ll 1 .$$

Hence, a small change in a boundary condition causes a large change in the solution if $\varepsilon \ll 1$. We assume indeed $\varepsilon \ll 1$, for reasons set forth in Sect. 1.6. Problems which have large sensitivity to perturbations of the boundary data (or other input, such as coefficients and right-hand side) are called *ill-posed*. Usually, but not always, ill-posedness of a problem indicates a fault in the formulation of the mathematical model. The opposite of ill-posed is *well-posed*. Since numerical approximations always involve perturbations, ill-posed problems can in general not be solved numerically with satisfactory accuracy, especially in more than one dimension (although there are special numerical methods for solving ill-posed problems with reasonable accuracy; this is a special field). In the present case ill-posedness is caused by wrong boundary conditions.

It is left to the reader to show in Exercise 2.2.2 that the following boundary conditions:

$$\varphi(0) = a, \quad \frac{d\varphi(1)}{dx} = b. \tag{2.9}$$

lead to a well-posed problem. The exact solution is now given by (verify this):

$$\varphi(x) = a + \varepsilon b(\mathrm{e}^{(x-1)/\varepsilon} - \mathrm{e}^{-1/\varepsilon}). \tag{2.10}$$

Note that equation (2.5) corresponds to a velocity $u = 1$, so that $x = 0$ is an inflow boundary. Hence in (2.9) we have a Dirichlet boundary condition at the inflow boundary and a Neumann boundary condition at the outflow boundary. If we assume $u = -1$, so that this is the other way around, then the problem is ill-posed as $\varepsilon \ll 1$ with boundary conditions (2.9). This follows from the result of Exercise 2.2.3. We conclude that it is wrong to give a Neumann condition at an inflow boundary.

Finally, let a Dirichlet condition is given at both boundaries:

$$\varphi(0) = a, \quad \varphi(1) = b. \tag{2.11}$$

The exact solution is

$$\varphi(x) = a + (b-a)\frac{\mathrm{e}^{x/\varepsilon} - 1}{\mathrm{e}^{1/\varepsilon} - 1} \tag{2.12}$$

The result of Exercise 2.2.4 shows that boundary conditions (2.11) give a well-posed problem.

To summarize: *To obtain a well-posed problem, the boundary conditions must be correct.*

Maximum principle

We rewrite equation (2.1) as

$$\frac{du\varphi}{dx} - \frac{d}{dx}\left(\varepsilon\frac{d\varphi}{dx}\right) = q(x)\ , \quad x \in \Omega\,. \tag{2.13}$$

Let $q(x) < 0$, $\forall x \in \Omega$. In an interior extremum of the solution in a point x_0 we have $d\varphi(x_0)/dx = 0$, so that

$$\varphi(x_0)\frac{du(x_0)}{dx} - \varepsilon\frac{d^2\varphi(x_0)}{dx^2} < 0.$$

Now suppose that $du/dx = 0$ (in more dimensions it suffices that div$\boldsymbol{u} = 0$, which is satisfied in incompressible flows). Then $d^2\varphi(x_0)/dx^2 > 0$, so that the extremum cannot be a maximum. This result is strengthened to the case $q(x) \leq 0$, $\forall x \in \Omega$, i.e. including the equality sign, in Theorem 2.4.1 of Wesseling (2001). Hence, if

$$u\frac{d\varphi}{dx} - \varepsilon\frac{d^2\varphi}{dx^2} < 0, \quad \forall x \in \Omega\,,$$

local maxima can occur only at the boundaries. This is called the *maximum principle.* By reversing signs we see that if $q(x) \leq 0$, $\forall x \in \Omega$ there cannot be an interior minimum. If $q(x) \equiv 0$ there cannot be an interior extremum, so that the solution is *monotone* (in one dimension). The maximum principle gives us important information about the solution, without having to determine the solution. Such information is called a priori information.

If the exact solution has no local maximum or minimum, then "*wiggles*" (oscillations) in a numerical solution are nor physical, but must be a numerical artifact.

This concludes our discussion of analytic aspects (especially for $\varepsilon \ll 1$) of the convection-diffusion equation. We now turn to numerical solution methods.

Exercise 2.2.1. Show that equation (2.3) is in conservation form, and that (2.4) is not.

Exercise 2.2.2. Show that the solution (2.10) satisfies

$$\frac{|\delta\varphi(x)|}{|\delta a|} = 1, \quad \frac{|\delta\varphi(x)|}{|\delta b|} < \varepsilon(1 + \mathrm{e}^{-1/\varepsilon}).$$

Hence, the solution is relatively insensitive to the boundary data a and b for all $\varepsilon > 0$.

Exercise 2.2.3. Show that with $u = -1$, $\varepsilon =$ constant and $q = 0$ the solution of equation (2.1) is given by

$$\varphi(x) = a + b\varepsilon\mathrm{e}^{1/\varepsilon}(1 - \mathrm{e}^{-x/\varepsilon}) \ ,$$

so that

$$\frac{\delta\varphi(1)}{\delta b} = \varepsilon(\mathrm{e}^{1/\varepsilon} - 1) \ .$$

Why does this mean that the problem is ill-posed for $\varepsilon \ll 1$?

Exercise 2.2.4. Show that it follows from the exact solution (2.12) that

$$\frac{|\delta\varphi(x)|}{|\delta a|} < 2 \ , \quad \frac{|\delta\varphi(x)|}{|\delta b|} < 1 \ .$$

□

2.3 Finite volume method

We now describe how equation (2.1) is discretized with the finite volume method. We rewrite (2.1) as

$$L\varphi \equiv \frac{du\varphi}{dx} - \frac{d}{dx}\left(\varepsilon\frac{d\varphi}{dx}\right) = q, \quad x \in \Omega \equiv (0,1) \ . \tag{2.14}$$

Let $x = 0$ be an inflow boundary, *i.e.* $u(0) > 0$. *As seen before, it would be wrong to prescribe a Neumann condition (if $\varepsilon \ll 1$), so we assume a Dirichlet condition:*

$$\varphi(0) = a \ . \tag{2.15}$$

Let $x = 1$ be an outflow boundary, i.e. $u(1) > 0$. We prescribe either a Neumann condition:

$$d\varphi(1)/dx = b \ , \tag{2.16}$$

or a Dirichlet condition:

$$\varphi(1) = b \ . \tag{2.17}$$

The finite volume method works as follows. The domain Ω is subdivided in segments $\Omega_j, \ j = 1, \cdots, J$, as shown in the upper part of Fig. 2.1. The segments are called cells or finite volumes or control volumes , and the segment

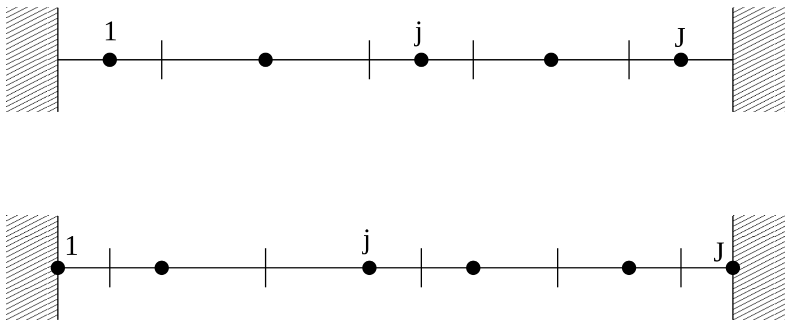

Fig. 2.1. Non-uniform cell-centered grid (above) and vertex-centered grid (below).

length, denoted by h_j, is called the mesh size. The coordinates of the centers of the cells are called x_j, the size of Ω_j is called h_j and the coordinate of the interface between Ω_j and Ω_{j+1} is called $x_{j+1/2}$. The cell centers are frequently called grid points or nodes. This is called a *cell-centered* grid; the nodes are in the centers of the cells and there are no nodes on the boundaries. In a *vertex-centered* grid one first distributes the nodes over the domain and puts nodes on the boundary; the boundaries of the control volumes are centered between the nodes; see the lower part of Fig. 2.1. We continue with a cell-centered grid. We integrate equation (2.14) over Ω_j and obtain:

$$\int\limits_{\Omega_j} L\varphi d\Omega = F|_{j-1/2}^{j+1/2} = \int\limits_{\Omega_j} q d\Omega \cong h_j q_j \ ,$$

with $F|_{j-1/2}^{j+1/2} \equiv F_{j+1/2} - F_{j-1/2},\ F_{j+1/2} = F(x_{j+1/2}),\ F(x) \equiv u\varphi - \varepsilon d\varphi/dx$. Often, $F(x)$ is called the *flux*. The following scheme is obtained:

$$L_h\varphi_j \equiv F_{j+1/2} - F_{j-1/2} = h_j q_j \ , \quad j = 1, \cdots, J \ . \tag{2.18}$$

We will call $u\varphi$ the convective flux and $\varepsilon d\varphi/dx$ the diffusive flux.

Conservative scheme

Summation of equation (2.18) over all cells gives

$$\sum_{j=1}^{J} L_h\varphi_j = F_{J+1/2} - F_{1/2} \ . \tag{2.19}$$

We see that only boundary fluxes remain, to that equation (2.19) mimics the conservation property (2.2) of the differential equation. Therefore the scheme (2.18) is called *conservative*. This property is generally beneficial for accuracy and physical realism.

Discretization of the flux

To complete the discretization, the flux $F_{j+1/2}$ has to be approximated in terms of neighboring grid function values; the result is called the *numerical flux*. *Central discretization* of the convection term is done by approximating the convective flux as follows:

$$(u\varphi)_{j+1/2} \cong u_{j+1/2}\varphi_{j+1/2}, \quad \varphi_{j+1/2} = \frac{1}{2}(\varphi_j + \varphi_{j+1})\ . \tag{2.20}$$

Since $u(x)$ is a known coefficient, $u_{j+1/2}$ is known. One might think that better accuracy on nonuniform grids is obtained by linear interpolation:

$$\varphi_{j+1/2} = \frac{h_j\varphi_{j+1} + h_{j+1}\varphi_j}{h_j + h_{j+1}}\ . \tag{2.21}$$

Surprisingly, the scheme (2.18) is *less accurate* with (2.21) than with (2.20),

as we will see. This is one of the important lessons that can be learned from the present simple one-dimensional example.

Upwind discretization is given by

$$(u\varphi)_{j+1/2} \cong \frac{1}{2}(u_{j+1/2} + |u_{j+1/2}|)\varphi_j + \frac{1}{2}(u_{j+1/2} - |u_{j+1/2}|)\varphi_{j+1}\ . \tag{2.22}$$

This means that $u\varphi$ is biased in upstream direction (to the left for $u > 0$, to the right for $u < 0$), which is why this is called upwind discretization.

The diffusive part of the flux is approximated by

$$(\varepsilon\frac{d\varphi}{dx})_{j+1/2} \cong \varepsilon_{j+1/2}(\varphi_{j+1} - \varphi_j)/h_{j+1/2}, \quad h_{j+1/2} = \frac{1}{2}(h_j + h_{j+1})\ . \tag{2.23}$$

Boundary conditions

At $x = 0$ we cannot approximate the diffusive flux by (2.23), since the node x_0 is missing. We use the Dirichlet boundary condition, and write:

$$(\varepsilon\frac{d\varphi}{dx})_{1/2} \cong 2\varepsilon_{1/2}(\varphi_1 - a)/h_1\ . \tag{2.24}$$

This is a one-sided approximation of $(\varepsilon\frac{d\varphi}{dx})_{1/2}$, which might impair the accuracy of the scheme. We will investigate later whether this is the case or not. The convective flux becomes simply

$$(u\varphi)_{1/2} \cong u_{1/2}a\ . \tag{2.25}$$

Next, consider the boundary $x = 1$. Assume we have the Neumann condition (2.16). The diffusive flux is given directly by the Neumann condition:

$$(\varepsilon \frac{d\varphi}{dx})_{J+1/2} \cong \varepsilon_{J+1/2} b. \tag{2.26}$$

Since $x = 1$ is assumed to be an outflow boundary, we have $u_{J+1/2} > 0$, so that for the upwind convective flux (2.22) an approximation for $\varphi_{J+1/2}$ is not required. For the central convective fluxes (2.20) or (2.21) we approximate $\varphi_{J+1/2}$ with extrapolation, using the Neumann condition:

$$\varphi_{J+1/2} \cong \varphi_J + h_J b/2 \ . \tag{2.27}$$

The Dirichlet condition (2.17) is handled in the same way as at $x = 0$.

The numerical scheme

The numerical flux as specified above can be written as

$$\begin{aligned} F_{j+1/2} &= \beta_j^0 \varphi_j + \beta_{j+1}^1 \varphi_{j+1}, \quad j = 1, \cdots, J-1 \ , \\ F_{1/2} &= \beta_1^1 \varphi_1 + \gamma_0 \ , \quad F_{J+1/2} = \beta_J^0 \varphi_J + \gamma_1 \ , \end{aligned} \tag{2.28}$$

where $\gamma_{0,1}$ are known terms arising from the boundary conditions. For eample, for the upwind scheme we obtain the results specified in Exercise 2.3.2.

For future reference, we also give the coefficients for the central schemes. For the central scheme (2.20) we find:

$$\begin{aligned} \beta_j^0 &= \frac{1}{2} u_{j+1/2} + (\varepsilon/h)_{j+1/2}, \quad j = 1, \cdots, J-1 \ , \\ \beta_{j+1}^1 &= \frac{1}{2} u_{j+1/2} - (\varepsilon/h)_{j+1/2}, \quad j = 1, \cdots, J-1 \ , \quad \beta_1^1 = -2\varepsilon_{1/2}/h_1 \ , \\ \gamma_0 &= (u_{1/2} + 2\varepsilon_{1/2}/h_1) a \ , \\ \beta_J^0 &= u_{J+1/2} \ , \quad \gamma_1 = u_{J+1/2} h_J b/2 - \varepsilon_{J+1/2} b \quad \text{(Neumann)}, \\ \beta_J^0 &= 2\varepsilon_{J+1/2}/h_J \ , \quad \gamma_1 = (u_{J+1/2} - 2\varepsilon_{J+1/2}/h_J) b \quad \text{(Dirichlet)}. \end{aligned} \tag{2.29}$$

For the central scheme (2.21) we find:

$$\begin{aligned} \beta_j^0 &= \frac{h_{j+1}}{2h_{j+1/2}} u_{j+1/2} + (\varepsilon/h)_{j+1/2}, \quad j = 1, \cdots, J-1 \ , \\ \beta_{j+1}^1 &= \frac{h_j}{2h_{j+1/2}} u_{j+1/2} - (\varepsilon/h)_{j+1/2}, \quad j = 1, \cdots, J-1 \ . \end{aligned} \tag{2.30}$$

The other coefficients (at the boundaries) are the same as in equation (2.29). On a uniform grid the central schemes (2.29) and (2.30) are identical.

Substitution of equations (2.66)–(2.30) in equation (2.18) gives the following linear algebraic system:

$$L_h\varphi_j = \alpha_j^{-1}\varphi_{j-1} + \alpha_j^0\varphi_j + \alpha_j^1\varphi_{j+1} = \tilde{q}_j, \quad j = 1,\cdots,J\ , \tag{2.31}$$

with $\alpha_1^{-1} = \alpha_J^1 = 0$. This is called the numerical scheme or the finite volume scheme. Its coefficients are related to those of the numerical flux (2.66) –(2.30) by

$$\begin{aligned} \alpha_j^{-1} &= -\beta_{j-1}^0\ , \quad j = 2,\cdots,J\ , \\ \alpha_j^0 &= \beta_j^0 - \beta_j^1\ , \quad j = 1,\cdots,J\ , \\ \alpha_j^1 &= \beta_{j+1}^1\ , \quad j = 1,\cdots,J-1\ . \end{aligned} \tag{2.32}$$

The right-hand side is found to be

$$\begin{aligned} \tilde{q}_j &= h_j q_j\ , \quad j = 2,\cdots,J-1\ , \\ \tilde{q}_1 &= h_1 q_1 + \gamma_0\ , \quad \tilde{q}_J = h_J q_J - \gamma_1\ . \end{aligned} \tag{2.33}$$

Stencil notation

The general form of a linear scheme is

$$L_h\varphi_j = \sum_{k\in K} \alpha_j^k \varphi_{j+k} = \tilde{q}_j\ , \tag{2.34}$$

with K some index set. For example, in the case of (2.34), $K = \{-1,\ 0,\ 1\}$. The stencil $[L_h]$ of the operator L_h is a tableau of the coefficients of the scheme of the following form:

$$[L_h]_j = [\ \alpha_j^{-1} \quad \alpha_j^0 \quad \alpha_j^1\ ]\ . \tag{2.35}$$

We will see later that this is often a convenient way to specify the coefficients. Equation (2.34) is the stencil notation of the scheme.

The matrix of the scheme

In matrix notation the scheme can be denoted as

$$Ay = b\ , \quad y = \begin{bmatrix} \varphi_1 \\ \vdots \\ \varphi_J \end{bmatrix}, \quad b = \begin{bmatrix} \tilde{q}_1 \\ \vdots \\ \tilde{q}_J \end{bmatrix}, \tag{2.36}$$

where A is the following tridiagonal matrix:

$$A = \begin{bmatrix} \alpha_1^0 & \alpha_1^1 & 0 & \cdots & 0 \\ \alpha_2^{-1} & \alpha_2^0 & \alpha_2^1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \alpha_{J-1}^{-1} & \alpha_{J-1}^0 & \alpha_{J-1}^1 \\ 0 & \cdots & 0 & \alpha_J^{-1} & \alpha_J^0 \end{bmatrix} . \qquad (2.37)$$

In MATLAB, A is simply constructed as a sparse matrix by (taking note of equation (2.32)):

```
A = spdiags([-beta0 beta0-beta1 beta1], -1:1, n, n);
```

with suitable definition of the algebraic vectors `beta0` and `beta1`. This is used in the MATLAB code `cd1` and several other of the MATLAB codes that go with this course. These programs are available at the author's website; see the Preface to these lecture notes.

Vertex-centered grid

In the interior of a vertex-centered grid the finite volume method works just as in the cell-centered case, so that further explanation is not necessary. But at the boundaries the procedure is a little different. If we have a Dirichlet condition, for example at $x = 0$, then an equation for φ_1 is not needed, because φ_1 is prescribed (x_1 is at the boundary, see Fig. 2.1). Suppose we have a Neumann condition at $x = 1$. Finite volume integration over the last control volume (which has x_J as the right end point, see Fig. 2.1) gives:

$$L_h\varphi_J \equiv F_J - F_{J-1/2} = h_J q_J \ ,$$

where we approximate F_J as follows, in the case of the central scheme for convection, for example:

$$F_J = u_J\varphi_J - \varepsilon_J b \ ,$$

where b is given in (2.16).

Symmetry

When the velocity $u \equiv 0$, the convection-diffusion equation reduces to the diffusion equation, also called heat equation. According to equations (2.66)—(2.30) we have in this case $\beta_j^1 = -\beta_{j-1}^0$, so that (2.32) gives $\alpha_j^1 = \alpha_{j+1}^{-1}$, which makes the matrix A *symmetric.* This holds also in the vertex-centered case. Symmetry can be exploited to save computer memory and to make solution methods more efficient. Sometimes the equations are scaled to make the coefficients of size $\mathcal{O}(1)$; but this destroys symmetry, unless the same scaling factor is used for every equation.

Two important questions

The two big questions asked in the numerical analysis of differential equations are:

• How well does the numerical solution approximate the exact solution of equation (2.14)?

• How accurately and efficiently can we solve the linear algebraic system (2.36)?

These questions will come up frequently in what follows. In the ideal case one shows theoretically that the numerical solution converges to the exact solution as the mesh size $h_j \downarrow 0$. In the present simple case, where we have the exact solution (2.6), we can check convergence by numerical experiment.

Numerical experiments on uniform grid

We take $u = 1$, ε constant, $q = 0$ and the grid cell-centered and uniform, with $h_j = h = 1/12$. We choose Dirichlet boundary conditions (2.11) with $a = 0.2$, $b = 1$. The exact solution is given by (2.12). The numerical results in this section have been obtained with the MATLAB code `cd1` . Fig. 2.2 gives results for two values of the Péclet number (remember that $\varepsilon = 1/\text{Pe}$). We see

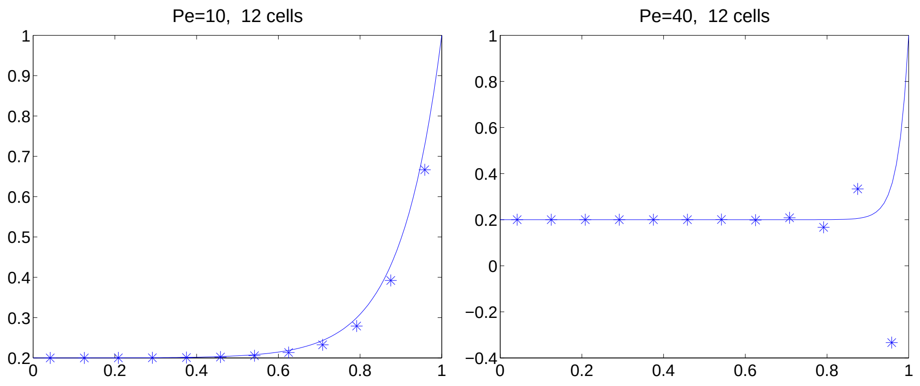

Fig. 2.2. Exact solution (—) and numerical solution (*).

a marked difference between the cases Pe = 10 and Pe = 40. The numerical solution for Pe = 40 is completely unacceptable. We will now analyze why this is so and look for remedies.

The maximum principle

In Sect. 2.2 we saw that according to the maximum principle, with $q = 0$ the solution of equation (2.14) cannot have local extrema in the interior. This is confirmed of course in Fig. 2.2. However, the numerical solution for $\mathrm{Pe} = 40$ shows local extrema. These undesirable numerical artifacts are often called "wiggles". It is desirable that the numerical scheme satisfies a similar maximum principle as the differential equation, so that artificial wiggles are excluded. This is the case for *positive schemes*, defined below. Let the scheme be written in stencil notation:

$$L_h\varphi_j = \sum_{k\in K} \alpha_j^k \varphi_{j+k} = \tilde{q}_j \ , \quad j = 1, \cdots, J \ .$$

Definition 2.3.1. The operator L_h is *of positive type* if

$$\sum_{k\in K} \alpha_j^k = 0, \quad j = 2, \cdots, J-1 \tag{2.38}$$

and

$$\alpha_j^k < 0, \quad k \neq 0, \quad j = 2, \cdots, J-1 \ . \tag{2.39}$$

Note that a condition is put on the coefficients only in the interior. The following theorem says that schemes of positive type satisfy a similar maximum principle as the differential equation.

Theorem 2.3.1. *Discrete maximum principle.*
If L_h is of positive type and

$$L_h\varphi_j \leq 0, \quad j = 2, \cdots, J-2 \ ,$$

then $\varphi_j \leq \max\{\varphi_1, \varphi_J\}$.

Corollary Let conditions (2.38) and (2.39) also hold for $j = J$. Then $\varphi_j \leq \varphi_1$.

A formal proof is given in Sect. 4.4 of Wesseling (2001), but it is easy to see that the theorem is true. Let $K = \{-1,\ 0,\ 1\}$. We have for every interior grid point x_j:

$$\varphi_j \leq w_{-1}\varphi_{j-1} + w_1\varphi_{j+1} \ , \quad w_{\pm 1} \equiv -\alpha_j^{\pm 1}/\alpha_j^0 \ .$$

Since $w_1 + w_1 = 1$ and $w_{\pm 1} > 0$, φ_j is a weighted average of its neighbors φ_{j-1} and φ_{j+1}. Hence, either $\varphi_j < \max\{\varphi_{j-1}, \varphi_{j+1}\}$ or $\varphi_j = \varphi_{j-1} = \varphi_{j+1}$.

Let us now see whether the scheme used for Fig. 2.2 is of positive type. Its stencil is given by equation (2.67):

$$[L_h] = \left[\begin{array}{ccc} -\frac{1}{2}u - \frac{\varepsilon}{h} & 2\frac{\varepsilon}{h} & \frac{1}{2}u - \frac{\varepsilon}{h} \end{array} \right] . \tag{2.40}$$

We see that this scheme is of positive type if and only if

$$p < 2 , \quad p \equiv \frac{|u|h}{\varepsilon} . \tag{2.41}$$

The dimensionless number p is called the *mesh Péclet number*.

For the left half of Fig. 2.2 we have $p = 10/12 < 2$, whereas for the right half $p = 40/12 > 2$, which explains the wiggles. In general, $\mathrm{Pe} = UL/\varepsilon$ and $p = Uh/\varepsilon$, so that $p = \mathrm{Pe}h/L$, with L the length of the domain Ω and U representative of the size of u, for example $U = \max[u(x) : x \in \Omega]$. and for $p < 2$ we must choose h small enough: $h/L < 2/\mathrm{Pe}$. Since in practice Pe is usually very large, as shown in Sect. 1.6, this is not feasible (certainly not in more than one dimension), due to computer time and memory limitations. Therefore a scheme is required that is of positive type for all values of Pe. Such a scheme is obtained if we approximate the convective flux $u\varphi$ such that a non-positive contribution is made to $\alpha_j^{\pm 1}$. This is precisely what the upwind scheme (2.22) is about. For the problem computed in Fig. 2.2 its stencil is, since $u > 0$:

$$[L_h] = \left[\begin{array}{ccc} -u - \frac{\varepsilon}{h} & u + 2\frac{\varepsilon}{h} & -\frac{\varepsilon}{h} \end{array} \right] . \tag{2.42}$$

It is easy to see that this scheme is of positive type for all Pe. Results are given in Fig. 2.3. We see that wiggles are absent, and that the numerical solution

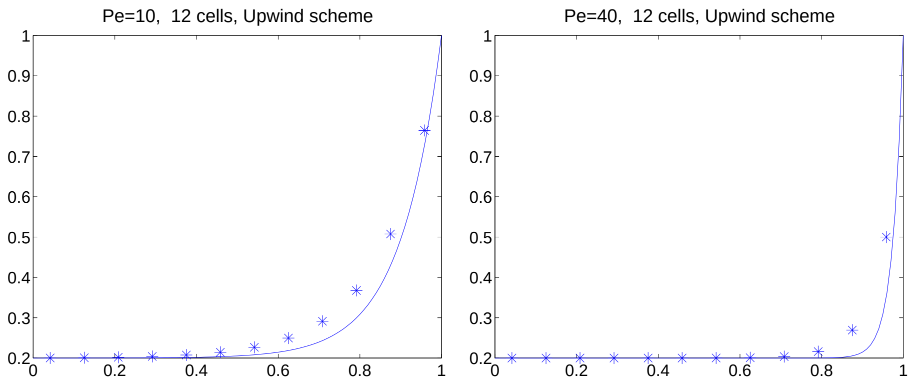

Fig. 2.3. Exact solution (—) and numerical solution (*).

satisfies the maximum principle. But the solution is smeared near the outflow boundary. It is as if the numerical solution has a smaller Péclet number than the exact solution. This is because the upwind scheme introduces *numerical diffusion*; the viscosity is increased with an *artificial viscosity coefficient* $\varepsilon_a = uh/2$. To see this, just replace ε by $\varepsilon + \varepsilon_a$ in the stencil of the central scheme (2.40): it becomes identical to the stencil of the upwind scheme (2.42).

Local grid refinement

The preceding figures show for Pe $= 40$ a rapid variation of the exact solution in a narrow zone near the outflow boundary $x = 1$. This zone is called a *boundary layer*. From the exact solution (2.13) it follows that the boundary layer thickness δ satisfies

$$\delta = \mathcal{O}(\varepsilon) = \mathcal{O}(\mathrm{Pe}^{-1}). \tag{2.43}$$

(Landau's order symbol $\mathcal{O}$ is defined later in this section). We will see later how to estimate the boundary layer thickness when an exact solution is not available. It is clear that to have reasonable accuracy in the boundary layer, the local mesh size must satisfy $h < \delta$, in order to have sufficient resolution (*i.e.* enough grid points) in the boundary layer. This is not the case in the right parts of the preceding figures. To improve the accuracy we refine the grid locally in the boundary layer. We define $\delta \equiv 6\varepsilon$; the factor 6 is somewhat arbitrary and has been determined by trial and error. We put 6 equal cells in $(0,\ 1-\delta)$ and 6 equal cells in $(1-\delta,\ 1)$. The result is shown in Fig. 2.4. Although the total number of cells remains the same, the accuracy of the

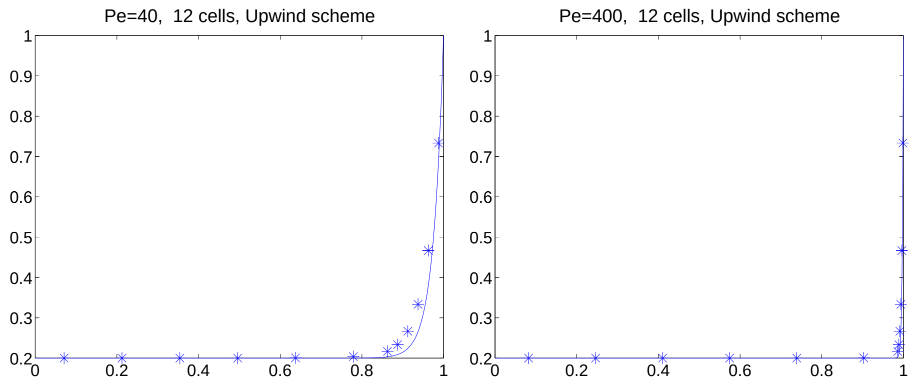

Fig. 2.4. Exact solution (—) and numerical solution (*) with local grid refinement; upwind scheme

upwind scheme has improved significantly. Even for Pe $= 400$, in which case the boundary layer is very thin, the accuracy is good.

Fig. 2.5 gives results for the central scheme (2.20). Surprisingly, the wiggles which destroyed the accuracy in Fig. 2.2 have become invisible. In the refinement zone the local mesh Péclet number satisfies $p = 1$, which is less than 2, so that according to the maximum principle there can be no wiggles in the refinement zone (see equation (2.41) and the discussion preceding (2.41)). However, inspection of the numbers shows that small wiggles remain outside the refinement zone.

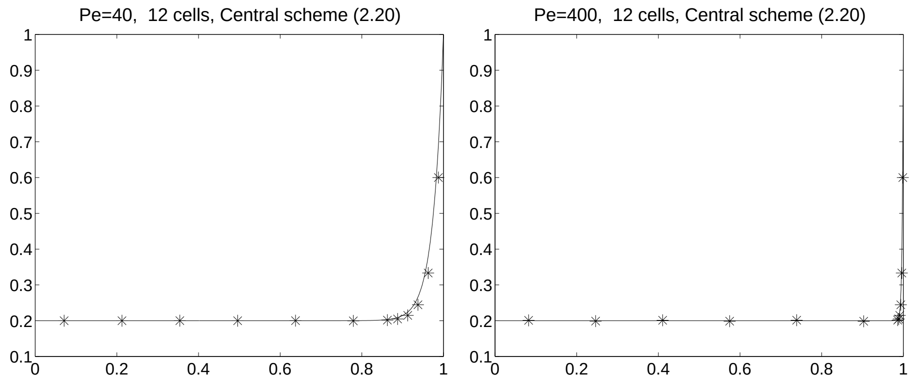

Fig. 2.5. Exact solution (—) and numerical solution (*) with local grid refinement; central scheme (2.20)

Fig. 2.6 gives results for the central scheme (2.21). This scheme might be

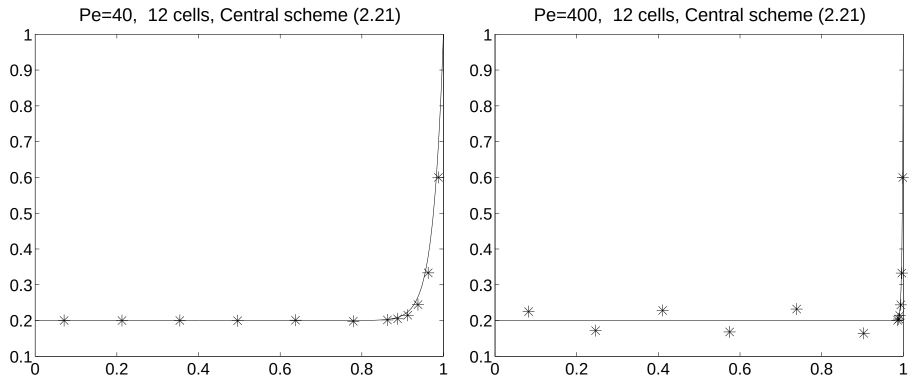

Fig. 2.6. Exact solution (—) and numerical solution (*) with local grid refinement; central scheme (2.21)

expected to be more accurate than central scheme (2.20), because linear interpolation to approximate $\varphi_{j+1/2}$ is more accurate than averaging on a nonuniform grid. However, we see that for Pe = 400 the opposite is true! Clearly, we are in need of theoretical error analysis. This subject will be touched upon later. A preliminary explanation is as follows. Let the boundary of the refinement zone be located between the nodes x_j and x_{j+1}. Call the mesh size inside and outside the refinement zone h and H, respectively. The stencil of the central scheme scheme (2.20) at x_j follows from equation (2.29) as:

$$[L_h] = \left[-\frac{1}{2} - \frac{\varepsilon}{H} \qquad \frac{\varepsilon}{H} + \frac{2\varepsilon}{h+H} \qquad \frac{1}{2} - \frac{2\varepsilon}{h+H} \right] . \tag{2.44}$$

For the central scheme (2.21) we find from equation (2.30):

$$[L_h] = \left[-\frac{1}{2} - \frac{\varepsilon}{H} \qquad -\frac{1}{2} + \frac{h}{h+H} + \frac{\varepsilon}{H} + \frac{2\varepsilon}{h+H} \qquad \frac{H}{h+H} - \frac{2\varepsilon}{h+H} \right] . \tag{2.45}$$

According to definition 2.3.1, one of the necessary conditions for a positive scheme scheme is that the third element in the above stencils is non-positive. For $\varepsilon \ll 1$ (and consequently $h/H \ll 1$) this element is about 1/2 in (2.44) and 1 in (2.45), which is worse. Furthermore, in (2.45) the central element is negative. We see that (2.45) deviates more from the conditions for a positive scheme than (2.44), so that it is more prone to wiggles.

Péclet-uniform accuracy and efficiency

The maximum norm of the error $e_j \equiv \varphi(x_j) - \varphi_j$ is defined as

$$\|e\|_\infty \equiv \max\{|e_j|, \quad j = 1, \cdots, J\},$$

where $\varphi(x)$ is the exact solution. Table 2.1 gives results. The number of cells is the same as in the preceding figures. For Pe = 10 a uniform grid is used, for the other cases the grid is locally refined, as before. We see that our doubts

Scheme	Pe=10	Pe=40	Pe=400	Pe=4000
Upwind	.0785	.0882	.0882	.0882
Central (2.20)	.0607	.0852	.0852	.0852
Central (2.21)	.0607	.0852	.0856	.3657

Table 2.1. Maximum error norm; 12 cells.

about the central scheme (2.21) are confirmed. For the other schemes we see that $\|e\|_\infty$ is almost independent of Pe. This is due to the adaptive (i.e. Pe-dependent) local grid refinement in the boundary layer. Of course, since the number of cells J required for a given accuracy does not depend on Pe, computing work and storage are also independent of Pe. We may conclude that *computing cost and accuracy are uniform in* Pe.

This is an important observation. A not uncommon misunderstanding is that numerical predictions of high Reynolds (or Péclet in the present case) number flows are inherently untrustworthy, because numerical discretization errors ('numerical viscosity') dominate the small viscous forces. This is not true, provided appropriate measures are taken, as was just shown. Local grid refinement in boundary layers enables us to obtain accuracy independent of the Reynolds number. Because in practice Re (or its equivalent such as the Péclet number) is often very large (see Sect. 1.5) it is an important (but not impossible) challenge to realize this also in more difficult multi-dimensional situations. An analysis of Pe-uniform accuracy for a two-dimensional singular perturbation problem will be given in Sect. 3.3.

Global and local truncation error

By using Taylor's formula (see below) it is easy to see that the numerical flux as specified above (equations(2.20—2.23)) approaches the exact flux as the grid is refined, *i.e.* as

$$\Delta \downarrow 0, \quad \Delta \equiv \max\{h_j, j = 1, \cdots, J\} \ . \tag{2.46}$$

But does this mean that the difference between the numerical and exact solution goes to zero? Surprisingly, this is no simple matter, but one of the deepest questions in numerical analysis. We will present only some basic considerations. We define

Definition 2.3.2. *Global truncation error*
The *global truncation error* is defined as

$$e_j \equiv \varphi(x_j) - \varphi_j \ , \quad j = 1, \cdots, J \ ,$$

with $\varphi(x)$ the exact solution.

Truncation errors are errors that are caused by truncation (to truncate means to shorten by cutting off) of an infinite process. The process we have in mind here is the limit $\Delta \downarrow 0$; we stop at a finite value of Δ. Rounding errors are the errors that are caused by the finite precision approximation of real numbers in computer memories. In the numerical approximation of differential equations these are usually much smaller than truncation errors. Here we just assume zero rounding error.

Obviously, the global truncation error is what we are after, but it cannot be estimated directly, because the exact solution is not available. Therefore a quantity is introduced that can be estimated, namely

Definition 2.3.3. *Local truncation error*
The *local truncation error* of the discrete operator L_h is defined as

$$\tau_j \equiv L_h e_j \ , \quad j = 1, \cdots, J \ . \tag{2.47}$$

It follows that $e = L_h^{-1}\tau$, with e and τ algebraic vectors with elements e_j, τ_j. Hence

$$\|e\| \leq \|L_h^{-1}\| \|\tau\| \, .$$

This suggests that a scheme with smaller $\|\tau\|$ will have a smaller $\|e\|$ than a scheme with a larger $\|\tau\|$. But this need not be so, because L_h is different for the two schemes, so that $\|L_h^{-1}\|$ is different. To improve our insight in accuracy, we will now dive into a somewhat complicated but elementary analysis.

Estimate of local truncation error in the interior

The purpose of the following elementary but laborious analysis is to eliminate two common misunderstandings. The first is that grids should be smooth for accuracy; this is not true in general, at least not for positive schemes (Definition 2.3.1). The second is that a large local truncation error at a boundary causes a large global truncation error; this is also not true in general.

We begin with estimating the local truncation error. For simplicity u and ε are assumed constant. We select the central scheme for convection. The scheme (2.40) with a Dirichlet condition at $x = 0$ and a Neumann condition at $x = 1$ (*cf.* equation (2.29)) can be written as

$$\begin{aligned}
L_h\varphi_1 \equiv & \left(\frac{u}{2} + \frac{\varepsilon}{h_{3/2}} + \frac{2\varepsilon}{h_1}\right)\varphi_1 + \left(\frac{u}{2} - \frac{\varepsilon}{h_{3/2}}\right)\varphi_2 = h_1 q_1 + \left(u + \frac{2\varepsilon}{h_1}\right) a\ , \\
L_h\varphi_j \equiv & -\left(\frac{u}{2} + \frac{\varepsilon}{h_{j-1/2}}\right)\varphi_{j-1} + \varepsilon\left(\frac{1}{h_{j-1/2}} + \frac{1}{h_{j+1/2}}\right)\varphi_j \\
& + \left(\frac{u}{2} - \frac{\varepsilon}{h_{j+1/2}}\right)\varphi_{j+1} = h_j q_j, \qquad j = 2, \cdots, J-1\ , \\
L_h\varphi_J \equiv & -\left(\frac{u}{2} + \frac{\varepsilon}{h_{J-1/2}}\right)\varphi_{J-1} + \left(\frac{u}{2} + \frac{\varepsilon}{h_{J-1/2}}\right)\varphi_J \\
= & h_J q_J - \left(\frac{u}{2}h_J - \varepsilon\right) b\ .
\end{aligned} \tag{2.48}$$

Taylor's formula

To estimate the local truncation error we need Taylor's formula:

$$f(x) = f(x_0) + \sum_{k=1}^{n-1} \frac{1}{k!}\frac{d^k f(x_0)}{dx^k}(x - x_0)^k + \frac{1}{n!}\frac{d^n f(\xi)}{dx^n}(x - x_0)^n \tag{2.49}$$

for some ξ between x and x_0. Of course, f must be sufficiently differentiable. This gives for the exact solution, writing $\varphi^{(k)}$ for $d^k\varphi(x)/dx^k$,

$$\begin{aligned}
\varphi(x_{j\pm1}) = & \varphi(x_j) \pm h_{j\pm1/2}\varphi^{(1)}(x_j) + \frac{1}{2}h_{j\pm1/2}^2\varphi^{(2)}(x_j) \pm \frac{1}{6}h_{j\pm1/2}^3\varphi^{(3)}(x_j) \\
& + \frac{1}{24}h_{j\pm1/2}^4\varphi^{(4)}(x_j) + \mathcal{O}(h_{j\pm1/2}^5)\ ,
\end{aligned} \tag{2.50}$$

where $\mathcal{O}$ is Landau's order symbol, defined as follows:

Definition 2.3.4. *Landau's order symbol*
A function $f(h) = \mathcal{O}(h^p)$ if there exist a constant M independent of h and a constant $h_0 > 0$ such that

$$\frac{|f(h)|}{h^p} < M, \quad \forall h \in (0, h_0) .$$

The relation $f(h) = \mathcal{O}(h^p)$ is pronounced as "f is of order h^p".

Estimate of local truncation error, continued

We will now see that although we do not know the exact solution, we can nevertheless determine the dependence of τ_j on h_j. We substitute (2.50) in $L_h(\varphi(x_j)$, and obtain, after some tedious work that cannot be avoided, for $j = 2, \cdots, J-1$:

$$\begin{aligned} L_h\varphi(x_j) = &\ \tilde{L}_h\varphi(x_j) + \mathcal{O}(\Delta^4) , \\ \tilde{L}_h\varphi(x_j) \equiv &\ \frac{1}{2}q_j(h_{j-1/2} + h_{j+1/2}) \\ &+ \left(\frac{1}{4}u\varphi^{(2)} - \frac{1}{6}\varepsilon\varphi^{(3)}\right)(h^2_{j+1/2} - h^2_{j-1/2}) \\ &+ \left(\frac{1}{12}u\varphi^{(3)} - \frac{1}{24}\varepsilon\varphi^{(4)}\right)(h^3_{j+1/2} + h^3_{j-1/2}) , \end{aligned} \tag{2.51}$$

where $\varphi^{(n)} = d^n\varphi(x_j)/dx^n$. We have

$$\tau_j = L_h e_j = L_h[\varphi(x_j) - \varphi_j] = \tilde{L}_h\varphi(x_j) - h_j q_j + \mathcal{O}(\Delta^4) ,$$

so that we obtain:

$$\begin{aligned} \tau_j = &\ \frac{1}{2}q_j(h_{j-1/2} - 2h_j + h_{j+1/2}) \\ &+ \left(\frac{1}{4}u\varphi^{(2)} - \frac{1}{6}\varepsilon\varphi^{(3)}\right)(h^2_{j+1/2} - h^2_{j-1/2}) \\ &+ \left(\frac{1}{12}u\varphi^{(3)} - \frac{1}{24}\varepsilon\varphi^{(4)}\right)(h^3_{j+1/2} + h^3_{j-1/2}) + \mathcal{O}(\Delta^4) . \end{aligned} \tag{2.52}$$

The grid is called smooth if the mesh size h_j varies slowly, or more precisely, if

$$|h_{j+1/2} - h_{j-1/2}| = \mathcal{O}(\Delta^2) \quad \text{and} \quad |h_{j-1/2} - 2h_j + h_{j+1/2}| = \mathcal{O}(\Delta^3) .$$

Therefore on smooth grids $\tau_j = \mathcal{O}(\Delta^3)$, but on rough grids $\tau_j = \mathcal{O}(\Delta)$. Therefore it is often thought that one should always work with smooth grids for better accuracy, but, surprisingly, this is not necessary in general. We will show why later. Note that the locally refined grid used in the preceding numerical experiments is rough, but nevertheless the accuracy was found to be satisfactory.

Estimate of local truncation error at the boundaries

For simplicity we now assume the grid uniform, with $h_j \equiv h$. Let the scheme be cell-centered. We start with the Dirichlet boundary $x = 0$. Proceeding as before, we find using Taylor's formula for $\varphi(x_2)$ in the first equation of (2.48),

$$\begin{aligned}
&L_h\varphi(x_1) = \tilde{L}_h\varphi(x_1) + \mathcal{O}(h^2) \ , \\
&\tilde{L}_h\varphi(x_1) \equiv (u + 2\varepsilon/h)\varphi(x_1) - \varepsilon\varphi^{(1)} + \frac{1}{2}q_1 h \ ,
\end{aligned}$$

where $\varphi^{(1)} = d\varphi(x_1)/dx$, and $\varphi(x)$ is the exact solution. We write

$$\begin{aligned}
\tau_1 &= L_h[\varphi(x_1) - \varphi_1] = \tilde{L}_h\varphi(x_1) - hq_1 - (u + 2\varepsilon/h)a + \mathcal{O}(h^2) \\
&= (u + 2\varepsilon/h)[\varphi(x_1) - a] - \varepsilon\varphi^{(1)} - \frac{1}{2}hq_1 + \mathcal{O}(h^2) \ .
\end{aligned}$$

We use Taylor's formula for $a = \varphi(0)$:

$$a = \varphi(0) = \varphi(x_1) - \frac{1}{2}h\varphi^{(1)} + \frac{1}{8}h^2\varphi^{(2)} + \mathcal{O}(h^3)$$

and find

$$\tau_1 = \frac{h}{4}\varepsilon\varphi^{(2)} + \mathcal{O}(h^2) \ . \tag{2.53}$$

In the interior we have $\tau_j = \mathcal{O}(h^3)$ on a uniform grid, as seen from equation (2.52). However, it is not necessary to improve the local accuracy near a Dirichlet boundary, which is one of the important messages of this section; we will show this below. But first we will estimate the local truncation error at the Neumann boundary $x = 1$. By using Taylor's formula for $\varphi(x_{J-1})$ in the third equation of (2.48) we get

$$\begin{aligned}
&L_h\varphi(x_J) = \tilde{L}_h\varphi(x_J) + \mathcal{O}(h^3) \, , \\
&\tilde{L}_h\varphi(x_J) \equiv \varepsilon\varphi^{(1)} + \frac{1}{2}q_J h - \left(\frac{u}{4}\varphi^{(2)} - \frac{\varepsilon}{6}\varphi^{(3)}\right) h^2 + \mathcal{O}(h^3) \ ,
\end{aligned}$$

where $\varphi^{(n)} = d^n\varphi(x_J)/dx^n$. We write

$$\begin{aligned}
\tau_J &= L_h[\varphi(x_J) - \varphi_J] = \tilde{L}_h\varphi(x_J) - h_J q_J + (uh_J/2 - \varepsilon)b + \mathcal{O}(h^3) \\
&= \varepsilon\varphi^{(1)} - q_J h/2 - (u\varphi^{(2)}/4 - \varepsilon\varphi^{(3)}/6)h^2 + (uh/2 - \varepsilon)b + \mathcal{O}(h^3) \ .
\end{aligned}$$

We use Taylor's formula for $b = d\varphi(1)/dx$:

$$b = \varphi^{(1)} + \frac{1}{2}h\varphi^{(2)} + \frac{1}{8}h^2\varphi^{(3)} + \mathcal{O}(h^3)$$

and find

$$\tau_J = \frac{1}{24}\varepsilon\varphi^{(3)}h^2 + \mathcal{O}(h^3)\ . \tag{2.54}$$

Error estimation with the maximum principle

The student is not expected to be able to carry out the following error analysis independently. This analysis is presented merely to make our assertions about accuracy on rough grids and at boundaries really convincing. We will use the maximum principle to derive estimates of the global truncation error from estimates of the local truncation error.

By $e < E$ we mean $e_j < E_j,\ j = 1, \cdots, J$ and by $|e|$ we mean the grid function with values $|e_j|$. We recall that the global and local truncation error are related by

$$L_h e = \tau\ . \tag{2.55}$$

Suppose we have a grid function E, which will be called a *barrier function*, such that

$$L_h E \geq |\tau|\ . \tag{2.56}$$

We are going to show: $|e| \leq E$. From (2.55) and (2.56) it follows that

$$L_h(\pm e - E) \leq 0\ .$$

Let the numerical scheme (2.48) satisfy the conditions of the corollary of Theorem 2.3.1; this is the case if

$$\frac{|u|h_{j+1/2}}{\varepsilon} < 2, \quad j = 1, \cdots, J-1\ . \tag{2.57}$$

Then the corollary says

$$\pm e_j - E_j \leq \pm e_1 - E_1, \quad j = 2, \cdots, J\ . \tag{2.58}$$

Next we show that $|e_1| \leq E_1$. From $L_h(\pm e_1 - E_1) \leq 0$ it follows (with the use of (2.58) for $j = 2$) that

$$\begin{aligned} &a(\pm e_1 - E_1) \leq b(\pm e_2 - E_2) \leq b(\pm e_1 - E_1)\ , \\ &a = u/2 + 3\varepsilon/h_1, \quad b = \varepsilon/h_1 - u/2\ , \end{aligned}$$

where we assume $h_2 = h_1$. Note that $0 < b < a$. Therefore $\pm e_1 - E_1 \leq 0$, hence $|e_1| \leq E_1$. Substitution in (2.58) results in

$$|e_j| \leq E_j, \quad j = 1, \cdots, J\ . \tag{2.59}$$

which we wanted to show. It remains to construct a suitable barrier function E. Finding a suitable E is an art.

Global error estimate on uniform grid

First, assume the grid is uniform: $h_j = h$. We choose the barrier function as follows:

$$E_j = M\psi(x_j), \quad \psi(x) \equiv 1 + 3x - x^2 , \tag{2.60}$$

with M a constant still to be chosen. We find (note that $u > 0$; otherwise the boundary conditions would be ill-posed for $\varepsilon \ll 1$, as seen in Sect. 2.2):

$$\begin{aligned}
&L_h\psi(x_1) = u(1 + 3h - 5h^2/4) + \frac{\varepsilon}{h}(2 + 3h^2/2) > 2\varepsilon/h \quad \text{for } h \text{ small enough} ,\\
&L_h\psi(x_j) = uh(3 - 2x_j) + 2\varepsilon h > 2\varepsilon h, \quad j = 2, \cdots , J - 1 ,\\
&L_h\psi(x_J) = \varepsilon(1 + 2h) + uh(1/2 + h) > \varepsilon .
\end{aligned}$$

According to equations (2.52)–(2.54) there exist constants M_1 , M_2 , M_3 such that for h small enough

$$\begin{aligned}
&\tau_1 < M_1 h ,\\
&\tau_j < M_2 h^3 , \quad j = 2, \cdots , J - 1 ,\\
&\tau_J < M_3 h^2 .
\end{aligned}$$

Hence, with

$$M = \frac{h^2}{\varepsilon} \max\{M_1/2,\ M_2/2,\ M_3\}$$

condition (2.56) is satisfied, so that

$$|e| < E = \mathcal{O}(h^2) .$$

This shows that the fact that the local truncation errors at the boundaries are of lower order than in the interior does not have a bad effect on the global truncation error.

Global error estimate on nonuniform grid

Next, we consider the effect of grid roughness. From equation (2.52) we see that in the interior

$$\tau_j = \mathcal{O}(\mathit{\Delta}), \quad \mathit{\Delta} = \max\{h_j,\ j = 1, \cdots J\} .$$

We will show that nevertheless $e = \mathcal{O}(\mathit{\Delta}^2)$, as for a uniform grid. The barrier function used before does not dominate τ sufficiently. Therefore we use the following stratagem. Define the following grid functions:

$$\mu^1_j \equiv h^2_j , \quad \mu^2_j \equiv \sum_{k=1}^{j} h^3_{k-1/2} , \quad \mu^3_j \equiv \sum_{k=1}^{j} (h^2_k + h^2_{k-1}) h_{k-1/2} ,$$

where $h_0 \equiv 0$. We find with L_h defined by (2.48) and $\psi_k(x)$, $k = 1, 2, 3$ smooth functions to be chosen later:

$$\begin{aligned} L_h(\psi_1(x_j)\mu_j^1) &= \varepsilon\psi_1(x_j)(-2h_{j+1} + 4h_j - 2h_{j-1}) + && (2.61)\\ &+ \{\varepsilon\frac{d\psi_1(x_j)}{dx} - \frac{1}{2}u\psi_1(x_j)\}(h_{j-1}^2 - h_{j+1}^2) + \mathcal{O}(\varDelta^3) , \\ L_h(\psi_2(x_j)\mu_j^2) &= \varepsilon\psi_2(x_j)(h_{j-1/2}^2 - h_{j+1/2}^2) + \mathcal{O}(\varDelta^3) , && (2.62)\\ L_h(\psi_3(x_j)\mu_j^3) &= \varepsilon\psi_3(x_j)(h_{j-1}^2 - h_{j+1}^2) + \mathcal{O}(\varDelta^3) . && (2.63) \end{aligned}$$

We choose

$$\psi_1 = -\frac{q(x)}{8\varepsilon} , \quad \psi_2 = \frac{1}{6}\varphi^{(3)} - \frac{u}{4\varepsilon}\varphi^{(2)} ,$$
$$\psi_3 = -\frac{d\psi_1}{dx} + \frac{u}{2\varepsilon}\psi_1$$

and define

$$e_j^k \equiv \psi_k(x_j)\mu_j^k , \quad k = 1, 2, 3 .$$

Remembering (2.55), comparison of (2.61)–(2.63) with (2.52) shows that

$$L_h(e_j - e_j^1 - e_j^2 - e_j^3) = \mathcal{O}(\varDelta^3) . \tag{2.64}$$

The right-hand side is of the same order as the local truncation error in the uniform grid case, and can be dominated by the barrier function (2.60) with $M = C\varDelta^2$, with C a constant that we will not bother to specify further. For simplicity we assume that $h_2 = h_1$ and $h_{J-1} = h_J$, so that the situation at the boundaries is the same as in the case of the uniform grid. Hence

$$|e_j - e_j^1 - e_j^2 - e_j^3| < C\varDelta^2(1 + 3x_j - x_j^2)$$

Since $e_j^k = \mathcal{O}(\varDelta^2)$, $k = 1, 2, 3$ we find

$$e_j = \mathcal{O}(\varDelta^2) . \tag{2.65}$$

which is what we wanted to show. Hence, the scheme defined by (2.48) has second order convergence on arbitrary grids, so that its widespread application is justified.

Vertex-centered grid

On a vertex-centered grid we have grid points on the boundary. Therefore the Dirichlet boundary condition at $x = 0$ gives zero local truncation error, which is markedly better than (2.53). Furthermore, because the cell boundaries are now midway between the nodes, the cell face approximation (2.20) is much more accurate. Indeed, the local truncation error is an order smaller on rough grids for vertex-centered schemes; we will not show this. Therefore it is sometimes thought that vertex-centered schemes are more accurate than cell-centered schemes. But this is not so. In both cases, $e = \mathcal{O}(\Delta^2)$. Because this is most surprising for cell-centered schemes, we have chosen to elaborate this case. In practice, both types of grid are widely used.

Having come to the end of this chapter, looking again at the list of items that we wanted to cover given in Sect. 2.1 will help the reader to remind himself of the main points that we wanted to emphasize.

Exercise 2.3.1. Derive equation (2.21). (Remember that linear interpolation is exact for functions of type $f(x) = a + bx$).

Exercise 2.3.2. Assume $u > 0$. Show that for the upwind scheme the coefficients in the numerical flux are:

$$\begin{aligned}
\beta_j^0 &= u_{j+1/2} + (\varepsilon/h)_{j+1/2}, \quad j = 1, \cdots, J-1\,, \\
\beta_{j+1}^1 &= -(\varepsilon/h)_{j+1/2}, \quad j = 1, \cdots, J-1\,, \quad \beta_1^1 = -2\varepsilon_{1/2}/h_1\,, \\
\gamma_0 &= (u_{1/2} + 2\varepsilon_{1/2}/h_1)a\,, \\
\beta_J^0 &= u_{J+1/2}\,, \quad \gamma_1 = -\varepsilon_{J+1/2}b \quad \text{(Neumann)}, \\
\beta_J^0 &= u_{J+1/2} + 2\varepsilon_{J+1/2}/h_J\,, \quad \gamma_1 = -2\varepsilon_{J+1/2}b/h_J \quad \text{(Dirichlet)}.
\end{aligned} \tag{2.66}$$

Exercise 2.3.3. Show that with ε and u constant on a uniform grid the stencil for the central scheme with Dirichlet boundary conditions is given by

$$\begin{aligned}
[L_h]_1 &= \begin{bmatrix} 0 & 3\dfrac{\varepsilon}{h} + \dfrac{1}{2}u & \dfrac{1}{2}u - \dfrac{\varepsilon}{h} \end{bmatrix}, \\
[L_h]_j &= \begin{bmatrix} -\dfrac{1}{2}u - \dfrac{\varepsilon}{h} & 2\dfrac{\varepsilon}{h} & \dfrac{1}{2}u - \dfrac{\varepsilon}{h} \end{bmatrix}, \qquad j = 2, \cdots, J-1\,, \\
[L_h]_J &= \begin{bmatrix} -\dfrac{1}{2}u - \dfrac{\varepsilon}{h} & 3\dfrac{\varepsilon}{h} - \dfrac{1}{2}u & 0 \end{bmatrix}.
\end{aligned} \tag{2.67}$$

Exercise 2.3.4. Show that in the refinement zone the local mesh Péclet number satisfies $p = 1$.

Exercise 2.3.5. Derive equations (2.44) and (2.45).

Exercise 2.3.6. Implement a Neumann boundary condition at $x = 1$ in the MATLAB program `cd1`. Derive and implement the corresponding exact solution. Study the error by numerical experiments. Implement wrong boundary conditions: Neumann at inflow, Dirichlet at outflow. See what happens.

Exercise 2.3.7. In the program `cd1` the size of the refinement zone is `del = 6/pe` . Find out how sensitive the results are to changes in the factor 6.

Exercise 2.3.8. Let $x_j = jh$ (uniform grid) and denote $\varphi(x_j)$ by φ_j. Show:

$$(\varphi_j - \varphi_{j-1})/h = \frac{d\varphi(x_j)}{dx} - \frac{h}{2}\frac{d^2\varphi(\xi)}{dx^2}\,, \tag{2.68}$$

$$(\varphi_{j+1} - \varphi_{j-1})/(2h) = \frac{d\varphi(x_j)}{dx} + \frac{h^2}{6}\frac{d^3\varphi(\xi)}{dx^3}\,, \tag{2.69}$$

$$(\varphi_{j-1} - 2\varphi_j + \varphi_{j+1})/h^2 = \frac{d^2\varphi(x_j)}{dx^2} + \frac{h^2}{12}\frac{d^4\varphi(\xi)}{dx^4}\,, \tag{2.70}$$

$$\tag{2.71}$$

with $\xi \in [x_{j-1}\,, x_{j+1}]$.

Exercise 2.3.9. Show that

$$\frac{\sin x}{\sqrt{x}} = \mathcal{O}(\sqrt{x}).$$

Some self-test questions

When is the convection-diffusion equation in conservation form?

Write down the Burgers equation.

When do we call a problem ill-posed?

Formulate the maximum principle.

When is a finite volume scheme in conservation form?

What are cell-centered and vertex-centered grids?

What are the conditions for a scheme to be of positive type? Which desirable property do positive schemes have?

Define the mesh-Péclet number.

Why is it important to have Péclet-uniform accuracy and efficiency?

Derive a finite volume scheme for the convection-diffusion equation.

Derive the condition to be satisfied by the step size h for the central scheme to be of positive type on a uniform grid.

Define the global and local truncation error.

Derive the exact solution of the convection-diffusion equation.

Write down Taylor's formula.

3. The stationary convection-diffusion equation in two dimensions

3.1 Introduction

We will discuss only new aspects that did not come up in the one-dimensional case. The equation to be studied is the two-dimensional stationary convection-diffusion equation:

$$\frac{\partial u\varphi}{\partial x}+\frac{\partial v\varphi}{\partial y}-\frac{\partial}{\partial x}\left(\varepsilon\frac{\partial\varphi}{\partial x}\right)-\frac{\partial}{\partial y}\left(\varepsilon\frac{\partial\varphi}{\partial y}\right)=q(x,y)\ , \quad (x,y)\in\Omega\equiv(0,1)\times(0,1)\ . \tag{3.1}$$

Suitable boundary conditions are:

$$\begin{aligned}\varphi &= f(x,y) \quad \text{on} \quad \partial\Omega_i \quad \text{(Dirichlet)}, && (3.2)\\ \varphi &= f(x,y) \quad \text{on} \quad \partial\Omega_o \quad \text{(Dirichlet)} \quad \text{or} && (3.3)\\ \frac{\partial\varphi}{\partial n} &= g(x,y) \quad \text{on} \quad \partial\Omega_o \quad \text{(Neumann)}, && (3.4)\end{aligned}$$

where n is the outward unit normal on the boundary $\partial\Omega$, $\partial\Omega_i$ is the inflow boundary (where $\mathbf{u}\cdot\mathbf{n}<0$) and $\partial\Omega_o$ is the remainder of $\partial\Omega$, to be called the outflow boundary.

We recall that $\varepsilon=1/\mathrm{Pe}$, with the Péclet number $\mathrm{Pe}\gg 1$. In the same way as in Sect. 2.2 it can be shown that equation (3.1) is in *conservation form.*

As in one dimension, we have a *maximum principle.* We write (3.1) in the following non-conservative form:

$$u\frac{\partial\varphi}{\partial x}+v\frac{\partial\varphi}{\partial y}-\frac{\partial}{\partial x}\left(\varepsilon\frac{\partial\varphi}{\partial x}\right)-\frac{\partial}{\partial y}\left(\varepsilon\frac{\partial\varphi}{\partial y}\right)=\tilde{q}\equiv q-\varphi\,\mathrm{div}\,\mathbf{u}\ . \tag{3.5}$$

If $\tilde{q}\le 0$ then local maxima can only occur on the boundary $\partial\Omega$. We will not show this here; the interested reader may consult Sect. 2.4 of Wesseling (2001).

Purpose of this chapter

The purpose of this chapter is:

• To explain how *singular perturbation theory* can be used to predict where for Pe $\gg 1$ thin layers (*boundary layers*) will occur without knowing the exact solution;

• To show which boundary conditions are suitable for Pe $\gg 1$;

• To introduce the *finite volume method* in two dimensions;

• To show how by means of local grid refinement accuracy and computing work can be made independent of the Péclet number;

• To introduce the *stencil* of the scheme and to show how to generate its coefficient matrix;

• To explain the *discrete maximum principle* in two dimensions;

• To illustrate the above points by numerical experiments.

Exercise 3.1.1. Show that equation (3.1) is in conservation form by using Theorem 1.2.1.

3.2 Singular perturbation theory

Before discussing numerical schemes, we will consider singular perturbation theory for the stationary convection-diffusion equation in two dimensions. In view of our experience in the one-dimensional case, we expect when $\varepsilon \ll 1$ the occurrence of thin layers in which the solution varies rapidly. Such layers are called *boundary layers*. As seen in Chapt. 2, local grid refinement is required in boundary layers for accuracy. Therefore it is necessary to know where boundary layers occur and the dependence of their thickness on ε. In Chapt. 2 this information was deduced from the exact solution; however, in general the exact solution is not available, of course. But the required information is provided by *singular perturbation theory*, also called *boundary layer theory.* The boundary layer concept was first introduced by Ludwig Prandtl in 1904, but the mathematical foundation was developed in the middle of the last century.

Subcharacteristics

When $\varepsilon \ll 1$ it is natural to approximate (3.1) by putting $\varepsilon = 0$, so that we obtain, switching to nonconservative form:

$$u\frac{\partial\varphi}{\partial x} + v\frac{\partial\varphi}{\partial y} = \tilde{q}\,, \quad \tilde{q} = q - \varphi\mathrm{div}\,\mathbf{u}\,. \tag{3.6}$$

This is the *convection equation.* Let us define curves called *characteristics* in Ω space by relations $x = x(s)$ $y = y(s)$, satisfying

$$\frac{dx}{ds} = u\,, \quad \frac{dy}{ds} = v\,. \tag{3.7}$$

For the derivative along the curve we have

$$\frac{d\varphi}{ds} = \frac{\partial\varphi}{\partial x}\frac{dx}{ds} + \frac{\partial\varphi}{\partial y}\frac{dy}{ds} = u\frac{\partial\varphi}{\partial x} + v\frac{\partial\varphi}{\partial y}\,.$$

Therefore equation (3.6) reduces to

$$\frac{d\varphi}{ds} = \tilde{q}\,. \tag{3.8}$$

We see that in the homogeneous case $\tilde{q} = 0$ *the solution φ is constant along the characteristics*, which is why these curves are important. When $\varepsilon > 0$ we do not have φ constant on the characteristics, but as we will see, these curves still play an important role when $\varepsilon \ll 1$. To avoid confusion, when $\varepsilon > 0$ the characteristics are called *subcharacteristics.*

A paradox

Let the pattern of streamlines (i.e. (sub)characteristics) be qualitatively as in Fig. 3.1. The characteristic C_1 intersects the boundary in points P_1 and P_2. Here a boundary condition is given, according to equations (3.2)—(3.4). On C_1 we have $\varphi = \text{constant} = \varphi(C_1)$. It is clear that in general $\varphi(C_1)$ cannot satisfy simultaneously the boundary conditions in P_1 and in P_2. For example, let $\varphi(P_1) \neq \varphi(P_2)$ be prescribed by a Dirichlet condition at $y = 0$ and at $x = 1$. Do we have $\varphi(C_1) = \varphi(P_1)$ or $\varphi(C_1) = \varphi(P_2)$ or $\varphi(C_1) = (\varphi(P_1) + \varphi(P_2))/2$ or something else? What value to take for $\varphi(C_1)$? The difficulty has to do with the change of type that the partial differential equation (3.1) undergoes when $\varepsilon = 0$: for $\varepsilon > 0$ it is *elliptic*, for $\varepsilon = 0$ it is *hyperbolic.* It is clear that we cannot get a good approximation to equation (3.1) for $\varepsilon \downarrow 0$ by simply deleting the small diffusion term. This paradoxical situation has baffled mathematicians in the nineteenth century, who found the drag of a body in an ideal fluid (zero viscosity) to be zero, whereas in physical experiments the drag around bluff bodies was found to be appreciable, even at very high

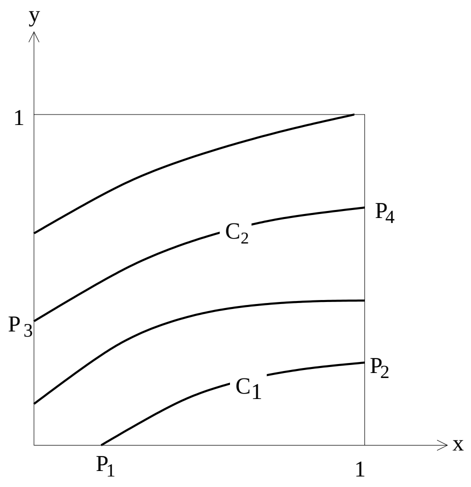

Fig. 3.1. Streamline pattern.

Reynolds numbers. This was called the *paradox of d'Alembert*. The problem remains in the nonhomogeneous case $\tilde{q} \neq 0$, because the first order equation (3.6) can satisfy only one boundary condition.

Problems that contain a small parameter are called perturbation problems. The terms that are multiplied by the small parameter are regarded as pertubations. If a good approximation can be obtained by simply neglecting the perturbations, we speak of a *regular perturbation problem.* If a good first approximation cannot be obtained in this way the perturbation problem is called *singular.* An example of a regular perturbation problem is the sun-earth-moon system. If we neglect the attraction of the moon we still get a good approximation of the orbit and the period of the earth. The above paradox shows that the convection-diffusion equation at large Péclet number is a singular perturbation problem.

Singular perturbation theory

The above paradox is resolved by *singular perturbation theory.* If we assume flow is from the right to the left in Fig. 3.1, so that $x = 1$ is an inflow boundary and $y = 0$ is an outflow boundary, then $\varphi = \varphi(C_1) = \varphi(P_2)$ is a good approximation for $\varepsilon \ll 1$ to the solution of (3.1)—(3.4) in $1 \geq y > \delta = \mathcal{O}(\varepsilon)$ (assuming $\tilde{q} = 0$), whereas (3.6) has to be replaced by a so-called *boundary layer equation* to obtain an approximation in $\delta > y \geq 0$. This can be seen as follows. First, assume that we indeed have $\varphi(C_1) = \varphi(P_2)$ in $1 \geq y > \delta$ with $\delta \ll 1$. In $\delta > y \geq 0$ we expect a rapid change of φ from $\varphi(P_2)$ to $\varphi(P_1)$. For derivatives of φ we expect

$$\frac{\partial^m \varphi}{\partial y^m} = \mathcal{O}(\delta^{-m}) \;, \tag{3.9}$$

so that perhaps the diffusion term in (3.1) cannot be neglected in the boundary layer; this will depend on the size of δ. Assume

$$\delta = \mathcal{O}(\varepsilon^{\alpha}) \;, \tag{3.10}$$

with α to be determined. In order to exhibit the dependence of the magnitude of derivatives on ε we introduce a *stretched coordinate* $\tilde{y}$:

$$\tilde{y} = y\varepsilon^{-\alpha} \;, \tag{3.11}$$

which is chosen such that $\tilde{y} = \mathcal{O}(1)$ in the boundary layer. We take ε constant for simplicity. It follows from (3.9)—(3.11) that

$$\frac{\partial^m \varphi}{\partial \tilde{y}^m} = \mathcal{O}(1) \tag{3.12}$$

in the boundary layer. In the stretched coordinate, equation (3.1) becomes

$$\frac{\partial u\varphi}{\partial x} + \varepsilon^{-\alpha}\frac{\partial v\varphi}{\partial \tilde{y}} - \varepsilon\frac{\partial^2 \varphi}{\partial x^2} - \varepsilon^{1-2\alpha}\frac{\partial^2 \varphi}{\partial \tilde{y}^2} = 0 \;. \tag{3.13}$$

Letting $\varepsilon \downarrow 0$ and using (3.12), equation (3.13) takes various forms, depending on α. The correct value of α follows from the requirement, that the solution of the $\varepsilon \downarrow 0$ limit of equation (3.13) satisfies the boundary condition at $y = 0$, and the so-called matching principle.

Matching principle

As $\tilde{y}$ increases, the solution of (the $\varepsilon \downarrow 0$ limit of) equation (3.13) has to somehow join up with the solution of (3.6), i.e. approach the value $\varphi(C_1)$. In singular perturbation theory this condition is formulated precisely, and is known as the *matching principle*:

$$\lim_{\tilde{y}\to\infty} \varphi_{\text{inner}}(x, \tilde{y}) = \lim_{y\downarrow 0} \varphi_{\text{outer}}(x, y) \;.$$

Here φ_{inner}, also called the *inner solution*, is the solution of the *inner equation* or *boundary layer equation*, which is the limit as $\varepsilon \downarrow 0$ of equation (3.13) for the correct value of α, which we are trying to determine. Furthermore, φ_{outer}, also called the *outer solution*, is the solution of the *outer equation*, which is the limit as $\varepsilon \downarrow 0$ of the original equation, i.e. equation (3.6). The matching principle becomes

$$\varphi_{\text{inner}}(x, \infty) = g(x) \equiv \lim_{y\downarrow 0} \varphi_{\text{outer}}(x, y) \;. \tag{3.14}$$

As already mentioned, the other condition to be satisfied is the boundary condition at $\tilde{y} = 0$:

$$\varphi(x, 0) = f(x) \ . \tag{3.15}$$

For $\alpha < 0$ (corresponding to compression rather than stretching) the limit as $\varepsilon \downarrow 0$ of (3.13) is, taking u constant for simplicity,

$$u \frac{\partial \varphi}{\partial x} = 0 \ , \tag{3.16}$$

so that $\varphi = \varphi(\tilde{y})$, which obviously cannot satisfy (3.14), so that the case $\alpha < 0$ has to be rejected. With $\alpha = 0$ equation (3.6) is obtained, which cannot satisfy both conditions at $x = 0$ and $y = \tilde{y} = 0$, as we saw.

For $0 < \alpha < 1$ the limit of (3.13) is, taking v constant for simplicity,

$$v \frac{\partial \varphi}{\partial \tilde{y}} = 0 \ ,$$

so that the inner solution is independent of $\tilde{y}$, hence, in general equations (3.14) and (3.15) cannot be satisfied simultaneously. This rules out the case $0 < \alpha < 1$.

For $\alpha = 1$ equation (3.13) becomes as $\varepsilon \downarrow 0$:

$$v(x, 0) \frac{\partial \varphi}{\partial \tilde{y}} - \frac{\partial^2 \varphi}{\partial \tilde{y}^2} = 0 \ , \tag{3.17}$$

where we have used that $v(x, y) = v(x, \varepsilon \tilde{y}) \to v(x, 0)$ as $\varepsilon \downarrow 0$. The general solution of (3.16) is

$$\varphi = A(x) + B(x) e^{\tilde{v}\tilde{y}} \ , \tag{3.18}$$

with $\tilde{v} = v(x, 0)$. We can satisfy both (3.14) and (3.15), remembering that we had assumed that $y = 0$ is an outflow boundary, so that $\tilde{v} < 0$. From (3.14) and (3.15) we find

$$A(x) = g(x), \quad B(x) = f(x) - g(x) \ .$$

This gives us the inner solution. In terms of the unstretched variable y the inner solution is given by

$$\varphi = g(x) + [f(x) - g(x)] e^{\tilde{v} y / \varepsilon} \ .$$

We see a rapid exponential variation from $g(x)$ to $f(x)$ in a thin layer of thickness $\delta = \mathcal{O}(\varepsilon)$, confirming our earlier statement about the behavior of the solution. Fig. 3.2 gives a sketch of the inner and outer solutions as a function of y for some given x. An asymptotic approximation for $\varepsilon \downarrow 0$ that is valid everywhere is given by $\varphi_{\text{inner}} + \varphi_{\text{outer}} - g(x)$ (not shown in the figure).

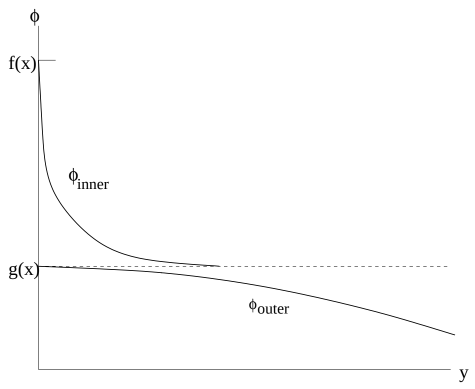

Fig. 3.2. Sketch of inner and outer solutions

The distinguished limit

The limit as $\varepsilon \downarrow 0$ of the stretched equation (3.13) for the special value $\alpha = 1$ for which the solution of the resulting inner equation can satisfy both the boundary condition and the matching principle is called the *distinguished limit*. In order to show that this limit is unique we will also investigate the remaining values of α that we did not yet consider, namely $\alpha > 1$. Now equation (3.13) gives the following inner equation:

$$\frac{\partial^2 \varphi}{\partial \tilde{y}^2} = 0 \ ,$$

with the general solution

$$\varphi = A(x) + B(x)\tilde{y} \ .$$

The limit of φ as $\tilde{y} \to \infty$ does not exist, so that the matching principle cannot be satisfied. Hence, $\alpha = 1$ is the only value that gives a distinguished limit.

The only element of arbitrariness that remains in this analysis is the assumption that we have a boundary layer at $y = 0$. Why no boundary layer at $x = 1$, and $\varphi(C_1) = \varphi(P_1)$ (cf. Fig. 3.1)?This can be investigated by assuming a boundary layer at $x = 1$, and determining whether a distinguished limit exists or not. This is left as an exercise. It turns out that boundary layers cannot arise at inflow boundaries.

The role of boundary conditions

The occurrence of boundary layers is strongly influenced by the type of boundary condition. Let (3.15) be replaced by a Neumann condition:

$$-\frac{\partial \varphi(x,0)}{\partial y} = f(x)\ . \tag{3.19}$$

As before, a boundary layer of thickness $\mathcal{O}(\varepsilon)$ is found at $y = 0$, and the boundary layer equation is given by (3.16), with general solution (3.18). Taking boundary condition (3.19) into account we find

$$B(x) = -\varepsilon f(x)/\tilde{v}\ ,$$

so that $B(x) \to 0$ as $\varepsilon \downarrow 0$. Hence, to first order, there is no boundary layer, and the outer solution (solution of (3.6)) is uniformly valid in Ω.

Parabolic and ordinary boundary layers

Those familiar with fluid dynamics may wonder at the boundary layer thickness $\mathcal{O}(\varepsilon) = \mathcal{O}(1/\mathrm{Pe})$, since in fluid dynamics laminar boundary layers have thickness $\mathcal{O}(1/\sqrt{\mathrm{Re}})$, so that one would have expected $\delta = \mathcal{O}(1/\sqrt{\mathrm{Pe}})$. We will now see that the convection-diffusion equation gives rise to two types of boundary layers.

Consider the case that $y = 0$ is a solid wall, so that $v(x,0) = 0$. The shape of the characteristics of the outer equation (3.6) might be as in Fig. 3.3, where also $y = 1$ is assumed to be a solid wall, so that we have a channel flow. Since

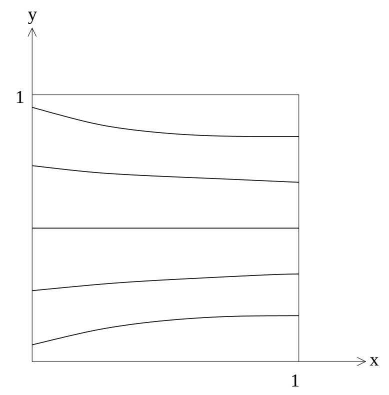

Fig. 3.3. Characteristics of equation (3.6) in a channel flow.

$v(x,0) = 0$, the wall $y = 0$ is a characteristic of the outer equation (3.6) according to (3.7), so that the solution along this characteristic is given by

$$\varphi(x,0) = f_1(0)\ , \tag{3.20}$$

assuming $x = 0$ is a inflow boundary with Dirichlet condition

$$\varphi(0, y) = f_1(y) \; . \tag{3.21}$$

Let there also be a Dirichlet condition at the wall $y = 0$:

$$\varphi(x, 0) = f_2(x) \; . \tag{3.22}$$

This condition cannot in general be satisfied by the outer solution, because it is constant along the wall, which is a characteristic. Hence, we expect a boundary layer at $y = 0$. Obviously, this boundary layer will be of different type than obtained before, because the boundary layer solution cannot be given by (3.18), since now we have $\tilde{v} = 0$. In order to derive the boundary layer equation, the same procedure is followed as before. Again, we introduce the stretched coordinate (3.11). Keeping in mind that $v = 0$, equation (3.1) goes over in

$$\frac{\partial u\varphi}{\partial x} - \varepsilon \frac{\partial^2 \varphi}{\partial x^2} - \varepsilon^{1-2\alpha} \frac{\partial^2 \varphi}{\partial \tilde{y}^2} = 0 \; . \tag{3.23}$$

The boundary condition is

$$\varphi(x, 0) = f_2(x) \; , \tag{3.24}$$

and the matching principle gives

$$\lim_{\tilde{y} \to \infty} \varphi(x, \tilde{y}) = \varphi_{\text{outer}}(x, 0) \; . \tag{3.25}$$

Now we take the limit of (3.23) as $\varepsilon \downarrow 0$. For $\alpha < 1/2$ the outer equation at $y = 0$ is recovered with solution (3.20), which cannot satisfy (3.25). For $\alpha = 1/2$ the limit of (3.23) is

$$\frac{\partial u\varphi}{\partial x} - \frac{\partial^2 \varphi}{\partial \tilde{y}^2} = 0 \; , \tag{3.26}$$

This is a parabolic partial differential equation, which in general cannot be solved explicitly, but for which it is known that boundary conditions at $\tilde{y} = 0$ and $\tilde{y} = \infty$ give a well-posed problem. Hence, $\alpha = 1/2$ gives the distinguished limit, and (3.26) is the boundary layer equation. The thickness of this type of boundary layer is $\mathcal{O}(\sqrt{\varepsilon})$, which is much larger than for the preceding type, and of the same order as laminar boundary layers in fluid dynamics, for which $\delta = O(1/\sqrt{\text{Re}})$.

In order to specify a unique solution for (3.26), in addition a boundary condition has to be specified at $x = 0$ (assuming $u > 0$). From (3.21) we obtain the following boundary condition for the boundary layer solution:

$$\varphi(0, \tilde{y}) = f_1(\tilde{y}\sqrt{\varepsilon}) \; ,$$

which to the present asymptotic order of approximation (we will not go into higher order boundary layer theory) may be replaced by

$$\varphi(0,\tilde{y}) = f_1(0)\ .$$

It is left to the reader to verify that $\alpha > 1/2$ does not give a distinguished limit.

The cause of the difference between the two boundary layer equations (3.17) (an ordinary differential equation) and (3.26) (a partial differential equation) is the angle which the characteristics of the outer equation (3.6) make with the boundary layer. In the first case this angle is nonzero (cf. Fig. 3.1),in the second case the characteristics do not intersect the boundary layer. The first type is called an *ordinary boundary layer* (the boundary layer equation is an ordinary differential equation), whereas the second type is called a *parabolic boundary layer* (parabolic boundary layer equation).

Summarizing, in the case of the channel flow depicted in Fig. 3.3, for $\varepsilon \ll 1$ there are parabolic boundary layers of thickness $\mathcal{O}(\sqrt{\varepsilon})$ at $y = 0$ and $y = 1$, and an ordinary boundary layer of thickness $\mathcal{O}(\varepsilon)$ at the outflow boundary, unless a Neumann boundary condition is prescribed there.

On outflow boundary conditions

It frequently happens that physically no outflow boundary condition is known, but that this is required mathematically. Singular perturbation theory helps to resolve this difficulty. If $\varepsilon = \mathcal{O}(1)$ such a physical model is incomplete, but for $\varepsilon \ll 1$ an artificial (invented) outflow condition may safely be used to complete the mathematical model, because this does not affect the solution to any significant extent. Furthermore, an artificial condition of Neumann type is to be preferred above one of Dirichlet type. This may be seen by means of singular perturbation theory, as follows.

Consider the following physical situation: an incompressible flow with given velocity field $\boldsymbol{u}$ through a channel, the walls of which are kept at a known temperature. We want to know the temperature of the fluid, especially at the outlet. This leads to the following mathematical model. The governing equation is (3.1), with φ the temperature. Assume $\varepsilon \ll 1$, and $u > 0$. We have φ prescribed at $x = 0$ and at $y = 0, 1$, but at $x = 1$ we know nothing. Hence, we cannot proceed with solving (3.1), either analytically or numerically. Now let us just postulate some temperature profile at $x = 1$:

$$\varphi(1, y) = f_3(y).$$

An ordinary boundary layer will occur at $x = 1$, with solution, derived in the way discussed earlier (cf. equation 3.18), given by

$$\varphi(x,y) = \varphi_{\text{outer}}(1,y) + \{f_3(y) - \varphi_{\text{outer}}(1,y)\}e^{\tilde{u}(x-1)\varepsilon}\ , \tag{3.27}$$

where $\tilde{u} \equiv u(1, y)$. This shows that the invented temperature profile $f_3(y)$ influences the solution only in the thin (artificially generated) boundary layer at $x = 1$. This means that the computed temperature outside this boundary layer will be correct, regardless what we take for $f_3(y)$. When $\varepsilon = \mathcal{O}(1)$ this is no longer true, and more information from physics is required, in order to specify $f_3(y)$ correctly. In physical reality there will not be a boundary layer at all at $x = 1$, of course. Therefore a more satisfactory artificial outflow boundary condition is

$$\frac{\partial \varphi(1, y)}{\partial x} = 0 ,$$

since with this Neumann boundary condition there will be no boundary layer at $x = 1$ in the mathematical model.

Exercise 3.2.1. Show that there is no boundary layer at $x = 1$, if this is an inflow boundary. Hint: choose as stretched coordinate $\tilde{x} = (x - 1)/\varepsilon^{\alpha}$.

Exercise 3.2.2. Consider equation (2.5). Show that with Dirichlet boundary conditions for $\varepsilon \ll 1$ there is a boundary layer of thickness $\mathcal{O}(\varepsilon)$ at $x = 1$ and not at $x = 0$.

Exercise 3.2.3. Derive equation (3.27).

3.3 Finite volume method

Problem statement

In this section we study the numerical approximation of the two-dimensional stationary convection-diffusion equation, for convenience written in Cartesian tensor notation:

$$L\varphi \equiv (u_\alpha \varphi)_{,\alpha} - (\varepsilon \varphi_{,\alpha})_{,\alpha} = q , \quad \alpha = 1, 2 , \quad (x_1, x_2) \in (0, 1) \times (0, 2) . \quad (3.28)$$

We assume that we have solid walls at $x_2 = 0, 2$, so that $u_2(x_1, 0) = u_2(x_1, 2)$ $= 0$. Let $u_1 < 0$, so that $x_1 = 0$ is an outflow boundary. In view of what we learned in Sect. 3.2 we choose a Neumann boundary condition at $x_1 = 0$, and Dirichlet boundary conditions at the other parts of the boundary:

$$\begin{array}{rclrcl} \varphi_{,1}(0, x_2) & = & g^4(x_2) , & \varphi(x_1, 0) & = & g^1(x_1) , \\ \varphi(x_1, 2) & = & g^2(x_1) , & \varphi(1, x_2) & = & g^3(x_2) . \end{array}$$

This corresponds to the channel flow problem studied before. We assume symmetry with respect to the centerline of the channel, so that we have to solve only in half the domain, i.e. in $\Omega \equiv (0, 1) \times (0, 1)$. At the centerline our boundary condition is the symmetry condition $\varphi_{,2} = 0$, so that the boundary conditions are:

$$\begin{aligned} \varphi_{,1}(0, x_2) &= g^4(x_2) \, , & \varphi(x_1, 0) &= g^1(x_1) \, , \\ \varphi_{,2}(x_1, 1) &= 0 \, , & \varphi(1, x_2) &= g^3(x_2) \, . \end{aligned} \tag{3.29}$$

As discussed before, it is best to choose at the outflow boundary a homogeneous Neumann condition, *i.e.* $g^4 \equiv 0$, but for the purpose of numerical experimentation we leave the possibility of choosing a nonhomogeneous Neumann condition open. We will not discuss the three-dimensional case, because this does not provide new insights.

Our purpose in this section is to show, as in Sect. 2.3, but this time in two dimensions, that (3.28) can be solved numerically such that *accuracy and computing cost are uniform in* Pe. Therefore fear that it is impossible to compute high Péclet (Reynolds) number flow accurately is unfounded, as argued before. For simplicity we assume horizontal flow: $u_2 \equiv 0$, and we will simply write u instead of u_1.

Choice of grid

In view of what we learned in Sect. 3.2, we expect a parabolic (because $u_2 = 0$) boundary layer at $x_2 = 0$ with thickness $\mathcal{O}(\sqrt{\varepsilon})$. If $g^4 \equiv 0$ we have a homogeneous Neumann condition at the outflow boundary, and there is no boundary layer at $x_1 = 0$. Just as in Sect. 2.3, in order to make accuracy and computing work uniform in ε, we choose a grid with local refinement in the boundary layer, as sketched in Fig. 3.4. We will apply grid refinement near the outflow boundary later. The region of refinement has thickness $\sigma = \mathcal{O}(\sqrt{\varepsilon})$. The precise choice of σ will be discussed later, and is such that the boundary layer falls inside the refinement region. The refined part of the grid is called G_f, the interface between the refined and unrefined parts is called Γ and the remainder of the grid is called G_c. The mesh sizes in G_f and G_c are uniform, as indicated in Fig. 3.4. Note that the location of the horizontal grid lines depends on ε.

Finite volume discretization

We choose a cell-centered scheme. The cell centers are labeled by integer two-tuples (i, j) in the usual way: Ω_{ij} is the cell with center at (x_i, y_j). Hence, for example, $(i + 1/2, j)$ refers to the center of a vertical cell edge. Cell-centered finite volume discretization is used as described in Sect. 2.3. For completeness the discretization is summarized below. The finite volume method gives by integration over Ω_{ij} and by using the divergence theorem:

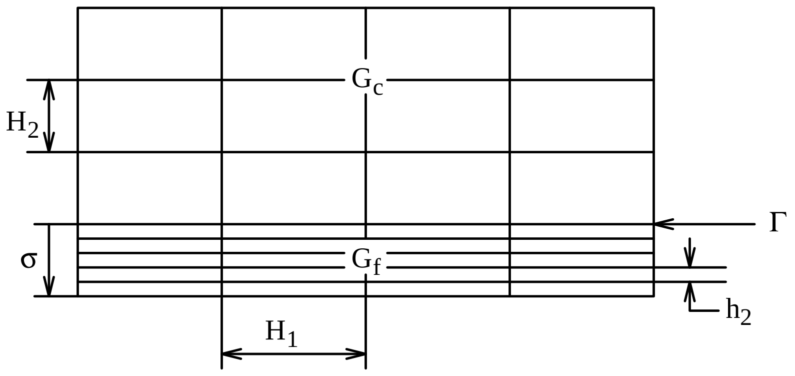

Fig. 3.4. Computational grid

$$\int_{\Omega_{ij}} L\varphi d\Omega = \left[\int_{\mathbf{x}_{i+1/2,j-1/2}}^{\mathbf{x}_{i+1/2,j+1/2}} - \int_{\mathbf{x}_{i-1/2,j-1/2}}^{\mathbf{x}_{i-1/2,j+1/2}}\right](u\varphi - \varepsilon\varphi_{,1})dx_2$$

$$+\left[\int_{\mathbf{x}_{i-1/2,j+1/2}}^{\mathbf{x}_{i+1/2,j+1/2}} - \int_{\mathbf{x}_{i-1/2,j-1/2}}^{\mathbf{x}_{i+1/2,j-1/2}}\right](-\varepsilon\varphi_{,2})dx_1$$

$$= F^1|_{i-1/2,j}^{i+1/2,j} + F^2|_{i,j-1/2}^{i,j+1/2} .$$

The approximation of the numerical fluxes $F^{1,2}$ is given below. The right-hand side of equation (3.28) is numerically integrated over Ω_{ij} as follows:

$$\int_{\Omega_{ij}} q d\Omega \cong \tilde{q}_{ij} \equiv H_1 K_j q(\mathbf{x}_{ij}) , \qquad (3.30)$$

where K_j is the vertical dimension of Ω_{ij} : $K_j = h_2$ in G_f and $K_j = H_2$ in G_c. The cells are numbered $i = 1, ..., I$ and $j = 1, ..., J$ in the x_1- and x_2-directions, respectively. The following scheme is obtained:

$$L_h\varphi_{ij} \equiv F^1|_{i-1/2,j}^{i+1/2,j} + F^2|_{i,j-1/2}^{i,j+1/2} = \tilde{q}_{ij} . \qquad (3.31)$$

If we sum (3.31) over all cells only boundary fluxes remain, so that the scheme is *conservative* (cf. Sect. 2.3).

The numerical flux

How to approximate the numerical fluxes $F^{1,2}$ in terms of neighboring grid function values follows directly from the one-dimensional case discussed in the preceding chapter. With upwind discretization for the first derivative (taking into account that $u < 0$), the numerical fluxes $F^{1,2}$ are approximated as follows:

$$\begin{aligned} F^1_{i+1/2,j} &\cong K_j[u_{i+1/2,j}\varphi_{i+1,j} - \varepsilon(\varphi_{i+1,j} - \varphi_{ij})/H_1] , \\ F^2_{i,j+1/2} &\cong -2H_1\varepsilon(\varphi_{i,j+1} - \varphi_{ij})/(K_j + K_{j+1}) , \end{aligned} \tag{3.32}$$

With the central scheme for the first derivative F^1 becomes:

$$F^1_{i+1/2,j} \cong K_j[u_{i+1/2,j}(\varphi_{i,j} + \varphi_{i+1,j})/2 - \varepsilon(\varphi_{i+1,j} - \varphi_{ij})/H_1] . \tag{3.33}$$

The boundary conditions are implemented as in Sect. 2.3. This gives both for the upwind and central schemes:

$$\begin{aligned} F^1_{1/2,j} &\cong K_j[u_{1/2,j}(\varphi_{1j} - H_1 g^4(y_j)/2) - \varepsilon g^4(y_j)] , \\ F^1_{I+1/2,j} &\cong K_j[u_{I+1/2,j}\varphi_{I+1/2,j} - 2\varepsilon(\varphi_{I+1/2,j} - \varphi_{Ij})/H_1] , \\ F^2_{i,1/2} &\cong -2H_1\varepsilon(\varphi_{i,1} - \varphi_{i,1/2})/h_2 , \\ F^2_{i,J+1/2} &= 0 , \end{aligned} \tag{3.34}$$

where $\varphi_{I+1/2,j}$ and $\varphi_{i,1/2}$ are given by the boundary conditions (3.29).

The stencil of the scheme

Although this is tedious, we will spell out more details of the scheme, as preparation for a MATLAB program. The numerical fluxes as specified above can be written as

$$\begin{aligned} F^1_{i+1/2,j} &= \beta^0_{ij}\varphi_{ij} + \beta^1_{i+1,j}\varphi_{i+1,j}, \quad i = 1,\cdots,I-1 , \ j = 1,\cdots,J , \\ F^1_{1/2,j} &= \beta^1_{1j}\varphi_{1,j} + \gamma^0_j , \quad j = 1,\cdots,J , \\ F^1_{I+1/2,j} &= \beta^0_{Ij}\varphi_{Ij} + \gamma^1_j , \quad j = 1,\cdots,J , \\ F^2_{i,j+1/2} &= \beta^2_{ij}\varphi_{ij} + \beta^3_{i,j+1}\varphi_{i,j+1}, \quad i = 1,\cdots,I , \ j = 1,\cdots,J-1 , \\ F^2_{i,1/2} &= \beta^3_{i1}\varphi_{i,1} + \gamma^2_i , \quad i = 1,\cdots,I , \\ F^2_{i,J+1/2} &= 0 , \quad i = 1,\cdots,I , \end{aligned} \tag{3.35}$$

where $\gamma^0, \gamma^1, \gamma^2$ are known terms arising from the boundary conditions. The β and γ coefficients follow easily from (3.34), and will not be written down. The scheme consists of a linear system of equations of the form

$$L_h\varphi_{ij} = \sum_{k=-1}^{1}\sum_{l=-1}^{1}\alpha^{kl}_{ij}\varphi_{i+k,j+l} , \tag{3.36}$$

where the only nonzero α coefficients are

$$\begin{aligned} \alpha^{-1,0}_{ij} &= -\beta^0_{i-1,j} , \quad \alpha^{1,0}_{ij} = \beta^1_{ij} \\ \alpha^{0,-1}_{ij} &= -\beta^2_{i,j-1} , \quad \alpha^{0,1}_{ij} = \beta^3_{ij} \\ \alpha^{0,0}_{ij} &= (\beta^0 - \beta^1 + \beta^2 - \beta^3)_{ij} . \end{aligned} \tag{3.37}$$

The scheme has a five-point stencil:

$$[L_h] = \begin{bmatrix} & \alpha^{0,1} & \\ \alpha^{-1,0} & \alpha^{0,0} & \alpha^{1,0} \\ & \alpha^{0,-1} & \end{bmatrix} .$$

The maximum principle

Just as in the one-dimensional case, we have a discrete maximum principle. We generalize Definition 2.3.1 to the two-dimensional case as follows:

Definition 3.3.1. The operator L_h is *of positive type* if for $i = 2, \cdots, I-1$ and $j = 2, \cdots, J-1$

$$\sum_{kl} \alpha_{ij}^{kl} = 0 \tag{3.38}$$

and

$$\alpha_{ij}^{kl} < 0, \quad (k,l) \neq (0,0) \ . \tag{3.39}$$

The following theorem says that schemes of positive type satisfy a similar maximum principle as the differential equation.

Theorem 3.3.1. *Discrete maximum principle.*
If L_h is of positive type and

$$L_h \varphi_{ij} \leq 0, \quad i = 2, \cdots, I-1, \quad j = 2, \cdots, J-2 \ ,$$

then $\varphi_{ij} \leq \max_{ij}\{\varphi_{1j},\ \varphi_{Ij},\ \varphi_{i1},\ \varphi_{iJ}\}$.

In other words, local maxima can only occur in cells adjacent to the boundaries.

It is left to the reader to show that with the upwind scheme L_h is of positive type, and with the central scheme this is the case if (taking u constant for simplicity) the *mesh Péclet number* satisfies

$$p < 2 \ , \quad p \equiv \frac{|u|h}{\varepsilon} \ ,$$

just as in the one-dimensional case (cf. equation (2.41)).

The matrix of the scheme

In matrix notation the scheme (3.31) can be denoted as

$$Ay = b \ .$$

Let the algebraic vector y contain the unknowns in *lexicographic order*:

$$y_m = \varphi_{ij}, \quad m = i + (j-1)I \; . \tag{3.40}$$

Vice-versa, i and j follow from m as follows:

$$j = \text{floor}[(m-1)/I], \quad i = m - (j-1)I \; , \tag{3.41}$$

where floor(a/b) is the largest integer $\leq a/b$. The right-hand side b contains $\tilde{q}_{ij}$ in lexicographic order. The relation between the grid indices i, j and the lexicographic index m is illustrated in Fig. 3.5 With lexicographic ordering,

9	10	11	12		1,3	2,3	3,3	4,3
5	6	7	8		1,2	2,2	3,2	4,2
1	2	3	4		1,1	2,1	3,1	4,1

Fig. 3.5. Relation between lexicographic and grid indices

A is an $IJ \times IJ$ matrix with the following block-tridiagonal structure:

$$A = \begin{bmatrix} B_1 & D_1 & 0 & \cdots & 0 \\ C_2 & B_2 & D_2 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & C_{J-1} & B_{J-1} & D_{J-1} \\ 0 & \cdots & 0 & C_J & B_J \end{bmatrix} ,$$

where B_j are $I \times I$ tridiagonal matrices, given by

$$B_j = \begin{bmatrix} \alpha_{1j}^{0,0} & \alpha_{1j}^{1,0} & 0 & \cdots & 0 \\ \alpha_{2j}^{-1,0} & \alpha_{2j}^{0,0} & \alpha_{2j}^{1,0} & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \alpha_{I-1,j}^{-1,0} & \alpha_{I-1,j}^{0,0} & \alpha_{I-1,j}^{1,0} \\ 0 & \cdots & 0 & \alpha_{I,j}^{-1,0} & \alpha_{Ij}^{0,0} \end{bmatrix} ,$$

and C_j and D_j are $I \times I$ diagonal matrices, given by

$$C_j = \text{diag}\{\alpha_{ij}^{0,-1}\}, \quad D_j = \text{diag}\{\alpha_{ij}^{0,1}\} \; .$$

Remarks on the MATLAB program cd2

The numerical scheme described above has been implemented in the MATLAB program `cd2`, available at the author's website; see the Preface.

The student is not expected to fully understand this program, because use is made of somewhat advanced features, such as `meshgrid` and `reshape`, in order to avoid `for` loops. Avoiding `for` loops is essential for efficiency in MATLAB, as can be seen in the code `cd2` by comparing computing time (using `tic`$\cdots$`toc`) for generating the matrices $A1$ and $A3$. Generation of $A1$ is done in a simple way with `for` loops, and requires 1.22 time units on a 32×72 grid, whereas the more sophisticated program for $A3$ takes 0.067 time units; this discrepancy grows rapidly with increasing grid size. The time used for solving the system $Ay = b$ is 0.32.

In MATLAB, A is generated as a sparse matrix by

```
d = [-I;  -1;  0;  1;  I];
A = spdiags([a1 a2 -a1-a2-a4-a5 a4 a5], d, n, n);
```

with suitable definition of the diagonals `a1`$\cdots$`a5`. Exploiting sparsity is essential for saving memory and computing time. The solution of the system $Ay = b$ is obtained in MATLAB by

```
y = A\b;
```

This means that we solve with a direct method using sparse LU factorization. For large systems, that occur particularly when partial differential equations are solved in three-dimensional domains, direct methods frequently demand intolerable amounts of computer time and memory, even when sparsity is exploited. Efficient solution methods for solving the algebraic systems arising from numerical schemes for partial differential equations will be discussed in Chapt. 6.

Numerical experiments

The purpose of the numerical experiments with the program `cd2` that we will now describe is to demonstrate, as we did in Sect. 2.3 for the one-dimensional case, that we can achieve Péclet-uniform accuracy and efficiency, and that accurate results can be obtained on grids with large jumps in mesh size. This is shown theoretically in Sect. 4.7 of Wesseling (2001), but here we confine ourselves to numerical illustration.

In order to be able to assess the error, we choose an exact solution. Of course, this solution has to exhibit the boundary layer behavior occurring in practice. We choose the following solution of the boundary layer (inner) equation (3.26):

$$\varphi = \frac{1}{\sqrt{2-x}}\left\{\exp(-\frac{y^2}{4\varepsilon(2-x)}) + \exp(-\frac{(2-y)^2}{4\varepsilon(2-x)})\right\} .$$

The right-hand side and boundary conditions in (3.29) are chosen accordingly. The exact solution is symmetric with respect to $y = 1$, as assumed by the boundary conditions (3.29).

Because the solution is extremely smooth in Ω_c, it turns out that in G_c the number of cells in the vertical direction can be fixed at 4; the maximum of the error is found to always occur in G_f. We take

$$\sigma = 8\sqrt{\varepsilon}\ . \tag{3.42}$$

Table 3.1 gives results for the cell-centered upwind case. Exactly the same results (not shown) are obtained for $\varepsilon = 10^{-5}$ and $\varepsilon = 10^{-7}$, showing ε-uniform accuracy. Of course, computing time and memory are also independent of ε, because they depend only on nx and ny. The maximum error is found to occur in the interior of the boundary layer. Because we use the upwind scheme

nx	ny	error $* 10^4$
8	32	54
32	64	14
128	128	3.6

Table 3.1. Maximum error as function of number of grid-cells for $\varepsilon = 10^{-3}$; cell-centered upwind discretization. nx: horizontal number of cells; ny : vertical number of cells in G_f.

in the x-direction and the central scheme in the y-direction, we expect for the error $e = \mathcal{O}(H_1 + h_2^2)$, so that the error should decrease by a factor 4 at each refinement in Table 3.1; this expectation is confirmed. Table 3.2 gives results for central discretization of the convection term. Visual inspection of graphical output (not shown) shows no visible wiggles. But very small wiggles

nx	ny	error $* 10^4$
8	16	92
16	32	28
32	64	7.8
64	128	2.1

Table 3.2. Cell-centered central discretization; $\varepsilon = 10^{-3}$.

are present. These are the cause that the rate of convergence is somewhat worse than the hoped for $\mathcal{O}(H_1^2 + h_2^2)$, but here again the same results are obtained for $\varepsilon = 10^{-7}$, showing uniformity in ε.

We may conclude that in practice work and accuracy can be made to be uniform in ε, by suitable local mesh refinement according to Fig. 3.4 and equation

(3.42). Hence, in principle, high Reynolds number flows are amenable to computation.

As before, having come to the end of this chapter, looking again at the list of items that we wanted to cover given in Sect. 3.1 will help the reader to remind himself of what the main points were that we wanted to emphasize.

Exercise 3.3.1. Derive equations (3.32) and (3.33).

Exercise 3.3.2. Take u_1, u_2 and ε constant, and the grid uniform. Discretize the convection-diffusion equation (3.28) with hte finite volume method, using the central scheme. Show that the resulting stancil is

$$[L_h] = \varepsilon \begin{bmatrix} & \frac{1}{2}p_2 - \frac{h_1}{h_2} & \\ -\frac{1}{2}p_1 - \frac{h_2}{h_1} & 2(\frac{h_1}{h_2} + \frac{h_2}{h_1}) & \frac{1}{2}p_1 - \frac{h_2}{h_1} \\ & -\frac{1}{2}p_2 - \frac{h_1}{h_2} & \end{bmatrix},$$

where $p_1 \equiv u_1 h_1/\varepsilon$ and $p_2 \equiv u_2 h_2/\varepsilon$ are the signed (i.e., they can be positive or negative) mesh Péclet numbers.

Exercise 3.3.3. Run `cd2` with a homogeneous Neumann condition at the outflow boundary, and compare with the case in which the x-derivative at the outflow boundary is prescribed in accordance with the exact solution. This exercise illustrates once again that Pe $\gg$ 1 it is safe to prescribe a homogeneous Neumann condition at outflow.

Exercise 3.3.4. Run `cd2` with the central scheme for the convection term. Observe that this does not give better results than the upwind scheme if the mesh Péclet number is larger than 2. The cause is the occurrence of (very small) wiggles. How small should dx be to bring the mesh Péclet number p below 2? Would $p < 2$ make the computing work nonuniform in ε ?

Some self-test questions

What is your favorite outflow boundary condition? Why?

Define the subcharacteristics of the convection-diffusion equation.

What is the difference between a regular and a singular perturbation problem?
Formulate the matching principle.

What is the essential feature that makes it possible to have accuracy and efficiency Péclet-uniform? Why do we want this property?

When is a scheme of positive type? Why is this nice?

Under what conditions is the scheme with stencil of Exercise 3.3.2 of positive type?

4. The nonstationary convection-diffusion equation

4.1 Introduction

In the nonstationary case, time is included. The equation to be studied is the two-dimensional nonstationary convection-diffusion equation:

$$\begin{gathered}\frac{\partial\varphi}{\partial t}+\frac{\partial u\varphi}{\partial x}+\frac{\partial v\varphi}{\partial y}-\frac{\partial}{\partial x}\Big(\varepsilon\frac{\partial\varphi}{\partial x}\Big)-\frac{\partial}{\partial y}\Big(\varepsilon\frac{\partial\varphi}{\partial y}\Big)=q(t,x,y)\ , \\ 0<t\le T,\quad (x,y)\in\Omega\equiv(0,1)\times(0,1)\ .\end{gathered}\tag{4.1}$$

The following initial condition is required:

$$\varphi(0,x,y)=\varphi_0(x,y)\ .\tag{4.2}$$

Suitable boundary conditions are:

$$\begin{aligned}\varphi(t,x,y)&=f(t,x,y)\quad\text{on}\quad\partial\Omega_i\quad\text{(Dirichlet)}, &(4.3)\\ \varphi(t,x,y)&=f(t,x,y)\quad\text{on}\quad\partial\Omega_o\quad\text{(Dirichlet)}\quad\text{or} &(4.4)\\ \frac{\partial\varphi(t,x,y)}{\partial n}&=g(t,x,y)\quad\text{on}\quad\partial\Omega_o\quad\text{(Neumann)}, &(4.5)\end{aligned}$$

where $\mathbf{n}$ is the outward unit normal on the boundary $\partial\Omega$, $\partial\Omega_i$ is the inflow boundary (where $\mathbf{u}\cdot\mathbf{n}<0$) and $\partial\Omega_o$ is the remainder of $\partial\Omega$, to be called the outflow boundary.

When $\varepsilon=1/\mathrm{Pe}\ll 1$, boundary layers may occur at the same location and with the same thickness as in the stationary case, as may be seen by means of singular perturbation theory, which is easily extended to the nonstationary case.

As in the stationary case, we have a maximum principle. The non-conservative form of (4.1) is:

$$\frac{\partial\varphi}{\partial t}+u\frac{\partial\varphi}{\partial x}+v\frac{\partial\varphi}{\partial y}-\frac{\partial}{\partial x}\Big(\varepsilon\frac{\partial\varphi}{\partial x}\Big)-\frac{\partial}{\partial y}\Big(\varepsilon\frac{\partial\varphi}{\partial y}\Big)=\tilde{q}\equiv q-\varphi\,\mathrm{div}\,\mathbf{u}\ .\tag{4.6}$$

The maximum principle for the nonstationary case says that if $\tilde{q}\le 0$ and if there is a local maximum in the interior of Ω for $t=t^*>0$, then φ = constant, $0\le t\le t^*$. In effect this says that local maxima can occur only at $t=0$

and at the boundaries. If $\tilde{q} \geq 0$ the same applies to local minima, so that in the homogeneous case $\tilde{q} = 0$ there can be no local extrema (in space *and* time) in the interior; any numerical wiggles must be regarded as numerical artifacts. See Sect. 2.4 of Wesseling (2001) for a more precise version of the maximum principle. We will not prove this maximum principle here. But a simple physical analogy will convince the reader that it must be true. Think of a copper plate with a nonuniform temperature distribution at $t = 0$. The evolution of the temperature φ is governed by the heat equation, i.e. equation (4.6) with $u = v = 0$. If no heat sources or sinks are present (i.e. $\tilde{q} = 0$) then the temperature distribution evolves to a uniform state, and a local hot spot must have been hotter at earlier times.

Purpose of this chapter

The purpose of this chapter is:

- To introduce methods for discretization in time;
- To explain the concepts of consistency, stability and convergence;
- To show that numerical schemes must be stable;
- To show how stability conditions can be derived by Fourier analysis.

4.2 A numerical example

Consider the one-dimensional heat equation, which is a special case of equation (4.1):

$$\frac{\partial \varphi}{\partial t} - \varepsilon \frac{\partial^2 \varphi}{\partial x^2} = 0 , \quad 0 < t \leq T , \quad x \in \Omega \equiv (0,1) , \tag{4.7}$$

with initial condition and homogeneous Neumann boundary conditions given by

$$\varphi(0,x) = \varphi_0(x) , \quad \frac{\partial \varphi(t,0)}{\partial x} = \frac{\partial \varphi(t,1)}{\partial x} = 0 . \tag{4.8}$$

This is a mathematical model for the evolution of the temperature φ in a thin insulated bar with initial temperature distribution $\varphi_0(x)$. We have

$$\lim_{t \to \infty} \varphi(t,x) = \text{constant} = \int_0^1 \varphi_0(x) dx . \tag{4.9}$$

Discretization in space

For discretization in space we use the finite volume method on the vertex-centered grid of Fig. 2.1 with uniform mesh size h. The following details have been covered in the preceding chapters, but we present them as an exercise. Integration over the control volumes gives, with ε constant:

$$\begin{aligned}
&\frac{h}{2}\frac{d\varphi_1}{dt} + F_{1/2} - F_{3/2} = 0 \ , \\
&h\frac{d\varphi_j}{dt} + F_{j-1/2} - F_{j+1/2} = 0 \ , \quad j = 2, \cdots , J-1 \ , \\
&\frac{h}{2}\frac{d\varphi_J}{dt} + F_{J-1/2} - F_{J+1/2} = 0 \ ,
\end{aligned} \tag{4.10}$$

where F is an approximation of $\varepsilon\partial\varphi/\partial x$, naturally chosen as follows:

$$F_{j+1/2} = \varepsilon(\varphi_{j+1} - \varphi_j)/h \ , \quad j = 1, \cdots , J-1 \ .$$

From the boundary conditions it follows that $F_{1/2} = F_{J+1/2} = 0$.

Discretization in time

Equation (4.10) is rewritten as

$$\frac{d\varphi_j}{dt} + L_h\varphi_j = 0 \ , \quad j = 1, \cdots , J \ . \tag{4.11}$$

For discretization in time we choose the *forward Euler scheme*:

$$(\varphi_j^{n+1} - \varphi_j^n)/\tau + L_h\varphi_j^n = 0 \ , \quad n = 0, \cdots , N \ , \quad N \equiv T/\tau \ , \tag{4.12}$$

with τ the time step, taken constant; φ_j^n is the numerical approximation of $\varphi(n\tau, x_j)$ with $\varphi(t, x)$ the exact solution. From Fig. 2.1 it follows that $x_j = (j-1)h, \ \ h = 1/(J-1)$. Equation (4.12) is equivalent to

$$\begin{aligned}
\varphi_1^{n+1} &= (1-2d)\varphi_1^n + 2d\varphi_2^n \ , \\
\varphi_j^{n+1} &= d\varphi_{j-1}^n + (1-2d)\varphi_j^n + d\varphi_{j+1}^n \ , \quad j = 2, \cdots , J-1 \ , \\
\varphi_J^{n+1} &= 2d\varphi_{J-1}^n + (1-2d)\varphi_J^n \ ,
\end{aligned} \tag{4.13}$$

where $d \equiv \varepsilon\tau/h^2$ is a dimensionless number, that we will call the *diffusion number.*

The matrix of the scheme

If we define $\boldsymbol{\varphi} \equiv (\varphi_1, \cdots , \varphi_J)^T$ then (4.13) can be rewritten as

$$\boldsymbol{\varphi}^{n+1} = A\boldsymbol{\varphi}^{n} \ ,$$

with A the following tridiagonal matrix:

$$A = \begin{bmatrix} 1-2d & 2d & 0 & \cdots & 0 \\ d & 1-2d & d & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & d & 1-2d & d \\ 0 & \cdots & 0 & 2d & 1-2d \end{bmatrix} . \tag{4.14}$$

This matrix is easily generated as a sparse matrix in MATLAB by the following statements:

```
e = ones(J,1);
de = d*e;
A = spdiags([de, e - 2*de, de], -1:1, J, J);
A(1,2) = 2*d;
A(J,J-1) = 2*d;
```

Numerical results

The following numerical results have been obtained with the MATLAB code `heq` . As initial solution we choose

$$\begin{aligned} \varphi_0(x) &= 0 \ , \quad 0 < x < 0.4 \ , \\ \varphi_0(x) &= 1 \ , \quad 0.4 < x < 0.6 \ , \\ \varphi_0(x) &= 0 \ , \quad 0.6 < x < 1 \ . \end{aligned} \tag{4.15}$$

Numerical results are shown in Fig. 4.1 for two values of the diffusion number d. The result at the left looks as expected: the temperature distribution is tending to uniformity, and the maximum principle is satisfied: no new maxima in space, and the maximum at $x = 1/2$ was larger at earlier times. But the result in the right part of the figure is wrong. Clearly, the value of the diffusion number has a crucial influence, and something strange happens between $d = 0.48$ and $d = 0.52$! This will be investigated in what follows.

Efficiency in MATLAB

But we first use the MATLAB implementation of the initial condition (4.15) as a nice example to illustrate the impact of vectorization on efficiency. Let `vo` be preallocated with `vo = zeros(J,1);` A non-vectorized implementation of (4.15) is:

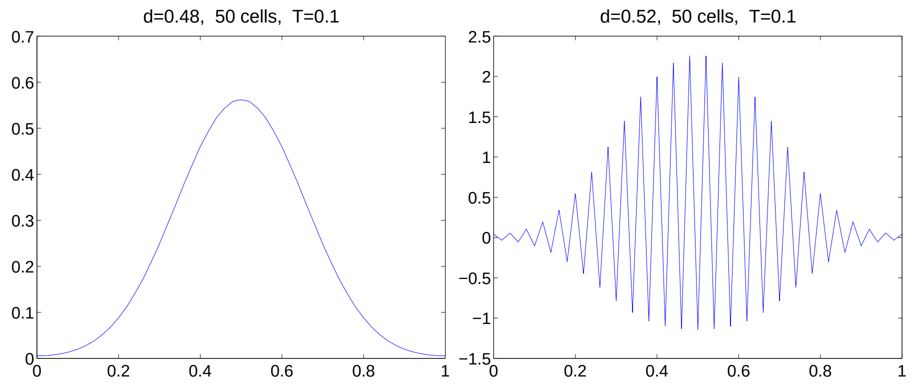

Fig. 4.1. Numerical solution of heat equation for two values of d.

```
for j = 1:J
  if (x(j) > 0.4) & (x(j) < 0.6), vo(j) = 1; end
end
```

A more efficient non-vectorized implementation is:

```
j1 = floor(1 + (J-1)*0.4); j2 = ceil(1 + (J-1)*0.6);
for j = j1:j2
  vo(j) = 1;
end
```

A vectorized implementation is:

```
j = floor(1 + (J-1)*0.4):ceil(1 + (J-1)*0.6);
vo(j) = 1;
```

A more efficient vectorized implementation is:

```
vo(floor(1 + (J-1)*x1):ceil(1 + (J-1)*x2)) = 1;
```

For these four versions, `tic ... toc` gives the following timings, respectively, for $J = 100$, on my Pentium II processor:

$$\{19.0 \quad 3.0 \quad 1.5 \quad 1.3\} * 10^{-4}$$

The message is:
Avoid `for` *loops.* If you cannot avoid them, *avoid* `if` *statements within* `for` *loops.*

Exercise 4.2.1. Prove equation (4.9).
Hints: Show that if $\frac{\partial \varphi}{\partial t} = 0$ the solution is constant. Show that $\frac{d}{dt}\int_0^1 \varphi dx = 0$. (This shows that with these boundary conditions the bar is insulated: the heat content is conserved).

4.3 Convergence, consistency and stability

One-step schemes

Schemes in which only two time levels are involved are called *one-step schemes*; the forward Euler method (4.12) is an example. The general form of linear one-step schemes is:

$$B_1\boldsymbol{\varphi}^{n+1} = B_0\boldsymbol{\varphi}^n + B_2\mathbf{q}^n + B_3\mathbf{q}^{n+1} \;, \tag{4.16}$$

where $B_0, \cdots, B_3$ are linear operators, and where $\mathbf{q}$ arises from the right-hand side of the differential equation and the boundary conditions. For simplicity we restrict ourselves here to one-step schemes.

Local and global truncation error

Let us denote the algebraic vector with elements the exact solution evaluated at the grid point and at time t_n by $\boldsymbol{\varphi}_e^n$. Let us define the *local truncation error* τ_j^n, gathered in an algebraic vector $\boldsymbol{\tau^n}$ (not to be confused with the time step τ) by the following equation:

$$B_1\boldsymbol{\varphi}_e^{n+1} = B_0\boldsymbol{\varphi}_e^n + B_2\mathbf{q}^n + B_3\mathbf{q}^{n+1} + \boldsymbol{\tau}^n \;. \tag{4.17}$$

The *global truncation error* is defined as

$$\mathbf{e}^n \equiv \boldsymbol{\varphi}_e^n - \boldsymbol{\varphi}^n \;. \tag{4.18}$$

By subtracting (4.17) and (4.16) we get the following relation between the global and the local truncation error:

$$B_1\mathbf{e}^{n+1} = B_0\mathbf{e}^n + \boldsymbol{\tau}^n \;. \tag{4.19}$$

This is very similar to equation (2.47), so that the above definitions are consistent with the truncation error definitions for the stationary case given in Sect. 2.3.

Convergence

Of course, we want $\|\mathbf{e}^n\| \downarrow 0, \;\; n = 1, \cdots, T/\tau$ as the grid is refined and $\tau \downarrow 0$ (while keeping T/τ integer, obviously). This is not the case for our previous numerical example when $d > 1/2$. No matter how small h and τ are chosen, the numerical solution will always look like the right part of Fig. 4.1, since it depends only on the parameter d, *cf.* (4.13). Clearly, we have to do some analysis to find conditions that guarantee that a numerical scheme is good.

Let the number of space dimensions be m and let us have an m-dimensional spatial grid G with grid points

$$\mathbf{x}_j = (x^1_{j_1}, \cdots , x^m_{j_m}) \ .$$

Hence, now j is a multi-index $(j_1, \cdots , j_m)$. Let the spatial mesh sizes and the time step be decreasing functions of a parameter h, that belongs to a sequence that decreases to zero. To emphasize the dependence of G on h we write G_h. We want at a fixed time T and a fixed location $\mathbf{x}$ the error to tend to 0 as $h \downarrow 0$. This implies that the number of time steps $n = T/\tau$ changes, and the multi-index j changes to keep $\mathbf{x}_j$ fixed. To emphasize this we write j_h. Of course, we assume that G_h is such that the fixed point $\mathbf{x}_{j_h}$ remains a grid point when the grid is refined. For example, in the preceding numerical example we could choose

$$\mathbf{x}_{j_h} = 1/2 \ , \quad h \in \{1/2,\ 1/4,\ 1/6, \cdots\}.$$

so that $j_h \in \{1,\ 2,\ 3, \cdots\}$. We are now ready for the definition of *convergence*.

Definition 4.3.1. *Convergence*
A scheme is called *convergent* if the global truncation error satisfies

$$\lim_{h\downarrow 0} e^{T/\tau}_{j_h} = 0, \quad \boldsymbol{x}_{j_h} \text{ fixed.}$$

Clearly, scheme (4.13) is not convergent for $d > 1/2$.

Consistency

It seems likely that for convergence it is necessary that the local truncation error is small enough. We therefore formulate a condition. Let the scheme (4.16) in a given grid point $\mathbf{x}_j$ approximate the differential equation times $\tau^\alpha |\partial \Omega_j|^\beta$ with $|\partial \Omega_j|$ the volume of some cell around $\boldsymbol{x}_j$ (For example, the scheme (4.13) obviously approximates equation (4.7) times τ, so that in this case $\alpha = 1,\ \beta = 0$). We define

Definition 4.3.2. *Consistency*
The scheme (4.16) is called *consistent* if

$$\lim_{h\downarrow 0} \tau^n_{j_h} / (\tau^\alpha |\partial \Omega_{j_h}|^\beta) = 0, \quad j \in G_h, \quad 1 \le n \le T/\tau \ .$$

In Exercise 4.3.1 the reader is asked to show that scheme (2.48) is *not* consistent on rough grids. This seems disturbing, because we are going to show that consistency is necessary for convergence, but in Sect. 2.3 it was shown that scheme (2.48) converges on rough grids. This apparent paradox arises from the fact that Def. 4.3.2 implies that the local truncation error is measured in the maximum norm. On rough grids we have consistency of scheme (2.48) in a more sophisticated norm, but not in the maximum norm; we will not go into this further. In this chapter only uniform grids are considered; on these grids the various appropriate norms are the same.

Stability

As illustrated by the numerical example in the preceding section, consistency does not imply convergence. In addition, *stability* is required. This concept will now be explained. Let $\boldsymbol{\delta}^0$ be a hypothetical arbitrary perturbation of $\boldsymbol{\varphi}^0$. The resulting perturbation of $\boldsymbol{\varphi}^n$ is called $\boldsymbol{\delta}^n$. It is left for the reader to derive from equation (4.16) that

$$B_1\boldsymbol{\delta}^{n+1} = B_0\boldsymbol{\delta}^n \ . \tag{4.20}$$

Let $\|\cdot\|_h$ be some norm for functions $G_h \to \mathbb{R}$. Stability means that $\boldsymbol{\delta}^n$ remains bounded as $n \to \infty$, for all $\boldsymbol{\delta}^0$. Two useful definitions are:

Definition 4.3.3. *Zero-stability*
A scheme is called *zero-stable* if there exists a bounded function $C(T)$ and a function $\tau_0(h)$ such that for arbitrary $\boldsymbol{\delta}^0$

$$\|\boldsymbol{\delta}^{T/\tau}\|_h \le C(T)\|\boldsymbol{\delta}^0\|_h \tag{4.21}$$

for all $\tau \le \tau_0(h)$ and all $h \le h_0$ for some fixed h_0.

The appellation "zero-stability" refers to the fact that the limit $h \downarrow 0$ is considered.

Definition 4.3.4. *Absolute stability*
A scheme is called *absolutely stable* if there exists a constant C and a function $\tau_0(h)$ such that for arbitrary $\boldsymbol{\delta}^0$

$$\|\boldsymbol{\delta}^n\|_h \le C\|\boldsymbol{\delta}^0\|_h \tag{4.22}$$

for h fixed, all $n > 0$ and all $\tau \le \tau_0(h)$.

The difference with zero-stability is that here h is fixed.

Lax's equivalence theorem

Definition 4.3.3 considers the perturbation at a fixed time T as $h \downarrow 0$, which is the same limit as in the definition of convergence. It can be shown that convergence implies zero-stability, and zero-stability plus consistency imply convergence. This is known as *Lax's equivalence theorem.* As a consequence, *zero-stability is necessary for convergence.* But absolute stability is also good to have, because it allows n to grow indefinitely, making it possible to continue time stepping until a steady state is reached. Absolute and zero-stability are not completely equivalent.

A remark on stability analysis

The purpose of stability analysis is to find a suitable function $\tau_0(h)$ such that (4.21) and (4.22) hold. This is the case if the linear operators in (4.20) satisfy

$$\|(B_1^{-1}B_0)^n\|_h \leq C \ .$$

In the case of absolute stability, h and hence the dimensions of the matrices B_0 and B_1 are fixed, and linear algebra can be used to find conditions under which $\|(B_1^{-1}B_0)^n\|$ is bounded as $n \to \infty$. But in the case of zero-stability, $n \to \infty$ and $h \downarrow 0$ simultaneously, and we have to study not just the behavior of the nth power of a matrix, but of a family of matrices of increasing size. This is not a familiar situation in linear algebra. We will see that Fourier analysis is well-suited to the study of both kinds of stability, if the boundary conditions are periodic.

Exercise 4.3.1. Show that for scheme(2.48) we have $\beta = 1$. Because this scheme is for the stationary case, time can be disregarded. Show that, using (2.52), scheme (2.48) is consistent on smooth grids (as defined below equation (2.52)), but *not* on rough grids.

4.4 Fourier stability analysis

Applicability of Fourier analysis

In general it is difficult to derive estimates like (4.21) and (4.22). But if the coefficients in the scheme are constant, the mesh uniform, the grid a rectangular block and the boundary conditions periodic, then Fourier analysis applies and the required estimates are often not difficult to obtain.

In practice, of course, the coefficients are usually not constant. The scheme is called *locally stable* if we have stability for the constant coefficients scheme that results from taking local values of the coefficients, and to assign these values to the coefficients in the whole domain (frozen coefficients method). Local stability in the whole domain is necessary for stability in the variable coefficients case. We will discuss stability only for constant coefficients.

Stability theory for non-periodic boundary conditions is complicated. But for explicit time stepping schemes it takes a while before the influence of the boundary conditions makes itself felt in the interior, so that Fourier stability theory applies during a certain initial time span. As a consequence, stability with periodic boundary conditions is desirable, even if the boundary conditions are of different type.

Example of frozen coefficients method

Consider the Burgers equation:

$$\frac{\partial \varphi}{\partial t} + \frac{1}{2}\frac{\partial \varphi^2}{\partial x} = 0\,.$$

Discretization with the forward Euler method in time and the upwind scheme in space gives:

$$\varphi_j^{n+1} - \varphi_j^n + \frac{\tau}{2h}[(\varphi_j^n)^2 - (\varphi_{j-1}^n)^2] = 0\,,$$

assuming $\varphi > 0$ and a uniform grid. For stability analysis we postulate a perturbation $\delta\varphi^0$ of the initial solution. The perturbed solution satisfies

$$(\varphi + \delta\varphi)_j^{n+1} - (\varphi + \delta\varphi)_j^n + \frac{\tau}{2h}\{[(\varphi + \delta\varphi)_j^n]^2 - [(\varphi + \delta\varphi)_{j-1}^n]^2\} = 0\,.$$

Subtraction of the preceding two equations and linearization (*i.e.* deletion of terms quadratic in $\delta\varphi$) gives:

$$\delta\varphi_j^{n+1} - \delta\varphi_j^n + \frac{\tau}{h}[(\varphi\delta\varphi)_j^n - (\varphi\delta\varphi)_{j-1}^n] = 0\,.$$

Freezing of the coefficients results in

$$\delta\varphi_j^{n+1} - \delta\varphi_j^n + \frac{c\tau}{h}(\delta\varphi_j^n - \delta\varphi_{j-1}^n) = 0\,,$$

where c is the frozen value of φ. This scheme allows Fourier stability analysis. Usually, a stability condition of the type

$$\frac{c\tau}{h} < C \tag{4.23}$$

results for some value of C. This condition is to be satisfied for the frozen coefficient c equal to all values that the variable coefficient φ_j^n takes. We see from (4.23) that it suffices to take $c = \max(|\varphi_j^n|)$. Frequently an informed guess for $\max(|\varphi_j^n|)$ can be made by looking at the boundary conditions.

In the remainder of this section we present the basic principles of Fourier stability analysis.

Fourier series

Let G_h be a uniform grid on the unit interval with nodes

$$x_j = jh\,, \quad j = 0, 1, \cdots, J-1 \equiv 1/h\;.$$

It can be shown that every grid function $\delta:\ G_h \to \mathbb{R}$ can be represented by what is called a *Fourier series*:

$$\delta_j = \sum_{-m}^{m+p} c_k e^{ij2\pi k/J}\,, \quad j = 0, 1, \cdots, J-1\,,$$

where $p = 0,\ m = (J-1)/2$ for J odd and $p = 1,\ m = (J-2)/2$ for J even. The proof is elementary, and can be found for instance in Chap. 7 of Wesseling (1992). It is convenient to rewrite this as

$$\delta_j = \sum_{\theta\in\Theta} c_\theta e^{ij\theta}\,, \quad \Theta \equiv \{\theta = 2\pi k/J\,,\ k = -m, -m+1, \cdots, m+p\}\;. \quad (4.24)$$

We note that $e^{ij\theta} = \cos j\theta + i \sin j\theta$ is complex, so that c_θ is also complex, such that δ_j is real. We can regard c_θ as the amplitude of the harmonic wave $e^{ij\theta}$. In (4.24) θ ranges approximately between $-\pi$ and π if $J \gg 1$. For $\theta = \pi$ we have the shortest wave that can be resolved on $G_h:\ \ e^{ij\pi} = (-1)^j$, and for $\theta = 0$ the wavelength is infinite: $e^0 = 1$. Note that $\boldsymbol{\delta}$ is *periodic*: $\delta_j = \delta_{j+J}$. The functions $e^{ij\theta},\ \theta \in \Theta$ are called *Fourier modes*. The parameter θ is called the *wavenumber*. The function $f(x) \equiv e^{ij\theta}$ has period or wavelength $2\pi/\theta$, since $f(x + 2\pi/\theta) = f(x)$.

The Fourier series (4.24) is easily extended to more dimensions. We restrict ourselves to the two-dimensional case. Let G_h be a uniform grid on the unit square with nodes

$$\boldsymbol{x}_j = (j_1 h_1,\ j_2 h_2)\,, \quad j_\alpha = 0, \cdots, J_\alpha - 1 \equiv 1/h_\alpha\,, \quad \alpha = 1, 2\;.$$

Define the set Θ of wavenumbers as

$$\begin{aligned}\Theta \equiv \{&\theta = (\theta_1,\ \theta_2): \quad \theta_\alpha = 2\pi k_\alpha/J_\alpha\,,\\ &k_\alpha = -m_\alpha, -m_\alpha+1, \cdots, m_\alpha + p_\alpha\,, \quad \alpha = 1, 2\}\,,\end{aligned}$$

where $p_\alpha = 0,\ m_\alpha = (n_\alpha - 1)/2$ for J_α odd and $p_\alpha = 1,\ m_\alpha = n_\alpha/2 - 1$ for J_α even. Define

$$j\theta = \sum_{\alpha=1}^{2} j_\alpha \theta_\alpha\;. \quad (4.25)$$

It can be shown that every grid function $\delta : G_h \to \mathbb{R}$ can be written as

$$\delta_j = \sum_{\theta\in\Theta} c_\theta e^{ij\theta}\;. \quad (4.26)$$

with the amplitudes c_θ given by

$$c_\theta = N^{-1} \sum_{j \in G_h} \delta_j e^{-ij\theta} , \quad N = J_1 J_2 .$$

As a consequence we have

$$\sum_{\theta \in \Theta} c_\theta e^{ij\theta} = 0, \quad \forall j \in G_h \qquad \Rightarrow \qquad c_\theta = 0, \quad \forall \theta \in \Theta . \tag{4.27}$$

Let us define the l_2-norm by

$$\|\boldsymbol{\delta}\| = N^{-\frac{1}{2}} \{ \sum_{j \in G_h} (\delta_j)^2 \}^{1/2}, \quad \|\boldsymbol{c}\| = N^{-\frac{1}{2}} \{ \sum_{\theta \in \Theta} |c_\theta|^2 \}^{1/2}$$

We will need

Theorem 4.4.1. *Parseval.*
If $\boldsymbol{\delta}$ and $\boldsymbol{c}$ are related by (4.26), then in the l_2-norm

$$\|\boldsymbol{\delta}\| = N^{1/2} \|\boldsymbol{c}\| .$$

An example

Consider the example of Sect. 4.2. The perturbation $\boldsymbol{\delta}^n$ in the numerical solution $\boldsymbol{\varphi}^n$ satisfies the same equation as $\boldsymbol{\varphi}^n$, i.e. equation (4.13) (show this!). But now we assume periodic boundary conditions (to make Fourier analysis applicable), so that the boundary conditions are replaced by the condition $\delta_j = \delta_{j+J}$. We have

$$\delta_j^{n+1} = d\delta_{j-1}^n + (1-2d)\delta_j^n + d\delta_{j+1}^n .$$

Substitution of (4.24) gives:

$$\sum_{\theta \in \Theta} e^{ij\theta} \left[c_\theta^{n+1} - c_\theta^n (de^{-i\theta} + 1 - 2d + de^{i\theta}) \right] = 0 .$$

Because ε and h are constant, d is constant; therefore the term between [ ] does not depend on j; if this were not the case the following step could not be taken and Fourier analysis falls through. But since the term between [ ] does not depend on j, it follows from (4.27) that the term between [ ] is zero, and we get rid of the sum in the preceding equation. Using $e^{-i\theta} + e^{i\theta} = 2\cos\theta$ we get

$$c_\theta^{n+1} = g(\theta) c_\theta^n , \quad g(\theta) \equiv 1 - 2d(1 - \cos\theta) , \tag{4.28}$$

where $g(\theta)$ is called the *amplification factor*: it measures the amplification (or damping) of the amplitude c_θ of the Fourier mode $e^{ij\theta}$. Before continuing with this example, we go to the general case.

The general case

Whenever Fourier stability analysis is applicable, we get relation (4.28) with some function $g(\theta)$ (that is complex in general). It follows that

$$c_\theta^n = g(\theta)^n c_\theta^0 \ ,$$

so that

$$\|\boldsymbol{c}^n\| \leq \bar{g}^n \|\boldsymbol{c}^0\| \ , \quad \bar{g} = \max\{|g(\theta)| : \ \theta \in \Theta\} \ , \tag{4.29}$$

with equality for the θ for which the maximum is attained. According to Parseval's theorem (Theorem 4.4.1) we have $\|\boldsymbol{c}\| = N^{-1/2}\|\boldsymbol{\delta}\|$, so that the preceding equation gives

$$\|\boldsymbol{\delta}^n\| \leq \bar{g}^n \|\boldsymbol{\delta}^0\| \ . \tag{4.30}$$

From Definition 4.3.4 it follows that for absolute stability we must have

$$\bar{g}^n < C, \quad \forall n$$

for some C; hence

$$\bar{g} < 1. \tag{4.31}$$

This is sufficient, but also necessary, because equality can occur in (4.29), since all Fourier modes can be present, because $\boldsymbol{\delta}^0$ is arbitrary (cf. Def. 4.3.4).

A sufficient condition for zero-stability is that there exists a constant C such that

$$\bar{g}^{(T/\tau)} \leq C \tag{4.32}$$

for $0 \leq \tau \leq \tau_0(h)$. Since

$$C^{\tau/T} = \exp(\frac{\tau}{T} \ln C) = 1 + \mathcal{O}(\tau) \ , \tag{4.33}$$

we may write

$$\bar{g} \leq 1 + \mathcal{O}(\tau) \ . \tag{4.34}$$

This is the *von Neumann condition* for zero-stability (after John von Neumann, who introduced the Fourier method for stability analysis around 1944 in Los Alamos; not to be confused with the nineteenth century mathematician Neumann of the boundary condition). The von Neumann condition is also necessary, because if $\bar{g} \geq 1 + \mu, \ \mu > 0$, then there is a θ with $|g(\theta)| = 1 + \mu$. Choosing $\delta_j^0 = e^{ij\theta}$ gives $\|\boldsymbol{\delta}^n\|/\|\boldsymbol{\delta}\| = (1+\mu)^n, \ n = T/\tau$, which is unbounded as $\tau \downarrow 0$. Note that the $\mathcal{O}(\tau)$ term (4.34) is not allowed to depend on h; this follows from (4.32)—(4.34).

We will neglect the $\mathcal{O}(\tau)$ term in (4.34), because this makes hardly any difference. For simplicity, we will also extend the set of wavenumbers Θ to

$$\Theta = (-\pi, \ \pi] \quad \text{(one dimension)}, \quad \Theta = (-\pi, \ \pi] \times (-\pi, \ \pi] \quad \text{(two dimensions)}.$$

This also makes hardly any difference, since in practice $J_\alpha \gg 1$. We end up with the following condition for absolute and zero-stability:

$$\bar{g} \leq 1\,, \quad \bar{g} \equiv \max\{|g(\theta)|:\ \forall \theta\}. \tag{4.35}$$

(Note that $g(\theta)$ is complex in general, so that $|g|$ is the modulus of a complex number).

An example, continued

In our example $g(\theta)$ is given by (4.28), so that $g(\theta)$ happens to be real, and equation (4.35) gives:

$$-1 \leq 1 - 2d(1-\cos\theta) \leq 1\,, \quad \forall \theta\ .$$

Th right inequality is always satisfied, since $d \geq 0$. The left inequality gives

$$d(1-\cos\theta) \leq 1\,, \quad \forall\theta\ ,$$

so that we must have

$$d \leq 1/2\ .$$

This beautifully explains the numerical results obtained in Fig. 4.1. It follows that the time step must satisfy

$$\tau \leq \frac{h^2}{2\varepsilon}\ . \tag{4.36}$$

Here we encounter an example of the function $\tau_0(h)$ in the definition of zero-stability:

$$\tau_0(h) = \frac{h^2}{2\varepsilon}\ .$$

Condition (4.36) is rather restrictive, because τ must be reduced much more than h, when h is decreased for better accuracy. It may be inefficient to use such small time steps. Therefore we will consider another scheme below. But first we give another example.

Second example

Consider the one-dimensional convection equation:

$$\frac{\partial\varphi}{\partial t} + u\frac{\partial\varphi}{\partial x} = 0. \tag{4.37}$$

We discretize in space with the upwind scheme and in time with the forward Euler scheme, and obtain (assuming $u > 0$):

$$\varphi_j^{n+1} = \varphi_j^n - c(\varphi_j^n - \varphi_{j-1}^n) , \quad c \equiv u\tau/h . \tag{4.38}$$

The dimensionless number c is called the *Courant-Friedrichs-Lewy number* or *CFL number*, after the authors of the 1928 paper in which the importance of numerical stability was first brought to light; c is also called the *Courant number*. The quick way to get the amplification factor is to take just one Fourier mode, and to substitute $\varphi_j^n = c_\theta^n e^{ij\theta}$; the amplification factor is $g(\theta) = c_\theta^{n+1}/c_\theta^n$. This gives

$$g(\theta) = 1 - c(1 - e^{-i\theta}) . \tag{4.39}$$

To study stability we must consider $|g(\theta)|$, which, however, takes a somewhat unpleasant form, so that we prefer the following more elegant geometrical approach. We note that the complex number $g(\theta)$, when θ varies between$-\pi$ and π, traces out a circle with center at $1 - c$ and radius c. From Fig. 4.4 it is clear that for $|g(\theta)|$ not to leave the unit circle we must have $0 \leq c \leq 1$,

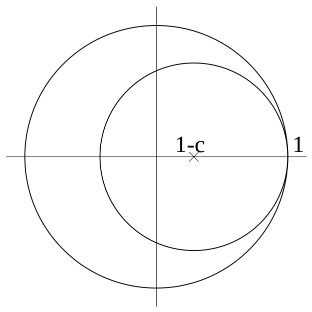

Fig. 4.2. Locus of $g(\theta)$ in complex plane.

resulting in the following stability condition on the time step:

$$\tau \leq h/u , \tag{4.40}$$

which provides another example of the function $\tau_0(h)$ in Def. 4.3.3.

The $\boldsymbol{\omega}$-scheme

The *backward Euler scheme* for equation (4.11) is given by

$$(\varphi_j^{n+1} - \varphi_j^n)/\tau + L_h \varphi_j^{n+1} = 0 . \tag{4.41}$$

Now we have to solve a system of equations for $\boldsymbol{\varphi}^{n+1}$, which is why this is called an *implicit* scheme; the forward Euler scheme (4.12) is of *explicit* type. Therefore a time step with an implicit scheme requires much more computing work than an explicit scheme, but this may be compensated by better stability properties, allowing a larger time step. The global truncation error of the Euler time stepping schemes (4.12) and (4.41) satisfies

$$\mathbf{e} = \mathcal{O}(\tau + h^m) \ ,$$

where m depends on the accuracy of the spatial discretization L_h. We can improve this by taking a linear combination of the forward and backward Euler schemes, as follows:

$$(\varphi_{jk}^{n+1} - \varphi_{jk}^n)/\tau + (1-\omega)L_h\varphi_{jk}^n + \omega L_h \varphi_{jk}^{n+1} = \omega q_{jk}^{n+1} + (1-\omega)q_{jk}^n \ , \quad (4.42)$$

where we now assume the right-hand side q in (4.1) to be nonzero, and where we have gone to the two-dimensional case. This is called the ω-scheme. For $\omega = 1/2$ this is called the *Crank-Nicolson scheme*, which is second order in time:

$$\mathbf{e} = \mathcal{O}(\tau^2 + h^m) \ .$$

With the central scheme for convection, the space discretization of equation (4.1) is, with u, v, ε constant and uniform mesh sizes h_1 and h_2 in the $x-$ and $y-$directions, respectively,:

$$\begin{aligned} L_h\varphi_{jk} =& \frac{u}{2h_1}(\varphi_{j+1,k} - \varphi_{j-1,k}) + \frac{v}{2h_2}(\varphi_{j,k+1} - \varphi_{j,k-1}) \\ &+ \frac{\varepsilon}{h_1^2}(-\varphi_{j-1,k} + 2\varphi_{jk} - \varphi_{j+1,k}) + \frac{\varepsilon}{h_2^2}(-\varphi_{j,k-1} + 2\varphi_{jk} - \varphi_{j,k+1}) \ . \end{aligned} \quad (4.43)$$

The equation for the perturbation δ_{jk}^n is identical to the homogeneous version (*i.e.* $q = 0$) of equation (4.43). For $1/2 < \omega < 1$ the stability analysis of the ω-scheme is easy. As said before, the quick way to determine the amplification factor is to substitute $\delta_j^n = c_\theta^n e^{ij\theta}$, where now j stands for $\{j, k\}$, and where $j\theta = j\theta_1 + k\theta_2$, *cf.* equation (4.25). Substitution gives

$$c_\theta^{n+1} - c_\theta^n + (1-\omega)\tau\hat{L}_h(\theta)c_\theta^n + \omega\tau\hat{L}_h(\theta)c_\theta^{n+1} = 0 \ ,$$

where

$$\hat{L}_h(\theta) \equiv e^{-ij\theta}L_h e^{ij\theta} \ .$$

It follows that

$$g(\theta) = \frac{c_\theta^{n+1}}{c_\theta^n} = \frac{1 - (1-\omega)\tau\hat{L}_h(\theta)}{1 + \omega\tau\hat{L}_h(\theta)} \ .$$

Let the real and imaginary part of the complex variable $\tau\hat{L}_h$ be w_1 and w_2, respectively: $\tau\hat{L}_h = w_1 + iw_2$. Noting that for two complex numbers $z_{1,2}$ we have $|z_1/z_2| = |z_1|/|z_2|$ and noting that $|\hat{L}_h|^2 = w_1^2 + w_2^2$, we get

$$|g(\theta)|^2 \equiv \frac{[1-(1-\omega)w_1]^2 + (1-\omega)^2 w_2^2}{(1+\omega w_1)^2 + \omega^2 w_2^2} \, .$$

Assume that $w_1 \geq 0$. Then for $1/2 \leq \omega \leq 1$ the denominator is not smaller than the numerator, so that $|g(\theta)| \leq 1$, and we have stability. To check whether $w_1 \geq 0$, we have to determine $\tau \hat{L}_h$. We find that

$$\tau \hat{L}_h = i(c_1 \sin\theta_1 + c_2 \sin\theta_2) + 2d_1(1-\cos\theta_1) + 2d_2(1-\cos\theta_2) \, , \tag{4.44}$$

where

$$c_1 \equiv \frac{u\tau}{h_1} \, , \; c_2 \equiv \frac{v\tau}{h_2} \, , \; d_1 \equiv \frac{\varepsilon\tau}{h_1^2} \, , \; d_2 \equiv \frac{\varepsilon\tau}{h_2^2} \, ,$$

from which it is obvious that $w_1 = \mathrm{Re}(\tau\hat{L}_h) \geq 0, \ \forall\theta$. Similarly, with the upwind scheme for convection we obtain, assuming $u, v \geq 0$,

$$\tau \hat{L}_h = c_1(1-e^{-i\theta_1}) + c_2(1-e^{-i\theta_2}) + 2d_1(1-\cos\theta_1) + 2d_2(1-\cos\theta_2) \, , \tag{4.45}$$

We have $w_1 = (c_1 + 2d_1)(1-\cos\theta_1) + (c_2 + 2d_2)(1-\cos\theta_2) \geq 0 \, , \ \forall\theta$, since $c_{1,2} \geq 0$.

Hence, we have established unconditional stability of the ω-scheme for the convection-diffusion equation, for $1/2 \leq \omega \leq 1$. The only interesting values of ω are $0, \ 1/2 + \mathcal{O}(\tau), \ 1$. The value $\omega = 0$ is of interest because this gives an explicit scheme, for which a time step is cheap. A value $\omega = 1/2 + \mathcal{O}(\tau)$ is of interest because this gives $\mathcal{O}(\tau^2)$ accuracy. Finally, $\omega = 1$ is of interest because this is necessary for the discrete maximum principle in the nonstationary case; we will not go into this. Therefore we will not give stability conditions for the ω-scheme for $0 \leq \omega < 1/2$, but only for $\omega = 0$. In this case the analysis becomes a bit complicated, and we will give only the result; for a derivation see Wesseling (2001) (Theorem 5.8.1). There it is shown that for the central scheme a necessary and sufficient stability condition is:

$$2\varepsilon\tau\Big(\frac{1}{h_1^2} + \frac{1}{h_2^2}\Big) \leq 1 \quad \text{and} \quad \frac{\tau}{2\varepsilon}(|u|^2 + |v|^2) \leq 1 \, , \tag{4.46}$$

and for the upwind scheme a sufficient stability condition is:

$$2\varepsilon\tau\Big(\frac{1}{h_1^2} + \frac{1}{h_2^2} + \frac{|u|}{2\varepsilon h_1} + \frac{|v|}{2\varepsilon h_2}\Big) \leq 1 \quad \text{and} \quad \frac{\tau}{2\varepsilon}\Big(\frac{|u|^2}{1+|u|h_1} + \frac{|v|^2}{1+|v|h_2}\Big) \leq 1 \, . \tag{4.47}$$

It is left to the reader to verify that equations (4.36) and (4.40) are included as special cases in (4.46) and (4.47), respectively.

Exercise 4.4.1. Suppose we have a numerical scheme for the convection equation

$$\frac{\partial\varphi}{\partial t}+c\frac{\partial\varphi}{\partial x}=0\,,\quad c\,\text{constant}\,,$$

that has a stability condition $c\tau/h < C$. What is the stability condition for this scheme for the nonlinear case $\partial\varphi/\partial t + \partial\varphi^m/\partial x = 0$?

Exercise 4.4.2. Derive equations (4.38) and (4.39).

Exercise 4.4.3. Show that scheme (4.38) is unconditionally unstable when $u < 0$. (This is one way to see why the downwind scheme is bad).

Exercise 4.4.4. Discretize (4.37) with the central scheme in space and the forward Euler scheme in time. Show that the scheme is unconditionally unstable.

Exercise 4.4.5. This exercise is meant to demonstrate the attractiveness of the geometric approach illustrated in Fig. 4.4. Determine $|g(\theta)|$, with $g(\theta)$ given by equation (4.39). Use analysis instead of geometry to find conditions on c such that $|g(\theta)| \le 1$, $\forall\theta$.

Exercise 4.4.6. Derive equations (4.43)—(4.45).

Exercise 4.4.7. Write down the upwind version of scheme (4.43).

4.5 Numerical experiments

Problem statement

Some numerical experiments will be presented for the following one-dimensional test problem:

$$\begin{aligned}&\frac{\partial\varphi}{\partial t}+u\frac{\partial\varphi}{\partial x}-\varepsilon\frac{\partial^2\varphi}{\partial x^2}=q, \qquad 0<x<1, \quad 0<t\le T\,,\\&q(t,x)=\beta^2\varepsilon\cos\beta(x-ut)\,,\end{aligned}\tag{4.48}$$

with ε and $u > 0$ constant, and β a parameter. An exact solution is given by

$$\varphi(t,x)=\cos\beta(x-ut)+e^{-\alpha^2\varepsilon t}\cos\alpha(x-ut)\,,\tag{4.49}$$

with α arbitrary. Spatial discretization is done with the second order central or with the first order upwind scheme on the vertex-centered grid of Fig. 2.1 with uniform mesh size h. For temporal discretization the ω-scheme is used. The resulting scheme can be written as

$$\frac{h_j}{\tau}(\varphi_j^{n+1}-\varphi_j^n)+\omega L_h\varphi_j^{n+1}+(1-\omega)L_h\varphi_j^n=h_j[\omega q_j^{n+1}(1-\omega)q_j^n]\,,\tag{4.50}$$

where h_j is the volume of the cell over which is integrated. We choose a Neumann condition at $x = 1$, so that $h_j = h,\ j \neq J,\ h_J = h/2$. In the interior we get for the central scheme:

$$\begin{aligned} L_h\varphi_j =& F_{j+1/2} - F_{j-1/2} \ , \\ F_{j+1/2} =& \frac{1}{2}u(\varphi_{j+1} + \varphi_j) - \frac{\varepsilon}{h}(\varphi_{j+1} - \varphi_j) \ , \quad j = 2, \cdots, J-1 \ . \end{aligned} \tag{4.51}$$

At the Neumann boundary we integrate over a half cell, and obtain

$$F_{J+1/2} = u\varphi_J - \varepsilon b(t) \ ,$$

assuming a Neumann condition $\partial\varphi(t,1)/\partial x = b(t)$.

Choice of time step, mesh size and time scale

The length scale $\mathcal{L}$ and time scale $\mathcal{T}$ of the exact solution are given by

$$\mathcal{L} = \pi/\max(\alpha,\beta), \quad \mathcal{T} = \min\{\mathcal{L}/u, (\varepsilon\alpha^2)^{-1}\} \ ,$$

where we take for the length scale of a harmonic function half its wavelength. We may expect accuracy to be sufficient if $\tau \ll \mathcal{T},\ h \ll \mathcal{L}$. For efficiency, we would like to avoid more stringent restrictions on τ and h, such as might arise from stability. In the numerical experiments to be described we take $\alpha = 4\pi,\ \beta = 2\pi$, so that $\mathcal{L} \cong 1/4$. We take mostly $h = 1/30$, giving $h/\mathcal{L} \cong 0.13$.

Remarks on the MATLAB program

The numerical experiments described below have been carried out with the code `cdns`. The matrix L_h is described in Sect. 2.3 for the cell-centered case (called A there), and is easily adapted to the vertex-centered case. It is generated (as in Sect. 2.3) by

```
L_h = spdiags([-beta0 beta0-beta1 beta1], -1:1, J, J);
```

The system to be solved for φ^{n+1} with the ω-scheme (4.50) can be written as

$$A\boldsymbol{\phi}^{n+1} = B\boldsymbol{\phi}^n + \boldsymbol{q} \ . \tag{4.52}$$

The matrices A and B can be generated by

```
x = linspace(0,1,J);              % Grid node positions
hh = h*ones(size(x'));
hh(J,1) = h/2;                    % Last cell has half size
D = spdiags([hh/tau],0,J,J);
```

```
A = D + omega*L_h;
B = D - (1 - omega)*L_h;
A(1,1) = 1; A(1,2) = 0;          % Correction for Dirichlet
B(1,1) = 0; B(1,2) = 0;          %   boundary condition
[L,U] = lu(A);                   % Save LU decomposition
```

Time stepping is done with

```
vn = U\(L\(B*vo + rhs));  vo = vn;
```

Numerical results

We prescribe a Dirichlet condition at $x = 0$ and a Neumann condition at $x = 1$. Initial and boundary conditions are chosen conforming with the exact solution (4.49). The left half of Fig. 4.3 shows a result. In this case we have

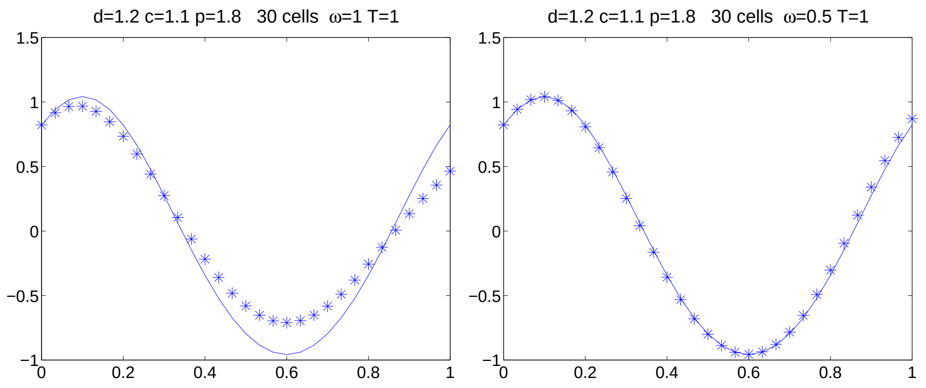

Fig. 4.3. Exact (—) and numerical solution (*) of (4.48). Central scheme; $u = 1.1$, $\varepsilon = 0.02$, $h = \tau = 1/30$, $t = 1$, $\alpha = 4\pi$, $\beta = 2\pi$. Left: $\omega = 1$, right: $\omega = 1/2$.

$\mathcal{T} \cong 0.23$, $\tau \cong 0.15\mathcal{T}$. The accuracy with $\omega = 1$ is disappointing. The cause is that the scheme is only first order accurate in time. With $\omega = 1/2$ the scheme is second order accurate in time, and the accuracy is much better. In this case we have $d \equiv 2\varepsilon\tau/h^2 = 1.2$, $\tau u^2/(2\varepsilon) = 1.0$, so that the explicit ($\omega = 0$) scheme is instable according to the one-dimensional version ($v = 0$, $h_2 = \infty$) of the stability conditions (4.46). To get comparable accuracy as in the right part of the figure with $\omega = 0$ we find that we need to decrease (in order to compensate for first order accuracy in time) τ to 1/300 (results not shown). This gives an elapsed time (with `tic...toc`) of 0.287, whereas for the right part of the figure we find an elapsed time of 0.027. The stability of the explicit ($\omega = 0$) scheme constrains the time step much more than accuracy. This is inefficient, so that it pays of to use a more complicated scheme with more work per time step, but with a less severe stability restriction on the time step.

Exercise 4.5.1. Verify the stability conditions (4.46) by numerical experiments with the MATLAB program used in this section.

Exercise 4.5.2. Make a cell-centered version of the code `cdns` .

Some self-test questions

Write down the instationary convection-diffusion equation.

Formulate the maximum principle for the instationary convection-diffusion equation.

Write down the one-dimensional heat equation, discretize it with the forward Euler scheme and write down the matrix of the scheme.

Define the global and local truncation error for the instationary convection-diffusion equation.

Formulate Lax's equivalence theorem.

Define zero-stability and absolute stability.

Write down the Fourier series for a grid function $\varDelta_j$ in d dimensions.

Write down the ω-scheme.

Which are the interesting values of ω for the ω-scheme? Why?

Show that the ω-scheme is unconditionally stable for $\omega \geq 1/2$.

5. The incompressible Navier-Stokes equations

5.1 Introduction

In this chapter, the incompressible Navier-Stokes equations in Cartesian coordinates discretized on Cartesian nonuniform grids will be considered, discussing most of the basic numerical principles in a simple setting. In practical applications, of course, nonuniform grids and general coordinate systems are prevalent; these will not be discussed here; see Wesseling (2001).

We can be relatively brief in discussing discretization of the Navier-Stokes equations, because we prepared the ground in our extensive discussion of the convection-diffusion equation in Chapters 2—4. Therefore it wil not be necessary to discuss again the various possibilities for discretizing convection, or stability conditions.

Only the *primitive variable formulation* will be discussed. This means that the velocity components and the pressure will be used as unknowns.

Purpose of this chapter

The purpose of this chapter is to present a numerical method for the incompressible Navier-Stokes equations. In particular, we will:

• Present suitable boundary conditions;
• Describe spatial discretization on a staggered grid;
• Describe the ω-scheme, the Adams-Bashforth scheme and the Adams-Bashforth-ω scheme for discretization in time;
• Show how to linearize with the Picard or Newton method;
• Describe the pressure-correction method;
• Discuss stability conditions;
• Present some numerical experiments with the MATLAB codes `ns1` and `ns2`;
• Discuss outflow boundary conditions;
• Discuss efficiency.

5.2 Equations of motion and boundary conditions

Equations of motion

We restrict ourselves to the two-dimensional case. The equations of motion have been discussed in Chap. 1. For ease of reference, the equations to be considered are repeated here. We assume incompressible flow, i.e. $D\rho/Dt = 0$, so that (*cf.* (1.8))

$$u_x + v_y = 0 \ , \tag{5.1}$$

where we denote partial differentiation by a subscript. The density is taken constant. The dimensionless incompressible Navier-Stokes equations are given by equation (1.23):

$$\begin{aligned} u_t + uu_x + vu_y &= -p_x + \mathrm{Re}^{-1}(u_{xx} + u_{yy}) \ , \\ v_t + uv_x + vv_y &= -p_y + \mathrm{Re}^{-1}(v_{xx} + v_{yy}) \ . \end{aligned} \tag{5.2}$$

By adding $u(u_x + v_y)$ and $v(u_x + v_y)$ (both zero according to (5.1)), respectively, this can be put in *conservation form*:

$$\begin{aligned} u_t + (uu)_x + (vu)_y &= -p_x + \mathrm{Re}^{-1}(u_{xx} + u_{yy}) \ , \\ v_t + (uv)_x + (vv)_y &= -p_y + \mathrm{Re}^{-1}(v_{xx} + v_{yy}) \ . \end{aligned} \tag{5.3}$$

The following units are chosen: velocity: U; length: L; density: ρ_0; pressure: $\rho_0 U^2$. Then the Reynolds number is given by

$$\mathrm{Re} = \rho_0 UL/\mu \ ,$$

with μ the dynamic viscosity coefficient, which was already assumed to be constant above.

The *deviatoric stress tensor* (i.e. the viscous part of the stress tensor) is denoted by $\sigma_{\alpha\beta}$. From equation (1.15) it follows that

$$\sigma_{xx} = 2\mathrm{Re}^{-1}u_x, \quad \sigma_{xy} = \sigma_{yx} = \mathrm{Re}^{-1}(u_y + v_x), \quad \sigma_{yy} = 2\mathrm{Re}^{-1}v_y \ . \tag{5.4}$$

The governing equations (5.1) and (5.2) need to be accompanied by initial and boundary conditions.

Initial conditions

For the momentum equations (5.2) the following initial conditions are required:

$$u(0, \boldsymbol{x}) = u_0(\boldsymbol{x}) \ , \quad v(0, \boldsymbol{x}) = v_0(\boldsymbol{x}) \ ,$$

with the prescribed initial velocity field $\boldsymbol{u}_0$ satisfying the continuity equation (5.1). Note that there is no initial condition for the pressure, since p_t does not occur.

No-slip condition

Viscous fluids cling to solid surfaces. This is called the *no-slip condition.* At a solid surface we have

$$\boldsymbol{u}(t,\boldsymbol{x}) = \boldsymbol{v}(t,\boldsymbol{x}) \; , \tag{5.5}$$

with $\boldsymbol{v}(t,\boldsymbol{x})$ the local wall velocity. The Dirichlet condition (5.5) holds also at open parts of the boundary where the velocity is prescribed, which may be the case at an inflow boundary. But at an inflow boundary one may also prescribe condition (5.6) given below.

Free surface conditions

At a free surface the tangential stress components are zero. We consider only the very special case where the free surface is fixed at $y = a =$ constant. For the general case, see Sect. 6.2 of Wesseling (2001). At a fixed free surface, the normal velocity and the tangential stress are zero:

$$v(t,x,a) = 0, \quad u_y(t,x,a) = 0 \; , \tag{5.6}$$

where we have used

$$\sigma_{xy}(t,x,a) = \mathrm{Re}^{-1}(u_y + v_x)(t,x,a) = \mathrm{Re}^{-1}u_y(t,x,a) \; .$$

We see that we have a Dirichlet condition for the normal velocity and a Neumann condition for the tangential velocity. A truly free surface moves, its shape must be determined and follows from the condition that the normal stress equals the ambient pressure. This case will not be considered. Conditions (5.6) may also arise at a plane of symmetry. In special cases one may wish to prescribe non-zero tangential stress in (5.6), for example, when one wishes to take the influence of wind shear on a water surface into account.

Inflow conditions

The momentum equations resemble convection-diffusion equations for u and v, so that the insights gained in the convection-diffusion equation in Chapt. 2—4 provide guidelines for numerical approximation. Based on what we learned about the convection-diffusion equation, we prescribe Dirichlet conditions at an inflow boundary. If, for example, $x = 0$ is an inflow boundary, we prescribe

$$u(t,0,y) = U(t,y) \; , \quad v(t,0,y) = V(t,y) \; . \tag{5.7}$$

Outflow conditions

At an outflow boundary, often not enough physical information is available on which to base a sufficient number of boundary conditions. Usually only the pressure is known. This is not as serious as it may seem, because when $\mathrm{Re} \gg 1$ 'wrong' information generated by an artificial boundary condition propagates upstream only over a distance of $\mathcal{O}(\mathrm{Re}^{-1})$. This is plausible because of the resemblance of (5.2) to the convection-diffusion equation, and may in fact be shown directly by applying singular perturbation analysis to (5.2) in a similar manner as in Sect. 3.2. In order to avoid spurious numerical wiggles it is advisable to choose as artificial outflow condition a homogeneous Neumann condition for the tangential velocity. For an outflow boundary at $x = a$ this gives:

$$p(t, a, y) = p_\infty, \quad v_x(t, a, y) = 0 \; . \tag{5.8}$$

Compatibility condition

At every part of the boundary exactly one of the boundary conditions (5.5), (5.6) or (5.8) needs to be prescribed. If it is the case that along the whole of the boundary $\partial\Omega$ the normal velocity $u^n(t, \boldsymbol{x})$ is prescribed, then it follows from (5.1) and the divergence theorem that the following *compatibility condition* must be satisfied:

$$\int\limits_{\partial\Omega} u^n(t, \boldsymbol{x}) dS = 0 \; . \tag{5.9}$$

It can be shown theoretically (for further information, see Sect. 6.2 of Wesseling (2001)) that in order for (5.1), (5.2) to be well-posed, the normal component of the prescribed initial velocity field $\boldsymbol{u}_0(\boldsymbol{x})$ and a prescribed normal velocity component must match at $t = 0$:

$$\boldsymbol{u}_0(\boldsymbol{x}) \cdot \boldsymbol{n} = u^n(0, \boldsymbol{x})$$

on parts of $\partial\Omega$ where the normal velocity is prescribed. But the tangential components of the initial and boundary velocity fields need not match at $t = 0$. Therefore, for example, a sliding wall may be set in motion instantaneously at $t = 0$ in a fluid originally at rest, but one should not let the speed of an arbitrarily shaped body or of an inlet flow change discontinuously.

5.3 Spatial discretization on staggered grid

Let the domain be rectangular and be covered with a nonuniform grid consisting of rectangular cells as sketched in Fig. 5.1. The oldest and most straightforward approach to discretizing the Navier-Stokes equations in space is the

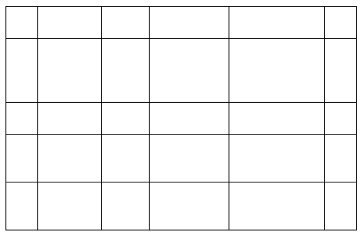

Fig. 5.1. Rectangular nonuniform grid

method proposed in 1965 by Harlow and Welch (see Sect 6.4 of Wesseling (2001) for references to the literature). On orthogonal grids it remains the method of choice.

Staggered grid

Grid points for different unknowns are staggered with respect to each other. The pressure resides in the cell centers, whereas the cell face centers contain the normal velocity components, *cf.* Fig. 5.2. The grid nodes are numbered

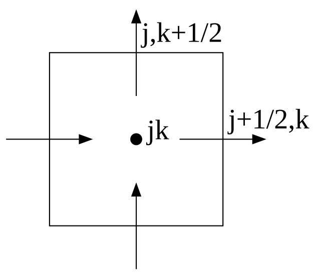

Fig. 5.2. Staggered placement of unknowns; $\rightarrow$, $\uparrow$: velocity components; $\bullet$: pressure.

as follows. The cell with center at $\boldsymbol{x}_{jk}$ is called

$$\Omega_{jk}, \quad j = 1, \cdots, J, \quad k = 1, \cdots, K\ .$$

The horizontal and vertical sides of Ω_{jk} have length h_j^x and h_k^y, respectively. The center of the 'east' side of Ω_{jk} is called $\boldsymbol{x}_{j+1/2,k}$, etc., see Fig. 5.2. Hence, Ω_{jk} contains the following unknowns: p_{jk}, $u_{j\pm1/2,k}$, $v_{j,k\pm1/2}$. Note that with a staggered grid we always have a mixture of vertex-centered and cell-centered discretization. Unavoidably, at a boundary, some unknowns will have nodes upon it, whereas other unknowns have no nodes on this boundary, but half a mesh size removed. Therefore it is fortunate, as seen in Chapt. 2, that vertex-centered and cell-centered discretization are on equal footing as far as global accuracy and ease of implementation of boundary conditions are concerned.

Discretization of continuity equation

The continuity equation (5.1) is integrated over Ω_{jk}, resulting in

$$h_k^y u|_{j-1/2,k}^{j+1/2,k} + h_j^x v|_{j,k-1/2}^{j,k+1/2} = 0 \ . \tag{5.10}$$

The advantage of the staggered placement of the unknowns is that no further approximation is necessary in this equation.

Discretization of momentum equations

Finite volume integration takes place over control volumes surrounding u and v grid points, with sides through neighboring pressure points. For example, the control volume for $u_{j+1/2,k}$ consists of the union of half of Ω_{jk} and half of $\Omega_{j+1,k}$, as illustrated in Fig. 5.3. This control volume is called $\Omega_{j+1/2,k}$. Finite volume integration gives

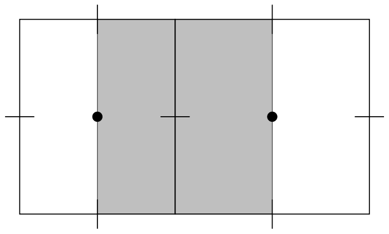

Fig. 5.3. Control volume $\Omega_{j+1/2,k}$ for $u_{j+1/2,k}$.

$$\begin{aligned} \int\limits_{\Omega_{j+1/2,k}} & \left[u_t + (uu + p - \mathrm{Re}^{-1}u_x)_x + (uv - \mathrm{Re}^{-1}u_y)_y\right] d\Omega \cong \\ & h_{j+1/2}^x h_k^y \frac{du_{j+1/2,k}}{dt} + h_k^y \big(uu + p - \mathrm{Re}^{-1}u_x\big)_{jk}^{j+1,k} \\ & + h_{j+1/2}^x \big(uv - \mathrm{Re}^{-1}u_y\big)_{j+1/2,k-1/2}^{j+1/2,k+1/2} = 0 \ . \end{aligned} \tag{5.11}$$

Here p occurs only in its own nodal points, and further approximation is not necessary. But the derivatives and u and v need to be approximated in terms of surrounding nodes.

The derivatives are approximated as follows:

$$\begin{aligned} u_x|_{jk} &\cong (u_{j+1/2,k} - u_{j-1/2,k})/h^x_j , \\ u_y|_{j+1/2,k+1/2} &\cong (u_{j+1/2,k+1} - u_{j+1/2,k})/h^y_{k+1/2} . \end{aligned}$$

The central scheme for the inertia term is obtained with the following approximations:

$$\begin{aligned} u^2_{jk} &\cong (u^2_{j-1/2,k} + u^2_{j+1/2,k})/2 , \\ (uv)_{j+1/2,k+1/2} &\cong (u_{j+1/2,k} + u_{j+1/2,k+1})(v_{j,k+1/2} + v_{j+1,k+1/2})/4 . \end{aligned}$$

The resulting stencil for $u_{j+1/2,k}$ is given in Fig. 5.4.

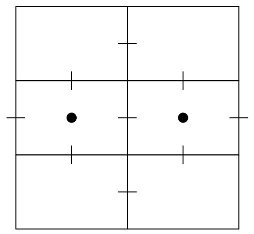

Fig. 5.4. Stencil for $u_{j+1/2,k}$.

The upwind scheme for the inertia term is obtained as follows. We do not wish to test on the sign of u and v, because `if` statements in `for` loops are very computer time consuming, cf. Sect. 4.3. We note that upwind approximation of a term $ud\varphi/dx$ can be implemented as follows, without an `if` statement:

$$u\varphi(x_{j+1/2}) \cong \frac{1}{2}\left[(u + |u|)\varphi_j + (u - |u|)\varphi_{j-1}\right] .$$

By using this idea we obtain the following upwind approximation of the inertia terms:

$$\begin{aligned} u^2_{jk} &\cong \frac{1}{4}\left[(u + |u|)^2_{j-1/2,k} + (u - |u|)^2_{j+1/2,k}\right] , \\ (uv)_{j+1/2,k+1/2} &\cong \frac{1}{2}\left[(v + |v|)_{j+1/2,k+1/2} u_{j+1/2,k}\right. \\ &\quad \left. +(v - |v|)_{j+1/2,k+1/2} u_{j+1/2,k+1}\right] , \end{aligned} \tag{5.12}$$

where

$$v_{j+1/2,k+1/2} \equiv (v_{j,k+1/2} + v_{j+1,k+1/2})/2 .$$

The momentum equation for v is discretized similarly. This completes finite volume discretization in the interior. We continue with the boundary conditions. On the staggered grid the implementation of the boundary conditions (5.5)—(5.8) is just as simple and done in the same way as for the convection-diffusion equation in Chapters 2 and 3.

The no-slip condition

Let $y = 0$ be a wall moving horizontally with velocity $U(t)$. Then the lower side of $\Omega_{j,1}$ is at the boundary. We have $v_{j,1/2} = 0$, so that no discretization for $v_{j,1/2}$ is required. In the finite volume scheme for $u_{j+1/2,1}$ we need, according to the stencil presented in Fig. 5.4, $u_{j+1/2,0}$, which is not available, because $\boldsymbol{x}_{j+1/2,0}$ is outside the domain. Values outside the domain are called *virtual values.*We write $u_{j+1/2,0} + u_{j+1/2,1} = 2U(t)$, so that

$$u_{j+1/2,0} = 2U(t) - u_{j+1/2,1} \ , \tag{5.13}$$

which is used to eliminate the virtual value $u_{j+1/2,0}$.

Free surface conditions

Let $y = a$ be a free surface boundary or a symmetry boundary, so that we have conditions (5.6). Let Ω_{jk} be at the boundary. We have $v_{j,K+1/2} = 0$, so that no discrete equation is required for $v_{j,K+1/2}$. In the stencil for $u_{j+1/2,K}$ we have $u_{j+1/2,K+1}$, according to Fig. 5.4, which is outside the domain and has to be eliminated. We put $0 = u_y \cong (u_{j+1/2,K+1} - u_{j+1/2,K})/h_K^y$, so that

$$u_{j+1/2,K+1} = u_{j+1/2,K} \ . \tag{5.14}$$

Inflow conditions

Let $x = 0$ be an inflow boundary, so that Ω_{1k} is at the boundary. According to (5.7) we have Dirichlet conditions for u and v, so that the situation is almost the same as for the no-slip condition. We put

$$u_{1/2,k} = U(t, y_k) \ , \quad v_{0,k+1/2} = 2V(t, y_{k+1/2}) - v_{1,k+1/2} \ . \tag{5.15}$$

Outflow conditions

Let Ω_{Jk} be at an outflow boundary $x = a$. We need discrete equations for $u_{J+1/2,k}$ and $v_{J,k+1/2}$. The control volume for $u_{J+1/2,k}$ consists of half of Ω_{Jk}, as illustrated in Fig. 5.5. Finite volume integration gives, similar to (5.11),

$$\int\limits_{\Omega_{J+1/2,k}} \left[u_t \quad +(uu + p - \mathrm{Re}^{-1}u_x)_x + (uv - \mathrm{Re}^{-1}u_y)_y\right] d\Omega \cong$$

$$\tfrac{1}{2}h_J^x h_k^y du_{J+1/2,k}/dt + h_k^y\left(uu + p - \mathrm{Re}^{-1}u_x\right)_{Jk}^{J+1/2,k}$$
$$+\tfrac{1}{2}h_J^x\left(uv - \mathrm{Re}^{-1}u_y\right)_{J+1/2,k-1/2}^{J+1/2,k+1/2} = 0 \ .$$

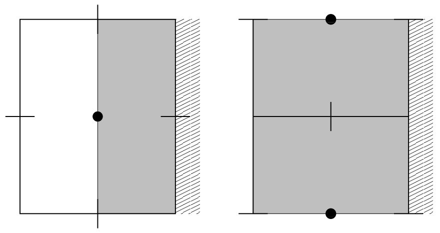

Fig. 5.5. Control volumes for $u_{J+1/2,k}$ and $v_{J,k+1/2}$ at outflow boundary.

Compared to the interior case, we have to change only terms at or near the outflow boundary. We put the normal stress at $\boldsymbol{x}_{J+1/2,k}$ equal to the prescribed pressure (cf. (5.8)):

$$(p - \mathrm{Re}^{-1} u_x)_{J+1/2,k} = p_\infty \ .$$

Furthermore,

$$(uv)_{J+1/2,k+1/2} = v_{J,k+1/2} u_{J+1/2,k+1/2} \ ,$$

where $u_{J+1/2,k+1/2}$ is approximated with the upwind scheme or the central scheme. Finally,

$$\mathrm{Re}^{-1}(u_y)_{J+1/2,k+1/2} \cong (u_{J+1/2,k+1} - u_{J+1/2,k})/h^y_{J+1/2,k+1/2} \ .$$

The scheme for $v_{J,k+1/2}$ brings nothing new. A virtual value $v_{J+1,k+1/2}$ occurs as may be seen form the stencil of the v-momentum equation, which is obtained from Fig. 5.4 by rotation over 90^o. This virtual value is eliminated by using $v_x = 0$ (cf. (5.8)), which results in

$$v_{J+1,k+1/2} = v_{J,k+1/2}$$

(*cf.* (5.14)).

Summary of equations

We put all unknown velocity components in some order in an algebraic vector u and all pressure unknowns in an algebraic vector p, and we divide by the coefficients of the time derivatives. Then the scheme can be written as a differential-algebraic system of the following structure:

$$\frac{du}{dt} + N(u) + Gp = f(t) \ , \qquad Du = g(t) \ . \tag{5.16}$$

Here N is a nonlinear algebraic operator arising from the discretization of the inertia and viscous terms, G is a linear algebraic operator representing the discretization of the pressure gradient, D is a linear algebraic operator representing the discretization of divergence operator in the continuity equation, and f and g are known source terms, arising from the boundary conditions. This is called a differential-algebraic system because it is a mixture of a system of ordinary differential equations (the first member of (5.16)), and algebraic equations (the second member of (5.16)). This completes our description of spatial discretization on the staggered grid.

5.4 Temporal discretization on staggered grid

We now discretize also in time. Equation (5.16) is our point of departure. With the forward Euler scheme we obtain:

$$\frac{1}{\tau}(u^n - u^{n-1}) + N(u^{n-1}) + Gp^{n-1/2} = f(t^{n-1}) \ , \qquad Du^n = g(t^n) \ . \quad (5.17)$$

Note that we have to take the pressure term at the new time t^n, or for better accuracy in time, at $t^{n-1/2}$, in order to have degrees of freedom to satisfy the algebraic solenoidality constraint (second equation of (5.17)). In other words, the pressure always has to be taken implicitly; this is a consequence of the differential-algebraic nature of (5.16). In incompressible flows, the pressure acts as a Lagrange multiplier, that makes it possible to satisfy the continuity equation.

With the ω-scheme we obtain:

$$\begin{aligned}\frac{1}{\tau}(u^n - u^{n-1}) &+ \omega N(u^n) + (1-\omega)N(u^{n-1}) + Gp^{n-1/2} \\ &= \omega f(t^n) + (1-\omega) f(t^{n-1}) \ , \qquad Du^n = g(t^n) \ .\end{aligned} \quad (5.18)$$

Linearization

Equation (5.18) is a nonlinear system, because the nonlinear inertia term makes the operator N nonlinear. To make the system more easily solvable we linearize the operator N. *Newton linearization* works as follows, writing $\delta u = u^n - u^{n-1}$:

$$N(u^n) = N(u^{n-1} + \delta u) \cong N(u^{n-1}) + C(u^{n-1})\delta u \ ,$$

where $C(u)$ is the Jacobian of N evaluated at u. We clarify this by taking as an example one term of the inertia term (cf. (5.11)):

$$(u_{jk}^{n+1})^2 = (u_{jk}^n + \delta u_{jk})^2 \cong (u_{jk}^n)^2 + 2u_{jk}^n \delta u_{jk} \ .$$

We eliminate δu and obtain:

$$(u_{jk}^{n+1})^2 \cong 2u_{jk}^n u_{jk}^{n+1} - (u_{jk}^n)^2 \ ,$$

which is linear in the unknown u_{jk}^{n+1}. *Picard linearization* works as follows:

$$(u_{jk}^{n+1})^2 \cong u_{jk}^n u_{jk}^{n+1} \ .$$

This is simpler, but temporal accuracy decreases to $\mathcal{O}(\tau)$. A second order accurate approximation is obtained if one replaces u^n by an extrapolation to t^{n+1}:

$$(u_{jk}^{n+1})^2 \cong (2u_{jk}^n - u_{jk}^{n-1})u_{jk}^{n+1} \ . \tag{5.19}$$

We will call this *extrapolated Picard linearization.*

After linearization, the matrix C that one obtains changes every time step, because C depends on u^{n-1}. Matrix generation is computer time consuming. Therefore a time step will be cheaper if the inertia term is discretized explicitly. This gives us a so-called IMEX (implicit-explicit) scheme. The resulting scheme is cheaper, because the implicit operator is now linear and independent of time, so that the corresponding matrix has to be generated only once, and other ingredients necessary for solving, such as an LU factorization, need also to be prepared only once. To maintain second order accuracy in time, it is attractive to use for the explicit part not the forward Euler scheme, but a second order explicit scheme, such as the *Adams-Bashforth scheme.* For a system of ordinary differential equations $dw/dt = f(w,t)$ this scheme is given by

$$\frac{1}{\tau}(w^n - w^{n-1}) = \frac{3}{2}f(w^{n-1}, t^{n-1}) - \frac{1}{2}f(w^{n-2}, t^{n-2}) \ . \tag{5.20}$$

This is called a *two-step method*, because two time steps are involved. At the initial time $t = 0$ one may define $w^{-1} = w^0$. Application to the Navier-Stokes equations takes place as follows. Let $N(u)$ in (5.18) be split in a nonlinear inertia part C and a linear viscous part B, as follows:

$$N(u) = C(u) + Bu \ .$$

Then the *Adams-Bashforth-Crank-Nicolson scheme* is obtained by using (5.18) for Bu and (5.20) for $C(u)$, so that we obtain:

$$\begin{aligned}\frac{1}{\tau}(u^n - u^{n-1}) + \ & \tfrac{3}{2}C(u^{n-1}) - \tfrac{1}{2}C(u^{n-2}) + \tfrac{1}{2}B(u^n + u^{n-1}) + Gp^{n-1/2} \\ &= f(t^{n+1/2}) \ , \qquad Du^n = g(t^n) \ .\end{aligned}$$

General formulation on staggered grid

As seen from these examples, time stepping methods applied to equation (5.11) can generally be written as

$$\begin{aligned} &A(u^n) + \tau G p^{n-1/2} = r^n \ , \\ &Du^n = g(t^n) \ , \end{aligned} \tag{5.21}$$

where r^n is known from previous time steps and the boundary conditions. For explicit methods, A is the identity I. The system (5.21) is a coupled system for u^n and $p^{n-1/2}$. Computing time is reduced if $p^{n-1/2}$ and u^n can be solved for separately. To this end the following method has been devised, which is the method of choice for nonstationary problems.

Pressure-correction method

Equation (5.21) is not solved as it stands, but first a prediction u^* of u^n is made that does not satisfy the continuity equation. Then a correction is computed involving the pressure, such that the continuity equation is satisfied. The method is given by:

$$A(u^*) + \tau G p^{n-3/2} = r^n \ , \tag{5.22}$$

$$u^n - u^* + \tau G(p^{n-1/2} - p^{n-3/2}) = 0 \ , \tag{5.23}$$

$$Du^n = g(t^n) \ . \tag{5.24}$$

Equation (5.22) more or less amounts to solving discretized convection-diffusion equations for the predicted velocity components. We use the best available guess for the pressure, namely $p^{n-3/2}$. Equation (5.23) is motivated by the fact, that if in the explicit case, where A is the identity, we eliminate u^* from (5.22) and (5.23), then the original system (5.21) is recovered. The pressure can be computed by applying the operator D to (5.23) and using (5.24), resulting in

$$DG\delta p = \frac{1}{\tau}\{Du^* - g(t^n)\}, \quad p^{n-1/2} = p^{n-3/2} + \delta p \ . \tag{5.25}$$

After $p^{n-1/2}$ has been computed, u^n follows from (5.23).

Equation (5.23) can be regarded as a correction of u^* for the change in pressure. Therefore (5.22)–(5.25) is called the *pressure-correction method*. It is an example of a *fractional step method*, in which a time step is split up in sub-steps, and different physical effects are accounted for separately in the sub-steps. Here pressure forces are accounted for in the second sub-step, and inertia and friction in the first sub-step. Confusingly, the term pressure-correction method is often also applied to various iterative methods to solve the stationary Navier-Stokes equations, in which velocity and pressure updates are carried out not simultaneously but successively. Such methods will be encountered in Chap. 6, where they will be called *distributive iteration methods*. These should not be confused with the pressure-correction method used in time accurate schemes as formulated above. This method may also be called a *projection method*, because in (5.23) the new velocity u^n is the projection of the intermediate velocity field u^* on the space of velocity fields with discretized divergence equal to zero.

Remembering that divgrad equals the Laplacian, we see that (5.25) looks very much like a discrete Poisson equation; it is frequently called the pressure Poisson equation. Note that no boundary condition needs to be invoked for δp (fortunately, for no such condition is given with the original equations, at least not on the complete boundary), because the boundary conditions have already been taken into account in the construction of D, G and g; the operator DG works exclusively on pressure values at grid points in the interior of the domain.

Even if the method is explicit (A is a diagonal matrix), we still have to solve an implicit system for δp. This is an unavoidable consequence of the differential-algebraic nature of (5.16).

As we remarked before, by elimination of u^* it is easily seen that in the explicit case the pressure-correction method (5.22)–(5.25) is equivalent to (5.21), and that this remains true if $p^{n-3/2}$ is neglected in (5.22) and (5.23). But in the implicit case this does not hold, and inclusion of a sufficiently accurate first guess, such as $p^{n-3/2}$, for the pressure in (5.22) seems to be necessary to obtain full, i.e. $\mathcal{O}(\tau^2)$, temporal accuracy. This may make it necessary to compute the initial pressure field at the starting step ($n = 1$), to be used instead of $p^{-1/2}$. This may be done as follows. Application of D to (5.16) at $t = 0$ gives

$$dg(0)/dt + DN(u(0)) + DGp(0) = Df(0) . \qquad (5.26)$$

After solving $p(0)$ from (5.26), we put $p^{-1/2} = p(0)$.

Discrete compatibility condition

In case the pressure is not involved in any of the boundary conditions, it follows from the incompressible Navier-Stokes equations that the pressure is determined up to a constant. The system (5.25) for δp is singular in this case, and (5.25) has a solution only if the right-hand side satisfies a compatibility condition. The boundary conditions discussed in Sect. 5.2 are such that if the pressure is not involved in any of the boundary conditions, then the normal velocity component is prescribed all along the boundary, and the compatibility condition (5.9) is satisfied. Summing the discrete continuity equation (5.10) over all cells reduces, due to cancellation in the interior, to the following sum over boundary points:

$$\sum_{k=1}^{K} h_k^y u\Big|_{(1/2,k)}^{(J+1/2,k)} + \sum_{j=1}^{J} h_j^x v\Big|_{(j,1/2)}^{(j,K+1/2)} = 0 \, , \qquad (5.27)$$

where u and v are the prescibed velocity components at the boundary. If (5.27) is not satisfied exactly one should adjust u and v at the boundaries, which should not be difficult, because of the compatibility condition (5.9). If (5.27) holds, then it turns out that the elements of the right-hand side vector of (5.25) sum to zero, which is precisely the compatibility condition required for existence of solutions of (5.25). If one desires to make the solution unique one can fix δp in some point, but iterative methods usually converge faster if one lets δp float.

Temporal accuracy

For literature on the accuracy of the pressure-correction method, see Wesseling (2001), Sect. 6.6 and references quoted there. Indications are that the temporal accuracy of $\boldsymbol{u}^n$ is of the same order as rhe order of accuracy of the underlying time stepping method (for example, $\mathcal{O}(\tau^2)$ for Adams-Bashforth-Crank-Nicolson), but that the accuracy of $p^{n-1/2}$ is only $\mathcal{O}(\tau)$, irrespective of the time stepping method used. If one desires, a pressure field with improved accuracy can be obtained after u^n has been computed (with the pressure-correction method) by proceeding in the same way as in the derivation of (5.26), leading to the following equation for $p^{n-1/2}$:

$$dg(t^n)/dt + DN(u^n) + DGp^{n-1/2} = Df(t^n) \ ,$$

which very likely results in a pressure field $p^{n-1/2}$ with the same order of temporal accuracy as the velocity field.

Stability

For stability of (5.22)—(5.25), it seems necessary that (5.22) is stable. It is conjectured that this is sufficient for the stability of (5.22)—(5.25); numerical evidence supports this conjecture. We restrict ourselves to Fourier stability analysis of (5.22). To this end (5.22) is linearized, the coefficients are taken constant ('frozen'), the boundary conditions are assumed to be periodic, and the known source terms $\tau G p^{n-3/2}$ and r^n are neglected. Hence, carrying out Fourier stability analysis for (5.22) implies that the discretization of the following simplified and linearized version of (5.2) is considered:

$$\begin{aligned} u_t + Uu_x + Vu_y - \mathrm{Re}^{-1}(u_{xx} + u_{yy}) = 0 \ , \\ v_t + Uv_x + Vv_y - \mathrm{Re}^{-1}(v_{xx} + v_{yy}) = 0 \ , \end{aligned} \tag{5.28}$$

Equation (5.28) consists of decoupled and identical convection-diffusion equations, for which Fourier stability analysis is presented in Chap. 4. The stability analysis of the Adams-Bashforth-Crank-Nicolson scheme is a bit involved and is not presented in Chap. 4. In Sect. 6.6 of Wesseling (2001) stability conditions for the Adams-Bashforth-Crank-Nicolson scheme are derived. With the central scheme for the inertia terms we have:

$$\begin{aligned}
\tau &\le \max[\tau_1, \min\{\tau_2, \tau_3\}] \ , \\
\tau_1 &\equiv \frac{4}{3\mathrm{Re}}\big[U^2 + V^2\big]^{-1} \ , \\
\tau_2 &\equiv \frac{\mathrm{Re}}{4}\big[(h^x)^{-2} + (h^y)^{-2}\big]^{-1} \ , \\
\tau_3 &\equiv \big(\frac{3}{\mathrm{Re}}\big)^{1/3}\big[(U^2/h^x)^{2/3} + (V^2/h^y)^{2/3}\big]^{-1} \ .
\end{aligned} \tag{5.29}$$

For the upwind scheme:

$$\begin{aligned}
\tau &\le \max[\tau_1, \min\{\tau_2, \tau_3\}] \ , \\
\tau_1 &\equiv \frac{2}{3\mathrm{Re}}\Big[\frac{U^2}{2 + Uh^x\mathrm{Re}} + \frac{V^2}{2 + Vh^y\mathrm{Re}}\Big]^{-1} \ , \\
\tau_2 &\equiv \frac{\mathrm{Re}}{4}\Big[\frac{1 + Uh^x\mathrm{Re}/2}{(h^x)^2} + \frac{1 + Vh^y\mathrm{Re}/2}{(h^y)^2}\Big]^{-1} \ , \\
\tau_3 &\equiv \big(\frac{3}{\mathrm{Re}}\big)^{1/3}\Big[\Big(\frac{U^4}{(h^x)^2 + U(h^x)^3\mathrm{Re}/2}\Big)^{1/3} + \Big(\frac{V^4}{(h^y)^2 + V(h^y)^3\mathrm{Re}/2}\Big)^{1/3}\Big]^{-1} \ .
\end{aligned} \tag{5.30}$$

The computing cost of checking in every grid point whether the stability condition is satisfied is often not negligible, so in practice this is often done only every 5 or 10 time steps or so, or only once, if one has a reasonable *a priori* estimate of $U \quad (= 2\max(u))$ and $V \quad (= 2\max(v))$. Here the factor 2 arises from the nonlinearity of the inertia terms, in the same way as for the Burgers equation in Sect. 4.4.

Exercise 5.4.1. What is the Newton linearization of uv?

5.5 Numerical experiments

The schemes just described have been implemented in the MATLAB codes `ns1` (ω-scheme) and `ns2` (Adams-Bashforth-ω-scheme), on a nonuniform Cartesian grid of the type shown in Fig. 5.1. For `ns1`, Picard linearization or extrapolated Picard linearization is used for the inertia terms. Three types of boundary conditions have been implemented: inflow, outflow and no-slip. The boundaries can be divided in segments, on which one can choose different boundary conditions and numbers of grid cells. For instance, we can do the so-called backward-facing step problem, illustrated in Fig. 5.6. Because the domain is assumed to be rectangular, we cannot handle the narrow inflow part at the left, and use the rectangular domain shown at the right. We choose $|AB| = |BC| = 1, |CD| = L$, with L to be chosen. It turns out that a recirculation zone is present attached to BC, with length dependent on the Reynolds number Re. The length of the domain L must be larger than the

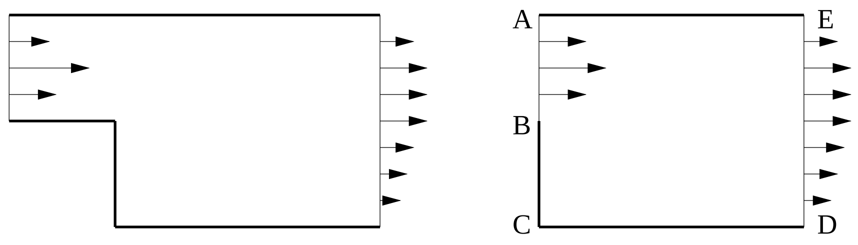

Fig. 5.6. The backward-facing step problem.

length of the recirculation zone, in order to have $u > 0$ at DE, so that the outflow boundary condition (5.8) is appropriate. Let the Reynolds number be based on the length of AB and the average inflow velocity. On the segment AB we prescribe inflow: $v = 0$, $u = f(y)$, f parabolic, such that the average inflow velocity is 1. On DE we prescribe outflow, and on the remaining boundary segments no-slip. The grid is uniform with $nx \times ny$ cells.

Wrong outflow conditions

First, we prescribe Dirichlet conditions, namely a parabolic outflow profile:

$$u = g(y), \quad v = 0 \tag{5.31}$$

on the outflow boundary DE. This is against the advice given concerning outflow boundary conditions in Sect. 5.2. Let us see what goes wrong. Fig. 5.7 shows a result, using Picard linearization. Note that the horizontal and vertical scales are different in the left part of the figure. The relative change

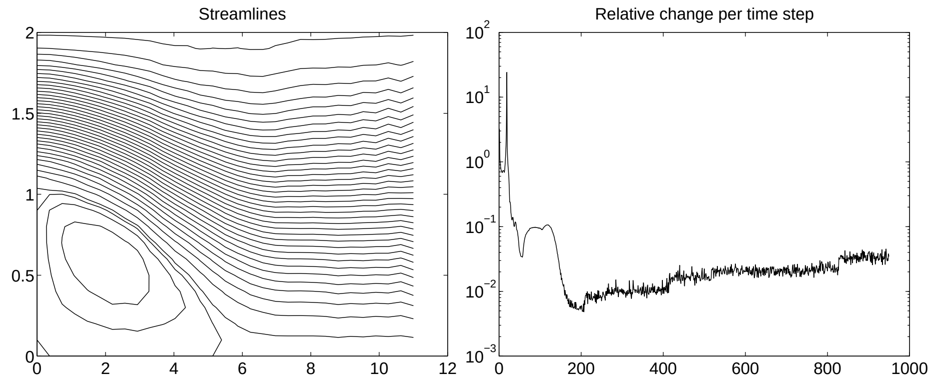

Fig. 5.7. Streamlines and convergence history for backward-facing step problem, code `ns1`, Dirichlet outflow conditions. Re $= 100$, $\tau = 0.2, t = 190, nx = 30, ny = 20, \omega = 1$, central scheme.

per time step is defined as

$$\frac{1}{dt}\max\left[\frac{\| u^{n+1}-u^n \|}{q^{n+1}},\ \frac{\| v^{n+1}-v^n \|}{q^{n+1}},\ \frac{\| p^{n+1}-p^n \|}{\| p^n \| + (q^{n+1})^2/2}\right],$$

where $q = \max(\| u \|,\ \| v \|)$, and the maximum norm is used. In the left part of the figure, we observe unphysical wiggles near the outflow boundary, and the convergence history shows that the solution hesitates to become really stationary. The cause of the wiggles and the remedy have been discussed in Sect. 5.2. For Re = 200 a stationary solution was not obtained. Furthermore, it was found that with extrapolated Picard linearization, convergence to a steady solution did not take place.

Further results on the backward facing step problem

From now on, we use extrapolated Picard linearization. With the correct outflow boundary conditions (5.8) the result shown in Fig. 5.8 is obtained. This

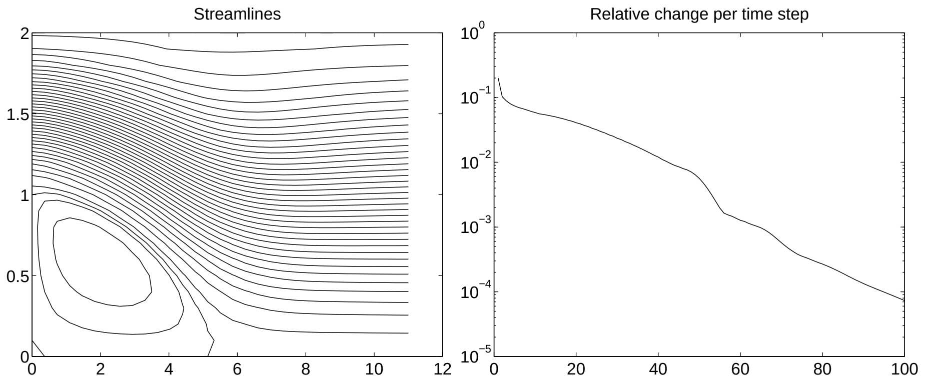

Fig. 5.8. Streamlines and convergence history for backward-facing step problem, code `ns1`, outflow conditions(5.8) . Re $= 100, \tau = 0.6, t = 60, nx = 30, ny = 20, \omega =$ 1, central scheme.

result looks more satisfactory. The length of the recirculation region agrees with results reported in the literature, and the solution seems to evolve to steady state. Due to the use of Picard linearization, `ns1` it is not unconditionally stable; we know no guidelines for choosing the time step τ. For linear problems, the ω-scheme with $\omega = 1$ is unconditionally stable, as seen in Sect. 4.4.

Fig. 5.9 shows what happens when the Reynolds number is increased. The length of the recirculation region increases; it is thought to be proportional

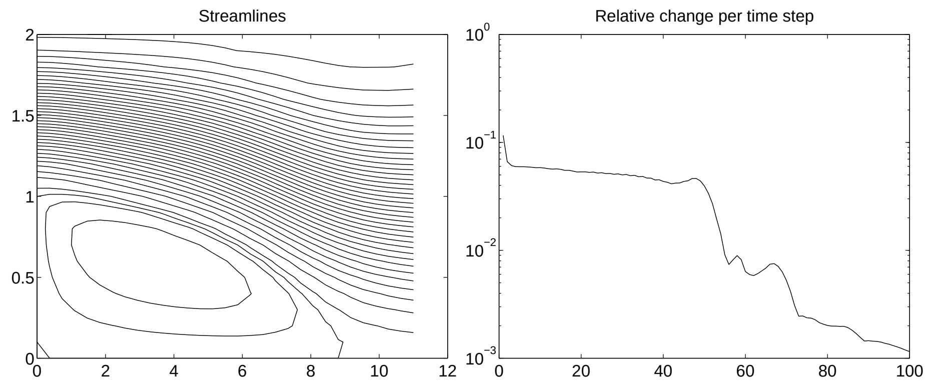

Fig. 5.9. Streamlines and convergence history for backward-facing step problem, code `ns1`. Re $= 200$, $\tau = 0.6, t = 60, nx = 30, ny = 20, \omega = 1$, central scheme.

to the Reynolds number. This flow we have also computed with the Adams-Bashforth-ω scheme (code `ns2`). The flow pattern obtained is the same as in Fig. 5.9. To determine the time step τ we have used the stability criterion for the Adams-Bashforth-Crank-Nicolson scheme (hence, $\omega = 1/2$) presented in Sect. 4.4, taking for safety τ 20% smaller than allowed, giving $\tau = 0.021$. The convergence history is shown in Fig. 5.10. Although the time required

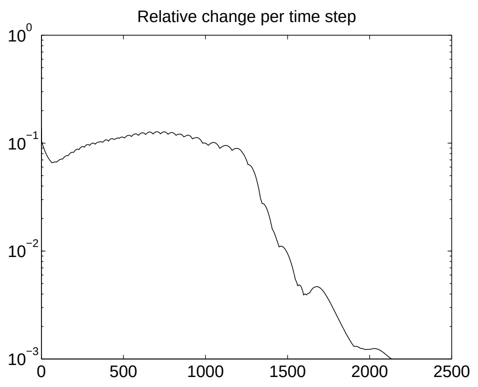

Fig. 5.10. Convergence history for backward-facing step problem, code `ns2`.

to execute a time step with `ns1` is seven times larger than for `ns2` (why is it larger, you think?), `ns1` is faster because its time step is much larger. The recirculation length in Fig. 5.9 agrees with reports in the literature. With the upwind scheme the recirculation length is found to 7, which is too small. This is because the upwind scheme adds artificial viscosity, and the recirculation length is known to increase with decreasing viscosity.

The case Re = 400, shown in Fig. 5.11 is found to be more difficult. By trial

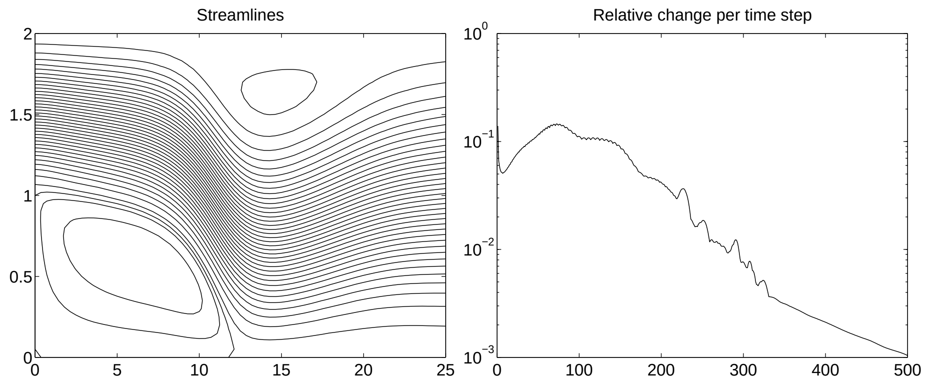

Fig. 5.11. Streamlines and convergence history for backward-facing step problem, code `ns1`. Re = 400, $\tau = 0.3, t = 150, nx = 70, ny = 40, \omega = 1$, central scheme.

and error we found that several changes have to be made in order to have the solution converge to steady state:

1) The length L has to be increased to make room for the (larger) separation zone and the secondary separation zone that appears at the top wall.
2) The vertical number of cells ny has to be increased. But it is found that for stability of `ns1` the mesh size ratio $\Delta x/\Delta y$ should not become too large. Therefore nx also had to be increased.
3) The time step has to be decreased.
4) The flow takes much longer to settle down to steady state.

As a consequence, Fig. 5.11 takes much more computing time than Fig. 5.9. Again, the recirculation length agrees with the literature.

Several other flow problems have been implemented in the programs `ns1` and `ns2`, and the reader is invited to experiment with these codes.

Efficiency

All numerical results in these course notes have been obtained with MATLAB version 5.3 on a Pentium III 550 Mhz processor with 512 MB internal memory. The computation of Fig. 5.11 is the first time that the wall-clock time becomes long enough (10 min) to be annoying. So now we start to get interested in the subject of *numerical efficiency*. In programming `ns1` and `ns1` we have followed the MATLAB efficiency guidelines discussed in Sect. 4.2. In order to see where the computational bottlenecks are, we make an inventory of the time spent in various parts of the program.

In `ns1` and `ns2` all linear systems (of type $Ay = b$) are solved by MATLAB's direct solver, with the statement

```
y = A\b
```

This means that the LU-decomposition of A is computed, and y is found from

```
y = U\(L\b)
```

The pressure-correction matrix does not change during time-stepping, so its LU-decomposition is computed once and stored. The viscous matrix also does not change, because the viscosity is constant. This fact is exploited in code `ns2` by means of the IMEX method, in which we have to solve systems with the viscous matrix; so its LU-decomposition is pre-stored as well in `ns2`. Of course, we store the matrices as sparse matrices, so that MATLAB exploits sparsity in the execution of `y = A\b`. What consumes most time is the work that has to be done every time step. This time is dominated by the time for solving the linear systems for the velocity components t_v and the pressure t_p, by the time for matrix generation (`inertia_matrix` in `ns1`) t_m, and by the time for right-hand side generation (`right_hand_side` in `ns2`) t_r. The total time is called t_t. Table 5.1 gives some runtime statistics for the two cases presented in Figs. 5.9, 5.10 (called case 1 in the table) and 5.11 (case 2).

	`ns1` case 1	`ns1` case 2	`ns2` case 1
t_m	0.039	0.184	—
t_v	0.113	0.896	0.009
t_p	0.007	0.060	0.007
t_r	—	—	0.006
t_t	19.2	600	60.1

Table 5.1. Runtime statistics (seconds).

Let us analyze these figures a bit. The number of unknowns (u, v or p) in x-direction is n_x+m and in y-direction n_y+m with $m = -1,\ 0$ or 1, depending on the type of unknown and the type of boundary conditions; of course, for estimating computing work, m can be neglected. This gives us matrices of size $n \times n,\ n = n_x n_y$, with bandwidth, with lexicographic ordering, equal to $2n_x - 1$. It is known that the work to compute the LU-decomposition W_{LU} and the work to solve a system W_S with this type of bandmatrix, using standard nethods, are given by:

$$W_{LU} \cong 2n_x^3 n_y \text{flops}, \quad W_S \cong 2n_x^2 n_y \text{flops}. \tag{5.32}$$

(A flop is a floating point operation). We therefore expect

$$t_v \sim n_x^3 n_y, \quad t_p \sim n_x^2 n_y, \quad t_m \sim n_x n_y \ .$$

Between cases 1 and 2 this equation predicts ratios of 25, 11 and 4.7 for t_v, t_p and t_m, respectively. Obviously, this prediction is much too pessimistic for t_v, but is not far off the mark for t_p and t_m. The reason MATLAB has a W_{LU} that in our case is much less than given by (5.32) is, that in execution of the command `y = A\b`, if A has been declared to be sparse, the equations are reorderded in a clever way, before the triangular factors L and U are computed, in order to reduce computing work. That W_{LU} may be diminished by reordering is obvious from equation (5.32): by a different numbering of the cells we can interchange the roles of n_x and n_y (cf. exercise 5.5.2), which in the present case changes the ratio just predicted for t_v from 25 to 19. We see that MATLAB does an even better job. Much numerical expertise is hidden behind the \ command in MATLAB.

The table shows that for an increase of $n \equiv n_x n_y$ by a factor 4.7, the time required by `ns1` for a time step increases by a factor 6.8. The total time t_t increases by a factor 31. This is mainly due to the much larger number of time steps required. We conclude that for high Reynolds number flows, time stepping is an inefficient way to compute a stationary solution.

Table 5.1 shows that for case 1 the computing time for a time step with `ns2` is a factor 7 smaller than with `ns1`. But because for stability reasons the time step is much smaller, the total computing time t_t is much larger.

In the next chapter we study other methods to compute stationary solutions, that are hopefully more efficient.

The reader is invited to consult the introduction of this chapter for the topics that we wanted to discuss.

Exercise 5.5.1. In Sect. 3.3 we saw that a Dirichlet outflow condition generates an artificial boundary layer at the outflow boundary. How does the thickness of the boundary layer caused by the wrong outflow condition (5.31) depend on the Reynolds number Re? In Sect. 2.3 we saw that outflow wiggles caused by an outflow Dirichlet condition go away when we apply mesh refinement near the outflow boundary, such the the mesh Péclet number $p < 2$. Try this for the problem of Fig. 5.7. Define and estimate the mesh Reynolds number, based on the maximum u, which occurs at the inflow boundary. How small should the local mesh size Δx be in the refinement zone? Implement a refinement zone in `ns1` by dividing the horizontal domain boundaries in two segments, similar to what is done in `ns1` for the vertical boundaries. See what happens.

Exercise 5.5.2. In the computation of Fig. 5.11, $n_x > n_y$. Equation (5.32) shows that it would be better if this were the other way around. Specify a vertical backward facing step problem in code `ns1`, and compare runtimes with the horizontal version.

Some self-test questions

Write down the free surface conditions.

What are your preferred outflow conditions, and why?

Discretize the x-momentum equation on a uniform Cartesian grid.

Describe the Adams-Bashforth-Crank-Nicolson scheme.

Write down the general formulation of the discretized nonstationary incompressible Navier-Stokes equations.

Formulate the pressure-correction method.

What is the backward-facing step problem?

Why is the computing time for a time step with the code `ns2` smaller than with `ns1`?

6. Iterative solution methods

6.1 Introduction

As we saw in the preceding chapter, solving the stationary Navier-Stokes equations by time-stepping with the nonstationary Navier-Stokes equations may be inefficient, due to the large number of time steps that may be required. Therefore we take a look in this chapter at numerical methods to solve the stationary Navier-Stokes equations without using time-stepping. For this efficient numerical methods to solve linear algebraic systems

$$Ay = b, \quad A \in \mathbb{R}^{n \times n}\ , \tag{6.1}$$

(which means that A is an $n \times n$ matrix) are required. Of course, such methods are useful for the nonstationary case as well; the methods discussed in the preceding chapter require solution of the pressure-correction equation and systems for the velocity prediction.

Efficient solution of equation (6.1) is one of the main topics in numerical linear algebra. We will not give a balanced coverage, but concentrate on methods relevant for the numerical solution of the incompressible Navier-Stokes equations, and restrict ourselves to an elementary introduction. The reader is supposed to be familiar with the basics of numerical linear algebra, as described for instance in van Kan and Segal (1993) (Chapt. 11), and, in great detail, in the well-known elementary numerical analysis textbook Burden and Faires (2001). A more advanced good standard textbook on numerical linear algebra is Golub and Van Loan (1989). A good advanced textbook on iterative methods is Hackbusch (1994). Chapt. 7 of Wesseling (2001) contains a more complete introduction to iterative methods for the incompressible Navier-Stokes equations than given here. Multigrid methods are discussed in Wesseling (1992), freely available on Internet, at `www.mgnet.org/mgnet-books-wesseling.html`, and more extensively in Trottenberg, Oosterlee, and Schüller (2001) (students will not be examined about material for which we refer to the literature).

Much high quality standard software to solve equation (6.1) is available on the Internet. Try for instance `math.nist.gov` or `www.netlib.org`

In many institutes the following FORTRAN software libraries are available: LAPACK, NAG, IMSL, ITPACK.

Some basics

For ease of reference we list a few basic facts and definitions, with which the reader is assumed to be familiar; more background can be found in the books cited above. Since we are now discussing linear algebra, by a vector we will mean not a physical vector but an algebraic vector:

$$y \in \mathbb{R}^n \quad \Longleftrightarrow \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \quad y_i \in \mathbb{R} \,,$$

where $\mathbb{R}^n$ is the vector space of real n-vectors. This is a column vector. By y^T we mean the corresponding row vector, so that $x^T y$ is the inner product $\sum_k x_k y_k$. By A^T we mean the transpose of A: if $B = A^T$, then $b_{ij} = a_{ji}$. This means that we interchange rows and columns.

Examples of vector norms are the *p-norms*:

$$\|y\|_p \equiv (|y_1|^p + \cdots + |y_n|^p)^{1/p}, \quad \|y\|_\infty \equiv \max\{|y_1|, \cdots, |y_n|\} \,.$$

When we write $\|y\|$, we leave the choice of norm open. As matrix norm we always use the norm induced by the vector norm:

$$\|A\| \equiv \sup_{y \neq 0} \frac{\|Ay\|}{\|y\|} \,,$$

so that

$$\|Ay\| \leq \|A\| \, \|y\| \,.$$

We say that the real or complex number λ is an eigenvalue and y an eigenvector of A if $Ay = \lambda y$. The spectral radius $\rho(A)$ of A is defined by

$$\rho(A) \equiv \max\{|\lambda(A)|\} \,.$$

We have[1]

$$\lim_{m \to \infty} \|A^m\|^{1/m} = \rho(A) \,. \tag{6.2}$$

The *condition number* of a matrix A is defined by

$$\mathrm{cond}(A) \equiv \frac{\max|\lambda(A)|}{\min|\lambda(A)|} \,.$$

[1] See Theorem 2.9.8 in Hackbusch (1994)

Extraction of the main diagonal is denoted by diag(A); this gives a diagonal matrix with elements a_{ii}.

A matrix $A \in \mathbb{R}^{n\times n}$ is called *diagonally dominant* if

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|, \quad i = 1 \cdots n \; .$$

A matrix is called *positive definite* if $x^T Ax > 0$ for all nonzero $x \in \mathbb{R}^n$. We say that A has *upper bandwidth* q if $a_{ij} = 0$ when $j > i + q$ and *lower bandwidth* p if $a_{ij} = 0$ when $i > j + p$; the *bandwidth* is $p + q - 1$. If most elements of A are zero, A is called *sparse*. If $a_{ij} = 0$ for all $j > i, \forall i$, A is called *lower triangular*; if $a_{ij} = 0$ for all $i > j, \forall j$, A is called *upper triangular*.

By $A > 0$ we mean $a_{ij} > 0,\ \forall i, j$.

A matrix is called an *M-matrix* if A is nonsingular, $A^{-1} \geq 0$ and $a_{ij} \leq 0,\ \ i \neq j,\ i, j = 1, \cdots, n$.

A matrix A is called a *K-matrix* if

$$\begin{aligned} a_{ii} &> 0, \quad i = 1, ..., n \; , \\ a_{ij} &\leq 0, \quad i, j = 1, ..., n, \quad j \neq i \; , \\ \text{and} \sum_j a_{ij} &\geq 0, \quad i = 1, ..., n \; , \end{aligned}$$

with strict inequality for at least one i. A matrix is called *irreducible* if the corresponding system does not consist of subsystems that are independent of each other. An irreducible K-matrix is an M-matrix[2], but not all M-matrices are K-matrices. Note that it is easy to check by inspection whether a matrix is a K-matrix, but the M-matrix property is more difficult to verify directly, because the elements of the inverse are not easily available, so that the condition $A^{-1} \geq 0$ is hard to check.

The above concepts will be used in what follows.

Purpose of this chapter

The purpose of this chapter is to discuss numerical methods for large sparse linear systems. We will study:

[2] See Theorem 7.2.5 in Wesseling (2001)

• Direct methods, and why we need iterative methods;

• Basic iterative methods: convergence, regular splittings;

• The Jacobi, Gauss-Seidel and ILU methods;

• Slow convergence of basic iterative methods for algebraic systems arising from elliptic partial differential equations;

• Iterative methods for the Navier-Stokes equations: SIMPLE, distributive iteration.

6.2 Direct methods for sparse systems

Gaussian elimination

Direct methods give the exact answer in a finite number of operations (in the hypothetical situation that no rounding errors are made). The prototype of a direct method is the elimination method of Gauss. In its modern version a lower and upper triangular matrix L and U are computed, such that

$$LU = A \ . \tag{6.3}$$

For L and U to exist, and/or to control rounding errors, it may be necessary to reorder the rows of A, *i.e.* to change the order of equations (this is often called *pivoting*). It is known[3] that pivoting is not necessary if A is diagonally dominant or positive definite. K-matrices are diagonally dominant. When L and U are available, the solution is easily obtained by solving by means of back-substitution, *i.e.* by solving

$$L\tilde{y} = b, \quad Uy = \tilde{y} \ . \tag{6.4}$$

Efficiency of direct methods for sparse systems

In the context of computational fluid dynamics, A is invariably very sparse. The matrices ocurring in the methods described in the preceding chapter have only at most five nonzero elements per row, namely $a_{i,i-nx},\ a_{i,i-1},\ a_{ii},\ a_{i,i+1}$ and $a_{i,i+nx}$. We see that the lower and upper bandwidth is nx. It turns out that L and U inherit the lower and upper bandwidth p and q, respectively, of A. If $p \ll n,\ q \ll n$, construction of L and U takes about $2npq$ flops *if no reordering is applied*; solution of $Ly = b$ takes about $2np$ flops and

[3] See sections 3.4.10 and 4.2 of Golub and Van Loan (1989).

solution of $Uy = b$ takes about $2nq$ flops.[4] Let us take as an example the 620×620 velocity matrix for the computation of Fig. 5.9 The MATLAB command `spy` gives the nonzero pattern shown in the left part of Fig. 6.1, and reports that only 2920 elements are nonzero, that is only 7.6% of the

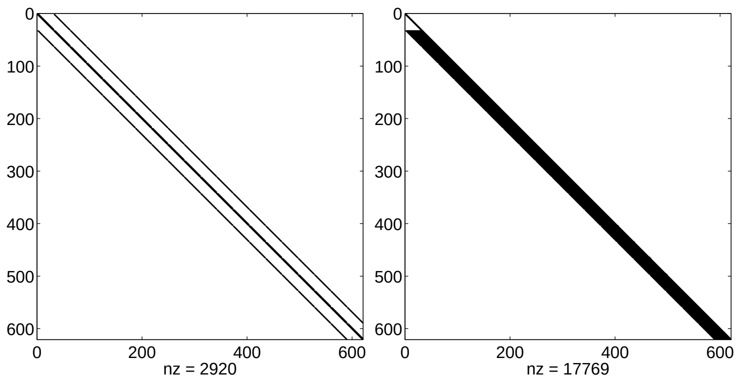

Fig. 6.1. Nonzero pattern of velocity matrix (left) and its L factor (right).

total number of elements. The number of nonzeros is slightly smaller than the $5 \times 620 = 3100$ which we just predicted, because the boundary conditions introduce a small number of extra zeros. The factors L and U may be obtained in MATLAB by the statement `[L,U] = lu(A)` The nonzero pattern of L is shown in the right part of Fig. 6.1, and has 17769 nonzeros, much more than the original A, and the same is true for U. The zeros inside the band have almost all disappeared. This is called *fill-in*. Fill-in may be diminished by clever reordering of the equations. A simple example was given in Sect. 5.5: equation (5.32) shows that when $n_x > n_y$ it pays off to interchange the x- and y- axes, so that $n_x := n_y,\ n_y := n_x$. MATLAB uses even more efficient reorderings in execution of `y=A\b` when A is sparse, but nevertheless, fill-in remains substantial. This leads to a waste of memory, which is intolerable in large applications, especially in three dimensions. Furthermore, computing time grows rapidly with the number of unknowns n. Taking $nx = ny = \sqrt{n}$ for simplicity, with n the number of unknowns, the number of flops W_{LU} for computing L and U, and the number of flops W_S for solving $LUy = b$ is approximately, using the formulas above,

$$W_{LU} = 2n^2, \quad W_S = 4n^{3/2} \ . \tag{6.5}$$

So both memory and work are superlinear in n. Since we have only n unknowns and A has only $5n$ nonzeros, an algorithm that requires only $\mathcal{O}(n)$ memory and work seems not to be impossible. Iterative methods indeed require only $\mathcal{O}(n)$ memory, whereas the work is $\mathcal{O}(n^\alpha)$, with α depending on the

[4] See sections 4.3.1 and 4.3.2 of Golub and Van Loan (1989).

method. Multigrid methods achieve the ideal value $\alpha = 1$. For these efficiency reasons, direct methods are seldom used in practical CFD. The remainder of this chapter is devoted to iterative methods. But first we will investigate what can be done with direct solution of the stationary Navier-Stokes equations.

Direct solution of the stationary Navier-Stokes equations

In the stationary case, the discretized form of the Navier–Stokes equations follows from equation (5.16) as

$$N(w) + Gp = f, \quad Dw = g \ ,$$

where the algebraic vector that contains all velocity unknowns is now called w. In order to apply a direct method for linear algebraic systems, we have to linearize $N(w)$ around an approximation w^{k-1} known from a preceding iteration. Let us use Picard linearization, as described in Sect. 5.4, and again in Sect. 6.4. This results in

$$N(w) \cong C^{k-1} w \ ,$$

where C^{k-1} is a matrix that depends on w^{k-1}. The following iterative method is obtained (Picard iteration):

Step 1. $k = 0$; choose w^k.

Step 2. Generate C^k and solve the following linear system:

$$\begin{bmatrix} C^k & G \\ D & 0 \end{bmatrix} \begin{bmatrix} w^{k+1} \\ p^{k+1} \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \ . \tag{6.6}$$

Step 3. $k = k + 1$; go back to step 2; repeat until convergence criterion is satisfied.

Although we use (Picard) iteration, we will call this a direct method, because equation 6.6 is solved by a direct method. The method has been implemented in the MATLAB code `ns3`. In this code the algebraic vector w is partioned as

$$w = \begin{bmatrix} u \\ v \end{bmatrix} \ ,$$

where u and v contain the horizontal and vertical velocity components. The partitioned system to be solved in step 2 now takes the following form:

$$\begin{bmatrix} C_u^k & 0 & G_u \\ 0 & C_v^k & G_v \\ D_u & D_v & 0 \end{bmatrix} \begin{bmatrix} u^{k+1} \\ v^{k+1} \\ p^{k+1} \end{bmatrix} = \begin{bmatrix} f_u \\ f_v \\ g \end{bmatrix} \ . \tag{6.7}$$

This system is easily solved directly in MATLAB by the statement `y=A\b`, with appropriate definition of A, b and y.

Driven cavity flow

We now apply the above direct method to the flow in a driven cavity. The domain is the unit square surrounded by solid walls. The top wall is sliding to the right with a velocity of magnitude 1. Fig. 6.2 shows the sparsity pattern

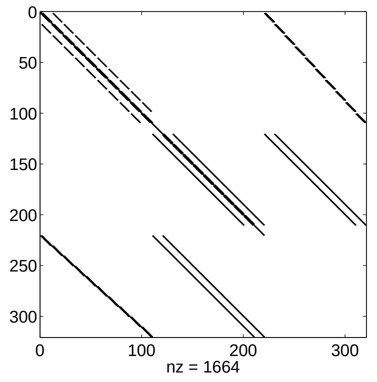

Fig. 6.2. Sparsity pattern of matrix in equation (6.7), 10×10 grid.

of the matrix in equation (6.6). The 3×3 block partioning of the matrix in equation (6.7) is clearly visible. Fig. 6.3 shows a result. The computational

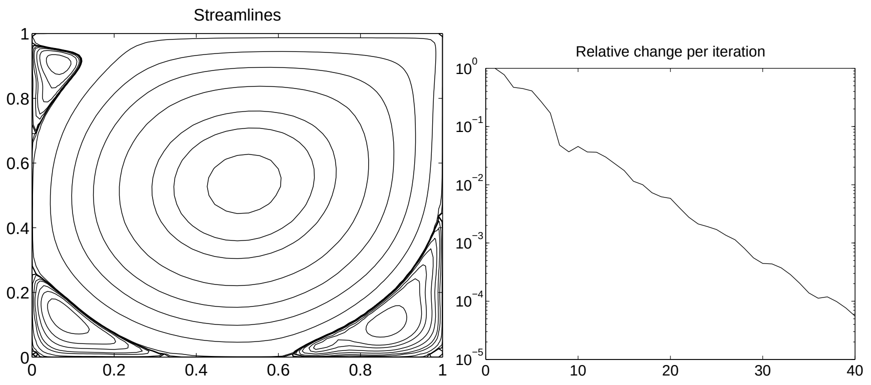

Fig. 6.3. Streamlines and convergence history for driven cavity problem, code `ns3`. Re $= 5000$, $nx = ny = 64$, nonuniform grid, central scheme.

grid is Cartesian and has $nx \times ny$ cells. The grid is nonuniform for better resolution in the corners. Along the x-axis the unit interval is divided in segments of length 0.2, 0.6 and 0.2, which are subdivided in 21, 22 and 21 cells, respectively; and similarly in the y-direction. So $nx = ny = 64$. The size of the internal memory of my PC (512MB) did not allow larger values of nx and ny without swapping to harddisk. We will see later whether iterative methods are less greedy for memory. Fig. 6.4 shows a similar but coarser grid

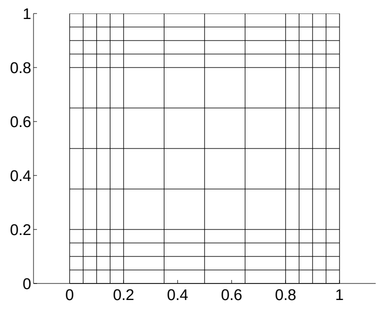

Fig. 6.4. A nonuniform Cartesian grid.

for illustration.

We discretize on this nonuniform grid in the way described in Sect. 2.3, using the central scheme (2.20, 2.29). It was shown in Sect. 2.3 that although this scheme has a large local truncation error at grid discontinuities, nevertheless the global truncation error remains small. This is confirmed by the result of Fig. 6.3: the streamline pattern does not show any trace of the grid discontinuities. Furthermore, on a uniform grid with the same number of cells almost the same result is obtained, but with less resolution in the corners.

Results of benchmark quality for this driven cavity flow are given in Ghia, Ghia, and Shin (1982), using a uniform 257×257 grid. The streamline pattern of Fig. 6.3 closely resembles that of Ghia, Ghia, and Shin (1982), but our minimum streamfunction value (-0.1138) is not as low as the benchmark value -0.118966; this is because our grid is not fine enough.

Fig. 6.5 shows the dependence of the time measured to solve equation (6.6) on the number of unknowns $n = nx \times ny$, with $nx = ny$. The dashed line has slope 2. So computing time seems proportional to n^2; we will *not* try to show this by analysis. Note that formula (6.5) applies .approximately.

The total computing time for the flow of Fig. 6.3 was 2107 sec. Would the time-stepping methods of the preceding chapter be faster? We leave this to

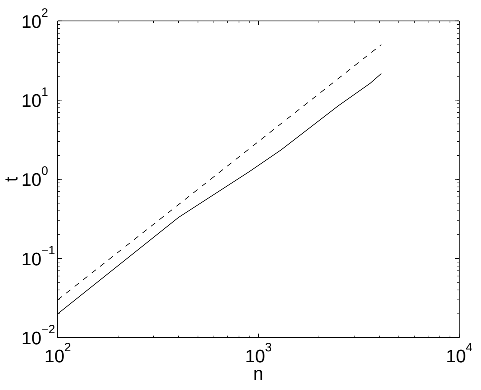

Fig. 6.5. Sparse direct solver time versus number of unknowns, driven cavity problem (solid curve).

the reader to find out.

Exercise 6.2.1. What is the bandwidth of the matrix in equation (6.7)? Why does formula (6.5) not apply?

Exercise 6.2.2. Compute the minimum streamfunction value for the driven cavity flow problem using uniform grids with 16×16, 32×32 and 64×64 cells, with central differencing and Re $= 2000$. How does the difference with the "exact" (benchmark) value for the streamfunction minimum depend on the mesh size $h = 1/nx = 1/ny$? Try to determine α such that h^α gives a good fit. Why would one expect $\alpha = 2$? How is the result influenced by the termination criterion (*i.e.* the relative change per iteration)?

Exercise 6.2.3. Compute the flow of Fig. 6.3 with the upwind scheme. Compare accuracy.

6.3 Basic iterative methods

A good source (in Dutch) for more background on iterative methods for large linear systems is: C. Vuik: Voortgezette numerieke lineaire algebra, course notes for WI4 010: Numerieke methoden voor grote lineaire stelsels.

An iterative method generates successive approximations $y^1, y^2, \cdots$. If

$$\lim_{k\to\infty} y^k = y \ ,$$

the method is said to converge. Although in general it takes an infinite number of iterations to obtain the solution y, this does not necessarily imply that iterative methods are less efficient than direct methods, because only a sufficiently accurate approximation to y is required. This is because y itself is an approximation, which differs by the global truncation error from the exact solution of the partial differential equation, of which the system to be solved $Ay = b$ is a discrete approximation. Hence, a finite number of iterations suffices.

Let us consider only *stationary iterative methods*; that is, methods in which the same algorithm is applied in every iteration. Let, furthermore, the computation of the new iterand y^k involve only matrix multiplication and vector addition:

$$y^k = By^{k-1} + c \, , \tag{6.8}$$

where the matrix B and the vector c remain to be chosen. For $\{y^k\}$ to converge to the solution $y = A^{-1}b$ it is obviously necessary that y is a stationary point of the iteration process (6.8), *i.e.* $y = By + c$, hence we must have $c = (I - B)A^{-1}b$, so that (6.8) can be rewritten as

$$My^k = Ny^{k-1} + b, \quad M - N = A \, . \tag{6.9}$$

(We have $M - N = A$ because $M = A(I - B)^{-1}, \; N = MB$). Equation (6.9) can be rewritten in the following equally useful form:

$$M\delta y = b - Ay^{k-1}, \quad y^k = y^{k-1} + \delta y. \tag{6.10}$$

Application of *underrelaxation* or *overrelaxation* with *relaxation parameter* α means that (6.10) is replaced by

$$M\delta y = \alpha(b - Ay^{k-1}), \quad y^k = y^{k-1} + \delta y. \tag{6.11}$$

We will call methods of type (6.9) or (6.10) or (6.11) *basic iterative methods*, or BIMs for brevity.

Instead of B one usually specifies M. One can think of M as an approximation of A; if $M = A$ and $\alpha = 1$ we have convergence in one iteration. We have (the reader should check this)

$$B = I - M^{-1}A \, .$$

The error $e^k \equiv y - y^k$ satisfies $e^k = Be^{k-1}$, so that convergence is governed by B, which is called the *iteration matrix*. Since $e^k = B^k e^0$ (where of course the superscript k on B is not an index but an exponent),

$$\|e^k\| \leq \|B^k\| \, \|e^0\| \, . \tag{6.12}$$

From equation (6.2) it follows that we have convergence ($\|e^k\| \to 0$) if and only if

$$\rho(B) < 1 \, . \tag{6.13}$$

When to stop?

For efficiency, which is what iterative methods are all about, it is necessary to stop iterating as soon as sufficient precision has been obtained. Hence, we need a good *termination criterion.* The difference between two successive iterates usually does *not* give a good indication of the precision achieved. This can be seen as follows. Let B have a single dominating real eigenvalue $\lambda = \rho(B)$, with corresponding eigenvector x. Then we have after many iterations $e^k \cong \alpha\lambda^k x$, with α some constant. It follows that

$$y^{k+1} - y^k \cong \alpha\lambda^k(\lambda - 1)x = (\lambda - 1)e^k \ ,$$

so that

$$e^k \cong \frac{y^{k+1} - y^k}{\lambda - 1} \ .$$

We see that if $\lambda = 1-\epsilon,\ 0 < \epsilon \ll 1$, as frequently happens (slow convergence), then the error is much larger than the difference between two successive iterates. Therefore it is better to derive a termination criterion from the residual $r^k \equiv b - Ay^k$. After k iterations we have solved exactly the following approximate problem:

$$Ay^k = b - r^k \ .$$

Frequently, perturbation of the right-hand side has a physical interpretation, or b is known to have an error with a known bound, which enables one to decide what size of $\|r^k\|$ is tolerable.

Why it is nice if A is a K-matrix

In sections 2.3 and 3.3 we saw that if the scheme L_h is of positive type, then it satisfies a maximum principle, so that unphysical oscillations (wiggles) are excluded. It is easy to see that if L_h is of positive type, then the corresponding matrix is a K-matrix. The second reason why it is nice if A is a K-matrix is, that it is easy to devise convergent BIMs. This is shown below.

Regular splittings

We just saw that a BIM is defined by a specification of an approximation M of A. How to choose M? Obviously, M must be such that equation (6.9) can be solved efficiently. Furthermore, the method must be convergent: $\rho(B) = \rho(I - M^{-1}N) < 1$. Finally, the BIM must converge rapidly. We will now see that it is easy to devise convergent BIMs if A is an M-matrix, for example a K-matrix. The speed of convergence will be examined later.

Definition 6.3.1. The splitting $A = M - N$ is called *regular* if $M^{-1} \geq 0$ and $N \geq 0$. The splitting is called *convergent* if (6.9) converges.

We have the following nice theorem:

Theorem 6.3.1. *A regular splitting of an M-matrix is convergent.*

Proof. See Theorem 3.13 in Varga (1962). □

Hence, we aim for regular splittings. The following theorem shows that regular splittings are easy to obtain for M-matrices:

Theorem 6.3.2. *Let A be an M-matrix. If M is obtained by replacing some elements a_{ij}, $j \neq i$ by b_{ij} satisfying $a_{ij} \leq b_{ij} \leq 0$, then $A = M - N$ is a regular splitting.*

Proof. See Theorem 7.2.6 in Wesseling (2001). □

This means, for example, that if A is a K-matrix, and if we make M by replacing arbitrary off-diagonal elements a_{ij}, $i \neq j$ by 0, then we have a regular splitting (note that for if A is a K-matrix, then $a_{ij} \leq 0$, $i \neq j$, so that M will be a K-matrix, and $N \geq 0$). This gives us great freedom in designing regular splittings.

Examples

Jacobi

The *Jacobi* method is obtained by taking $M = D \equiv \text{diag}(A)$. This gives the following BIM:

$$Dy^k = (D - A)y^{k-1} + b \; .$$

Gauss-Seidel

The *Gauss-Seidel* method is obtained as follows. Write $A = D - E - F$, with $D = \text{diag}(A)$, E strictly lower triangular (*i.e.* $E_{ij} = 0$, $j \geq i$) and F strictly upper triangular (E is simply the strictly lower triangular part of $-A$, and F the upper part of $-A$). We take $M = D - E$ (forward Gauss-Seidel) or $M = D - F$ (backward Gauss-Seidel). With this M it is also easy to solve for y^k, because M is lower or upper triangular; y^k is obtained by back-substitution. For example, forward Gauss-Seidel can be written as

$$y_i^k = \left(b_i - \sum_{j=1}^{i-1} a_{ij} y_j^k - \sum_{j=i+1}^{n} a_{ij} y_j^{k-1} \right) / a_{ii} \; .$$

ILU

Unlike the preceding two methods, the incomplete LU (ILU) method is not obtained by replacing elements of A by zero. The idea is to determine lower and upper tridiagonal matrices L and U such that $LU \cong A$, and to choose $M = LU$. If $LU = A$ we get $N = 0$, and we are back at the direct (Gaussian elimination) method described before. But the difference is that the cause of the inefficiency is eliminated by forbidding fill-in; hence, $LU \neq A$, but the hope is that LU will not be far from A, so that we have rapid convergence. ILU decomposition works as follows. The elements of L and U are prescribed to be zero, except if $(i,j) \in Q$, where Q is the set of index pairs where we allow nonzeros:

$$Q \subset Q_n, \quad Q_n \equiv \{(i,j),\ 1 \leq i \leq n,\ 1 \leq j \leq n\}\ .$$

An example of a suitable set Q will follow. ILU is based on the following theorem:

Theorem 6.3.3. *Let A be an M-matrix. Then for every $Q \subset Q_n$ there exist a lower triangular matrix L with $l_{ii} = 1$, an upper triangular matrix U and a matrix $N \geq 0$ such that $A = LU - N$ and*

$$\begin{aligned} l_{ij} &= 0 \quad \textit{and} \quad u_{ij} = 0 \quad \textit{if} \quad (i,j) \notin Q\ , \\ n_{ij} &= 0 \quad \textit{if} \quad (i,j) \in Q\ . \end{aligned}$$

Furthermore, the iterations $LUy^k = b + Ny^{k-1}$ converge.

Proof. See Meijerink and van der Vorst (1977). □

Again we see that the M-matrix property is crucial.

We have

$$(LU)_{ij} = a_{ij}, \quad (i,j) \in Q\ ,$$

but we can have $(LU)_{ij} \neq a_{ij},\ (i,j) \notin Q$. The so-called first order ILU decomposition is obtained by choosing Q equal to the nonzero pattern of A. For instance, if the scheme L_h has the following five point stencil:

$$[L_h] = \begin{bmatrix} & * & \\ * & * & * \\ & * & \end{bmatrix}\ , \tag{6.14}$$

then the nonzero pattern Q_A of A is, with lexicographic ordering

$$Q_A = \{i,j\} \in Q_n,\ j = i \text{ or } j = i \pm 1 \text{ or } j = i \pm nx\ .$$

It comes as a nice surprise that the nonzero pattern of $N = LU - A$ turns out to be very small:

$$Q_N = \{i,j\} \in Q_n,\ j = i + nx - 1 \text{ or } j = i - nx + 1\ .$$

So N has only two nonzero diagonals. We hope the elements will be small, because the closer $M = LU$ is to A, the faster convergence will be.

Incomplete Cholesky decomposition

If a matrix A is symmetric and positive definite, then it has not only an LU decomposition, but also a Cholesky decomposition:

$$LL^T = A$$

with now $\mathrm{diag}(L) \neq I$. An advantage is that there is no need to store U. Similarly to ILU, there exist incomplete Cholesky decompositions (ICs). We now present an algorithm to compute an IC. Let the matrix A (symmetric, of course) be a discretization on an $n \times n$ or $n \times n \times n$ grid, so the size of A is either $n^2 \times n^2$ or $n^3 \times n^3$. It is convenient to temporarily write the IC decomposition as $LD^{-1}L^T$ with $\mathrm{diag}(L) = D$. We want to make a first order IC. This means that $l_{ij} \neq 0$ only if $a_{ij} \neq 0$. We choose $l_{ij} = a_{ij},\ i \neq j$. In the two-dimensional case, let A correspond to the stencil (6.14), so that the nonzero elements are $a_{ii}, a_{i,i\pm1}, a_{i,i\pm n}$. If we define

$$d_{ii} = a_{ii} - a_{i,i-1}^2/d_{i-1,i-1} - a_{i,i-n}^2/d_{i-n,i-n}\,, \tag{6.15}$$

where term with an index < 1 are to be deleted, then we have

$$(LD^{-1}L^T)_{ij} = a_{ij}, \quad \text{if } a_{ij} \neq 0\,. \tag{6.16}$$

It is left to the reader to verify this. Similarly, in three dimensions:

$$d_{ii} = a_{ii} - a_{i,i-1}^2/d_{i-1,i-1} - a_{i,i-n}^2/d_{i-n,i-n} - a_{i,i-n^2}^2/d_{i-n^2,i-n^2}\,. \tag{6.17}$$

Finally, we bring the IC in the form LL^T by defining

$$L := LD^{-1/2}\,,$$

where $D^{-1/2} = \mathrm{diag}(d^{-1/2})$. Equations (6.15) and (6.17) have been implemented, for the model problem described below, in the MATLAB M-file `my_cholinc`, because MATLAB's `cholinc` is inefficient.

A model problem

Let us consider the Poisson equation on the unit square with a homogeneous Dirichlet boundary condition:

$$-\varphi_{xx} - \varphi_{yy} = f(x,y), \quad (x,y) \in \Omega \equiv (0,1)\times(0,1), \quad \varphi|_{\partial\Omega} = 0\,. \tag{6.18}$$

We discretize on a homogeneous $(n+2)\times(n+2)$ vertex-centered grid such as shown in Fig. 6.6. We index only the interior points with $i,j = 1,\cdots,n$, so $n = 2$ in Fig. 6.6. Standard finite difference or finite volume discretization of (6.18) gives

$$L_h\varphi_{ij} \equiv 4\varphi_{ij} - \varphi_{i-1,j} - \varphi_{i+1,j} - \varphi_{i,j-1} - \varphi_{i,j+1} = h^2 f_{ij}, \quad i,j = 1,\cdots,n\,, \tag{6.19}$$

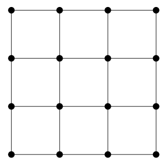

Fig. 6.6. Uniform vertex-centered grid

where $h = 1/(n+1)$. A term with an index that refers to the boundary (*e.g.* $i = 0$) can be neglected, because of the homogeneity of the Dirichlet boundary condition. The operator L_h has the five-point stencil defined in (6.14). We use lexicographic ordering. The smart way to generate the matrix A associated with the linear system (6.19) in MATLAB (see the program `po`) is as follows:

```
e = ones(n,1);
B = spdiags([-e 2*e -e],[-1 0 1],n,n); I = eye(n);
A = kron(I,B) + kron(B,I);
```

The `kron` function evaluates the *Kronecker product* of two matrices. The Kronecker product of an $n \times n$ matrix A and an $m \times m$ matrix B is of size $mn \times mn$ and can be expressed as a block $n \times n$ matrix with (i,j) block $a_{ij}B$.

ILU decompositions can be computed in MATLAB with `luinc` . The statement `[L,U,P] = luinc(A,'0');` corresponds to the first order ILU decomposition . The first order incomplete Cholesky decomposition in obtained in MATLAB with the statement `L = cholinc(A,'0')` The MATLAB command `spy(A)` gives a picture of the nonzero pattern of A. Fig. 6.7 shows the nonzero patterns of A and of the first order ILU factors L and U obtained. Furthermore, the nonzero pattern of $N = LU - A$ is shown; it does not quite look like the two-diagonal pattern Q_N announced above; this is due to rounding errors. The size of the nonzero elements outside the two diagonals is negligible. Furthermore, we find that $|n_{ij}| < 0.3$, so that in this case (incomplete) LU seems to be a pretty good approximation of A. Because in this example A is symmetric and positive definite, *incomplete Cholesky decompositions* exist as well.

The rate of convergence

The rate of convergence depends on the initial vector y^0; in the unlikely case that we start with the exact solution y, we discover after one iteration that the residual $r^1 = 0$, so we are finished in one iteration. We can only say something general about the rate of convergence of an iterative method in the asymptotic case where the number of iterations is very large. Equations

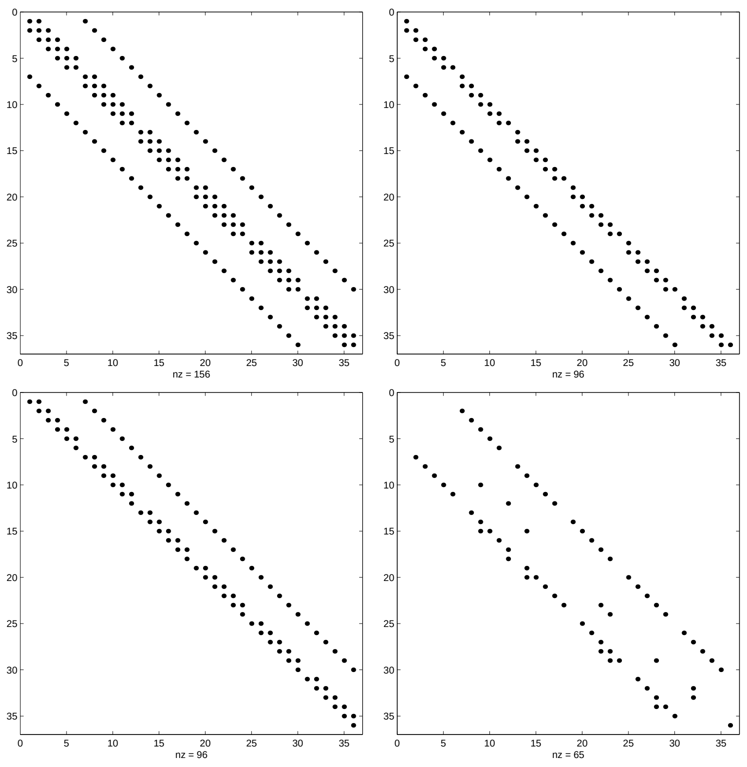

Fig. 6.7. Nonzero patterns of A, of ILU factors L and U, and of $N \equiv LU - A$; first order ILU decomposition.

(6.2) and (6.12) show that for $k \gg 1$ the error $e^k = y - y^k$ satisfies

$$\|e^k\| \leq \rho(B)\|e^{k-1}\|, \quad B = I - M^{-1}A \, . \tag{6.20}$$

So the quality measure of a BIM is its spectral radius $\rho(B)$: the smaller $\rho(B)$, the better (of course, the computing work per iteration should also be taken into account). However, in general, $\rho(B)$ is not easy to determine. We will do this for an illustrative example, namely the Poisson problem introduced above. Analysis is easier in the difference form than in the matrix form. The Jacobi method applied to the scheme (6.19) gives:

$$4\varphi_{ij}^{k+1} = \varphi_{i-1,j}^k + \varphi_{i+1,j}^k + \varphi_{i,j-1}^k + \varphi_{i,j+1}^k + h^2 f_{ij}, \quad i,j = 1, \cdots, n \, . \tag{6.21}$$

Here and in what follows terms with i or $j \notin \{1, \cdots, n\}$ are defined to be zero. The exact solution φ_{ij} satisfies (6.21). Subtraction gives for the error $e_{ij}^k = \varphi_{ij} - \varphi_{ij}^k$:

$$e^{k+1} = Be^k, \quad Be_{ij} \equiv (e_{i-1,j} + e_{i+1,j} + e_{i,j-1} + e_{i,j+1})/4 \, , \tag{6.22}$$

together with the boundary condition $e_{0,j} = e_{n+1,j} = e_{i,0} = e_{i,n+1} = 0$. Our predecessors have found out that the eigenfunctions of the operator B are:

$$\psi_{ij} = \sin iph\pi \sin jqh\pi, \quad h = 1/(n+1), \quad p, q \in \{1, \cdots, n\} , \tag{6.23}$$

which is easily verified by inspection: substitution of (6.23) in (6.22) gives

$$B\psi = \lambda\psi, \quad \lambda = \frac{1}{2}(\cos ph\pi + \cos qh\pi) , \tag{6.24}$$

where one needs the trigonometric identity

$$\sin\alpha + \sin\beta = 2\sin\frac{1}{2}(\alpha+\beta)\cos\frac{1}{2}(\alpha-\beta) .$$

Clearly, we have found n^2 independent eigenfunctions (or eigenvectors) ψ, since each pair $\{p, q\}$ admitted by (6.23) gives us an eigenfunction. Since B corresponds to an $n^2 \times n^2$ matrix, we have found all its eigenvectors; hence, (6.24) gives us all eigenvalues. The maximum $|\lambda|$ delivers the spectral radius:

$$\rho(B) = \cos\pi h . \tag{6.25}$$

Hence,

$$\rho(B) \cong 1 - \mathcal{O}(h^2) . \tag{6.26}$$

Equation (6.26) turns out to be true in general for BIMs applied to numerical schemes for second order elliptic partial differential equations on general grids with mesh size proportional to h.

How many iterations are required to have an error reduction of 10^{-d}, *i.e.* to gain d decimal digits? According to equations (6.12) and (6.2), we must have $\rho(B)^k < 10^{-d}$, hence $k\ln\rho(B) < -d\ln 10$, or

$$k > -d\ln 10/\ln\rho(B) .$$

We have (*cf.* Exercise 6.3.5)

$$\ln\rho(B) \cong -\frac{1}{2}(\pi h)^2 ,$$

hence the number of iterations required m satisfies

$$m \cong \frac{2d\ln 10}{\pi^2}N .$$

where $N \cong 1/h^2$ temporarily stands for the total number of unknowns. From (6.21) it follows that a Jacobi iteration takes $5N$ flops, so the total work $W_J = 5Nm$ satisfies

$$W_J = \mathcal{O}(N^2) . \tag{6.27}$$

As seen before, direct methods also have $W = \mathcal{O}(N^2)$, so the Jacobi method (and other BIMs) do not bring us closer to the ideal $W = \mathcal{O}(N)$, although Gauss-Seidel converges twice as fast than Jacobi, see Exercise 6.3.8 (this is not surprising, because Gauss-Seidel uses updated variables as soon as they become available). BIMs only bring a saving in computer memory.

Exercise 6.3.1. Derive equation (6.10) from equation (6.9).

Exercise 6.3.2. Express the iteration matrix B in terms of M and A in the case that relaxation with parameter α is applied.

Exercise 6.3.3. Verify equation (6.16).

Exercise 6.3.4. Verify equation (6.24).

Exercise 6.3.5. Show that if $|\varepsilon| \ll 1$

$$\ln \cos \varepsilon \cong -\frac{1}{2}\varepsilon^2 \ .$$

Exercise 6.3.6. Show that if A is an M-matrix, then Jacobi and Gauss-Seidel correspond to regular splittings. Why is this nice?

Exercise 6.3.7. Write down the Gauss-Seidel variants (forward and backward) of equation (6.21).

Exercise 6.3.8. It is known[5] that, for matrices generated by discretizations on the type of grid considered in these lecture notes, the spectral radii of the Gauss-Seidel and Jacobi methods are related by

$$\rho(B_{GS}) = \rho(B_J)^2 \ .$$

Show that Gauss-Seidel requires half as much computing work as Jacobi.

6.4 Krylov subspace methods

Acceleration of basic iterative methods

So why bother with BIMs? The answer is: BIMs are essential as indispensable building blocks inside more complicated but very efficient methods.

To begin with, BIMs can be accelerated by *Krylov subspace methods.* This means that we take a BIM, corresponding to a splitting $A = M - N$, and

[5] See Corollary 2.3 in Chapt. 5 in Young (1971)

use M to *precondition* the problem to be solved $Ay = b$, which means that $Ay = b$ is replaced by its preconditioned version:

$$M^{-1}Ay = M^{-1}b \ . \tag{6.28}$$

We call M a good *preconditioner* if (6.28) lends itself well for efficient solution by *Krylov subspace methods*. Therefore an iterative method is frequently called a preconditioner. One may say that convergence of the BIM is accelerated by a Krylov subspace method, or that convergence of the Krylov subspace method is accelerated by the BIM.

Secondly, BIMs can be accelerated by *multigrid methods*. We call a BIM a good *smoother* if the BIM makes the error rapidly *smooth* (although not rapidly small). Then the error can be made rapidly small with a *multigrid method*. In principle this approach leads to the ideal efficiency

$$\text{Work} = \mathcal{O}(N)$$

with N the total number of unknowns.

We have no time in this course to explain the principles of Krylov subspace and multigrid methods, but will show how Krylov subspace methods that have been implemented in MATLAB can be used.

Preconditioned conjugate gradients

The *conjugate gradients (CG) method* is a particular Krylov subspace method for linear systems $Ay = b$ with a symmetric positive definite matrix A. It is called in MATLAB by

```
  y = pcg(A,b,tol,maxit)
```

with `maxit` the maximum number of iterations allowed, and `tol` the relative error in the residual, defined by

$$\frac{\|Ay - b\|}{\|b\|} \ .$$

For information on Krylov subspace methods for nonsymmetric A that have been implemented in MATLAB, type: `help pcg` , and call `help` for the functions listed under "See also".

It is known[6] that the number of iterations required by CG is proportional to $\sqrt{\text{cond}}(A)$. To make cond(A) smaller, the system is preconditioned, as in

[6] This follows from Theorem 10.2.5 in Golub and Van Loan (1989)

equation (6.28). Because the preconditioned matrix must be symmetric in order to apply CG, we choose the preconditioned system as follows:

$$L^{-1}AL^{-T}\tilde{y} = L^{-1}b, \quad y = L^{-T}\tilde{y}\,, \tag{6.29}$$

where LL^T is an incomplete Cholesky decomposition of A, and $L^{-T} = (L^T)^{-1}$. If CG is applied with incomplete Cholesky preconditioning, the resulting method is called *ICCG* (incomplete Cholesky conjugate gradients).

We will now determine the condition number of A for our model problem (6.19). We have $A = 4(I - B)$, with B given in equation (6.22). If $p(x)$ is a polynomial, then $\lambda(p(A)) = p(\lambda(A))$. Hence, $\lambda(A) = 4 - 4\lambda(B)$, so that

$$\lambda(A) = 4\big(\sin^2 \tfrac{1}{2}p\pi h + \sin^2 \tfrac{1}{2}q\pi h\big), \quad p, q = 1, \cdots, n, \quad h = 1/(n+1)\,,$$

hence

$$\min|\lambda(A)| = 8\sin^2 \pi/2(n+1), \quad \max|\lambda(A)| = 8\sin^2 n\pi/2(n+1)\,,$$

For

$$\text{cond}(A) \equiv \frac{\max|\lambda(A)|}{\min|\lambda(A)|}$$

we find, taking $n = 50$:

$$\text{cond}(A) = 1.0535e + 03$$

We can also estimate cond(A) with MATLAB. By running `po` with `dim = 2` and `n = 50` and typing `condest(A)` we find:

$$\text{cond}(A) = 1.5315e + 03\,,$$

which is not very accurate but gives the right size. By typing

```
B=inv(L)*A*inv(L');
cond(B)
```

we find

$$\text{cond}(L^{-1}AL^{-T}) = 221\,.$$

This reduction in condition number that IC preconditioning brings is what makes ICCG a great succes; the reduction factor grows with n.

The following MATLAB code is an implementation of ICCG:

```
L = cholinc(A,'0');             % First order IC decomposition
y = pcg(A,b,1e-2,200,L',L);     % Preconditioned cg method
```

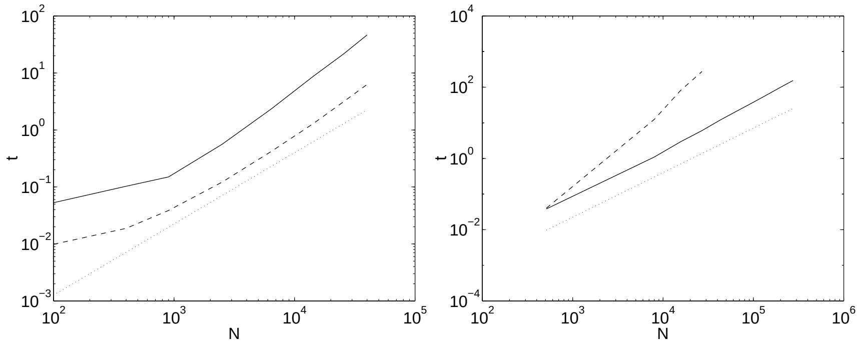

Fig. 6.8. ICCG computing time versus number of unknowns. Solid line: ICCG; - - -: direct solution. Left: two-dimensional case; right: three-dimensional case

Because the MATLAB function `cholinc` is very slow, we have made our own version, called `my_cholinc`, based on equations (6.15) and (6.17). We use `my_cholinc` in the program `po` . Fig. 6.8 (left) gives a plot of the computing time needed by ICCG and the MATLAB sparse direct solver for the problem (6.19). As initial guess for the solution, a random guess was supplied to `pcg` . The figure was created with the codes `po` and `wp_iccg` . We see that direct solution takes less computing time than ICCG, which is surprising. Apparently, the efficiency of the implementation of `pcg` in MATLAB leaves something to be desired, whereas direct sparse solution `y=A\b` works very efficiently, as we have noted before. Also `cholinc` is implemented inefficiently, since it takes more time than `y=A\b` . The slope of the dotted line is proportional to $N^{5/4}$, with N the number of unknowns. We see that for ICCG we have

$$\text{Work} = \mathcal{O}(N^{5/4})\,.$$

This is not very far from the ideal $\mathcal{O}(N)$.

But in three dimensions, where efficiency really counts, we get a different outcome. The three-dimensional version of our model problem (6.19) is:

$$\begin{aligned} L_h\varphi_{ijk} \equiv \quad & 6\varphi_{ijk} - \varphi_{i-1,jk} - \varphi_{i+1,jk} - \varphi_{i,j-1,k} - \varphi_{i,j+1,k} \\ & -\varphi_{i,j,k-1} - \varphi_{i,j,k+1} = h^2 f_{ijk}, \quad i,j,k = 1,\cdots,n\,. \end{aligned} \tag{6.30}$$

The smart way to generate the matrix A corresponding to this three-dimensional discrete Laplace operator in MATLAB is as follows (see the program `po`):

```
A = kron(I,kron(I,B)) + kron(I,kron(B,I)) + kron(B,kron(I,I));
```

with `I` and `B` as in the preceding section. Results for the computing time (obtained with the same codes as above) are shown in the right part of Fig. 6.8. Again, the time for ICCG is proportional to $N^{5/4}$. The MATLAB direct solver is now much less efficient. This illustrates that in three dimensions iterative methods are indispensable.

Exercise 6.4.1. Show that the matrix $L^{-1}AL^{-T}$ in equation (6.29) is symmetric positive definite if A is symmetric positive definite.

Exercise 6.4.2. Show that if $p(x)$ is a polynomial, then $\lambda(p(A)) = p(\lambda(A))$.

Exercise 6.4.3. Compare the efficiency of unpreconditioned and preconditioned CG by suitably modifying and running the program `po`

6.5 Distributive iteration

We will now apply basic iterative methods to the stationary Navier-Stokes equations. The discretized stationary Navier-Stokes equations are obtained from equation (5.16) in the following form:

$$N(u) + Gp = f, \quad Du = g \, , \tag{6.31}$$

with N a nonlinear algebraic operator. This nonlinear system has to be solved iteratively. We use Picard iteration, as decribed in Sect. 5.4. This works as follows. Let superscript k be the iteration counter. Then, for example, terms like u_{jk}^2 and $(uv)_{j+1/2,k+1/2}$ (with u and v now temporarily denoting the two velocity components) in the discretized x-momentum equation (5.11) are replaced by

$$u_{jk}^2 \cong (u^{k-1}u^k)_{jk}, \quad (uv)_{j+1/2,k+1/2} \cong (u^k v^{k-1})_{j+1/2,k+1/2} \, .$$

As a consequence (using u again to denote the algebraic vector containing all velocity unknowns), $N(u)$ gets replaced by $C^{k-1}u^k$, with C^{k-1} a matrix that depends on u^{k-1}. The system (6.31) now can be written as:

$$\begin{bmatrix} C^{k-1} & G \\ D & 0 \end{bmatrix} \begin{bmatrix} u^k \\ p^k \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \, . \tag{6.32}$$

Picard iteration works as follows:

Step 1. $k = 0$. Choose $u^0, \ p^0$.
Step 2. $k = k + 1$. Generate the matrix C^{k-1}.
Step 3. Solve equation (6.32).
Step 4. Return to step 2. Repeat until convergence criterion is satisfied.

The block-partioned matrix in (6.32) is not a K-matrix and also not an M-matrix, because of the zero block on the main diagonal. So we cannot obtain a regular splitting. In other words, it is not trivial to design a simple convergent iterative method.

The SIMPLE method

In 1972 an iterative method was proposed by Patankar and Spalding, which they called the SIMPLE method (SIMPLE stands for semi-implicit method for pressure-linked equations). With SIMPLE we do not solve equation (6.32) accurately, but perform only one iteration step. SIMPLE consists of the following steps:

Step 1. $k = 0$. Choose $u^0,\ p^0$.
Step 2. $k = k + 1$. Generate the matrix C^{k-1}.
Step 3. Update u by an approximate solve of the first row of (6.32):

$$C^{k-1}u^k = f - Gp^{k-1} \ ,$$

using a splitting $M_u - N_u = C^{k-1}$ of C^{k-1}, that is regular if C is a K-matrix, which is the case if the mesh Reynolds number is less than 2 or if we use the upwind scheme for the inertia terms; see Sect. 2.3 (equation (2.41)). Hence, if we would execute this step repeatedly it would converge. But we do only one iteration and then improve the pressure in step 4. We use the BIM in the form (6.11):

$$\frac{1}{\alpha_u} M_u \delta u = f - C^{k-1}u^{k-1} - Gp^{k-1}, \quad u^k = u^{k-1} + \delta u \ ,$$

where α_u is an underrelaxation factor that, unfortunately, must be determined by trial and error; $\alpha_u = 0.5$ seems to be a suitable value.

Step 4. Update p by an approximate solve of the following equation:

$$D\tilde{C}^{-1}G\delta p = Du^k \ , \tag{6.33}$$

where $\tilde{C} \equiv \text{diag}(C^{k-1})$. At first sight, the motivation for equation (6.33) seems unclear; it is inspired by the pressure-correction method, and is further justified below. It turns out that $D\tilde{C}^{-1}G$ is a K-matrix. So we use a splitting $M_p - N_p = D\tilde{C}^{-1}G$. By employing the underrelaxed BIM formulation (6.11) we obtain :

$$\frac{1}{\alpha_p} M_p \delta p = Du^k, \quad p^k = p^{k-1} + \delta p \ .$$

where the underrelaxation factor α_p must, unfortunately, be determined by trial and error; $\alpha_p = 0.8$ seems to be a suitable value.

Step 5. Second velocity update:

$$u^k := u^k - \tilde{C}^{-1}G\delta p \ .$$

Step 6. $k := k + 1$, and return to step 2. Repeat until convergence criterion is satisfied.

Distributive iteration

It is clear that upon convergence, when we have $u^k = u^{k-1}$ and $\delta p = 0$, we have indeed solved equation (6.28). But does SIMPLE converge? To see this it is convenient to take a modern point of view, and to regard SIMPLE as a member of the class of *distributive iteration methods*. This framework also makes equation (6.33) more understandable. The idea is as follows.
Suppose we have a linear system $Ay = b$. The system is *postconditioned*:

$$AB\hat{y} = b, \quad y = B\hat{y} \ . \tag{6.34}$$

The postconditioning matrix B is chosen such that (6.34) is easier to solve iteratively than $Ay = b$. For example, AB is an M-matrix, while A is not, as in our Navier-Stokes case above. For the iterative solution of (6.34) the splitting

$$AB = M - N \tag{6.35}$$

is introduced, to which corresponds the following splitting of the original matrix A:

$$A = MB^{-1} - NB^{-1} \ .$$

This leads to the following stationary iterative method for $Ay = b$:

$$MB^{-1}y^{k+1} = NB^{-1}y^k + b \ ,$$

or

$$y^{k+1} = y^k + BM^{-1}(b - Ay^k) \ . \tag{6.36}$$

This can be solved as follows:

Step 1. Solve $M\delta y = b - Ay^k$.
Step 2. $\delta y = B\delta y$.
Step 3. $y^{k+1} = y^k + \delta y$.

Note that if the iterative method defined by (6.36) converges ($y^{k+1} = y^k$), it solves $Ay = b$. This means that when designing a distributive iteration method, B and M in (6.36) need not be precisely the same as in (6.35). For example, one could use a complicated B to design a good M for (6.35), with N small, and then change M a little in (6.36) to make in step 1 $M\delta y = b - Ay^k$ more easily solvable, and also B may be simplified to make (6.36) easy to apply.

The method (6.36) is called *distributive iteration* for the following reason. Equation (6.36) and step 2 show that the correction $M^{-1}(b - Ay^k)$ corresponding to non-distributive ($B = I$) iteration is distributed, so to speak, by multiplication with B, over the elements of y^{k+1}, whence the name of the method.

Distributive iteration for Navier-Stokes

A number of well-known iteration methods for the incompressible Navier-Stokes equations can be interpreted as distributive methods. The distributive iteration framework permits us to give a unified description.

The system to be solved is (6.32). Define (writing C instead of C^{k-1} for brevity)

$$A = \begin{bmatrix} C & G \\ D & 0 \end{bmatrix} . \tag{6.37}$$

A possible choice for B is:

$$B = \begin{bmatrix} I & -\tilde{C}^{-1}G \\ 0 & I \end{bmatrix} , \tag{6.38}$$

resulting in

$$AB = \begin{bmatrix} C & (I - C\tilde{C}^{-1})G \\ D & -D\tilde{C}^{-1}G \end{bmatrix} .$$

Note that the zero block on the main diagonal has disappeared (this is the purpose of distributive iteration), so that basic iterative methods have a chance to converge. It remains to specify a suitable approximation M of AB. Thinking of $\tilde{C}$ as an approximation of C, we hope that $I - C\tilde{C}^{-1}$ will be small; this is the motivation behind equation (6.33). So we delete the block $(I - C\tilde{C}^{-1})G$. Furthermore, we replace C by M_u/α_u of step 3 of the SIMPLE method and $D\tilde{C}^{-1}G$ by M_p/α_p of step 4. This gives

$$M = \begin{bmatrix} \frac{1}{\alpha_u}M_u & 0 \\ D & -\frac{1}{\alpha_p}M_p \end{bmatrix} . \tag{6.39}$$

Substitution of (6.38) and (6.39) in (6.36) gives the following iterative method:

Step 1. Determine

$$\begin{bmatrix} r_u \\ r_p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} - \begin{bmatrix} C & G \\ D & 0 \end{bmatrix} \begin{bmatrix} u^k \\ p^k \end{bmatrix} . \tag{6.40}$$

Step 2. Solve

$$\begin{bmatrix} \frac{1}{\alpha_u}M_u & 0 \\ D & -\frac{1}{\alpha_p}M_p \end{bmatrix} \begin{bmatrix} \delta u \\ \delta p \end{bmatrix} = \begin{bmatrix} r_u \\ r_p \end{bmatrix} . \tag{6.41}$$

Step 3. Carry out the distribution step:

$$\begin{bmatrix} \delta u \\ \delta p \end{bmatrix} := B \begin{bmatrix} \delta u \\ \delta p \end{bmatrix} = \begin{bmatrix} \delta u - \tilde{C}^{-1}G\delta p \\ \delta p \end{bmatrix} . \tag{6.42}$$

This is completely equivalent to the SIMPLE method described earlier; the reader should verify this. By changing M and B different variants of the SIMPLE method that have appeared in the literature such as SIMPLER and SIMPLEC, and another distributive method called distributive Gauss-Seidel may be obtained. So distributive iteration is a useful unifying framework.

Roughly speaking, SIMPLE does for Navier-Stokes what Jacobi and Gauss-Seidel do for the Poisson equation studied earlier. So convergence will be slow and the work will still be $\mathcal{O}(N^2)$. But, like BIMs, SIMPLE lends itself well for acceleration by Krylov subspace methods and multigrid. We will not discuss this here. The interested reader is referred to Chapt. 7 of Wesseling (2001), and references given there.

When to stop SIMPLE iterations? The advice is to make the velocity field sufficiently divergence free, *i.e.* to make Du^k suficiently small, for instance

$$\parallel Du^k \parallel_\infty < \mathrm{tol}\, U/L, \quad \mathrm{tol} \ll 1 \ ,$$

where U and L are typical magnitudes for the velocity and the size of the domain. The reason is that if in incompressible flows $\operatorname{div} \mathbf{u}$ is not sufficiently small, we have numerical mass sources and sinks, which is usually very detrimental for physical realism.

Final remark

As we have seen, basic iterative methods usually are very slow to converge. As we remarked before, they are very useful if they are accelerated with Krylov subspace or multigrid methods. Unfortunately, in this course we have no time to discuss these methods.

Exercise 6.5.1. Verify that the distributive and the original formulation of the SIMPLE method are equivalent.

Some self-test questions

How many flops does it take to solve a system $Ay = b$ with A a symmetric $n \times n$ matrix with bandwidth $2m - 1$, $m \ll n$, with a direct method without reordering?

What is fill-in?

Describe the matrix structure of the linearized staggered scheme.

Describe the driven cavity benchmark problem.

What is a basic iterative method?

Give a good termination criterion for basic iterative methods.

What is a K-matrix? What is an M-matrix?

What is a regular splitting? What is the nice property of a regular splitting?

Describe the Jacobi, Gauss-Seidel, ILU and IC methods.

Let the iteration matrix of a BIM have spectral radius $\rho < 1$. How many iterations are equired to reduce the error by a factor r?

Describe distributive iteration. Why is it useful for Navier-Stokes?

References

Aris, R. (1962). *Vectors, Tensors and the Basic Equations of Fluid Mechanics.* Englewood Cliffs, N.J.: Prentice-Hall, Inc. Reprinted, Dover, New York, 1989.

Batchelor, G.K. (1967). *An Introduction to Fluid Dynamics.* Cambridge, UK: Cambridge University Press.

Burden, R.L. and J.D. Faires (2001). *Numerical analysis.* Pacific Grove: Brooks/Cole.

de Saint-Venant, B. (1843). Mémoire sur la dynamique des fluides. *C. R. Acad. Sci. Paris* **17**, 1240–1242.

Ghia, U., K.N. Ghia, and C.T. Shin (1982). High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *J. Comp. Phys.* **48**, 387–411.

Golub, G.H. and C.F. Van Loan (1989). *Matrix Computations (second edition).* Baltimore: The Johns Hopkins University Press.

Hackbusch, W. (1994). *Iterative Solution of Large Sparse Systems Equations.* New York: Springer.

Hinze, J.O. (1975). *Turbulence.* New York: McGraw-Hill.

Hirsch, C. (1988). *Numerical Computation of Internal and External Flows. Vol.1: Fundamentals of Numerical Discretization.* Chichester: Wiley.

Lighthill, J. (1986). The recently recognized failure of predictability in Newtonian dynamics. *Proc. R. Soc. London* **A407**, 35–50.

Meijerink, J.A. and H.A. van der Vorst (1977). An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.* **31**, 148–162.

Nakayama, Y. and W.A. Woods (Eds.) (1988). *Visualized Flow; Fluid Motion in Basic and Engineering Situations Revealed by Flow Visualization.* Oxford: Pergamon.

Navier, C.L.M.H. (1823). Mémoire sur les lois du mouvement des fluides. *Mém. Acad. R. Sci. Paris* **6**, 389–416.

Poisson, S.D. (1831). Mémoire sur les équations générales de l'équilibre et du mouvement des corps solides élastiques et des fluides. *Journal de l'Ecole Polytechnique de Paris* **13**, 139–166.

Stokes, G.G. (1845). On the theories of the internal friction of fluids in motion, and of the equilibrium and motion of elastic solids. *Trans. Camb. Phil. Soc.* **8**, 287–305.

Stokes, G.G. (1851). On the effect of the internal friction of fluids on the motion of pendulums. *Trans. Camb. Phil. Soc.* **9, Pt. II**, 8–106.

Trottenberg, U., C.W. Oosterlee, and A. Schüller (2001). *Multigrid*. London: Academic Press.

Van Dyke, M. (1982). *An Album of Fluid Motion*. Stanford: The Parabolic Press.

van Kan, J. and A. Segal (1993). *Numerieke methoden voor partiële differentiaalvergelijkingen*. Delft: Delftse Uitgevers Maatschappij.

Varga, R.S. (1962). *Matrix Iterative Analysis*. Englewood Cliffs, N.J.: Prentice-Hall.

Wesseling, P. (1992). *An Introduction to Multigrid Methods*. Chichester: Wiley. Available on Internet: www.mgnet.org/mgnet-books-wesseling.html.

Wesseling, P. (2001). *Principles of Computational Fluid Dynamics*. Heidelberg: Springer.

Young, D.M. (1971). *Iterative Solution of Large Linear Systems*. New York: Academic Press.

Index