9 APPLICATIONS OF MULTIGRID METHODS IN COMPUTATIONAL FLUID DYNAMICS

9.1. Introduction

The discipline to which multigrid has been applied most widely and shown its usefulness is computational fluid dynamics (CFD). We will, therefore, discuss some applications of multigrid in this field. It should, however, be emphasized again that multigrid methods are much more widely applicable, as discussed in Chapter 1. An early outline of applications to computational fluid dynamics is given in Brandt (1980); a recent survey is given by Wesseling (1990).

The principal aim of computational fluid dynamics is the computation of flows in complicated three-dimensional geometries, using accurate mathematical models. Thanks to advances in computer technology and numerical algorithms, this goal is now coming within reach. For example, in 1986 the Euler equations were solved numerically for the flow around a complete four-engined aircraft (Jameson and Baker 1986), probably for the first time. The main obstacles to be overcome are computing time requirements and the generation of computational grids in complex three-dimensional geometries. Multigrid can be a big help in overcoming these obstacles.

Grid generation

Grid generation can be assisted by multigrid by using overlays of locally refined grids in difficult subregions. By comparing solutions on overlapping grids of different mesh-size local errors can be assessed and local adaptive grid refinements can be implemented. Some publications in this area are:

**Computational complexity of computational fluid dynamics**

The two main dimensionless parameters governing the nature of fluid flows are the Mach number (ratio of flow velocity and sound speed (=300 ms$^{-1}$ in the atmosphere at sea level)) and the Reynolds number, defined as

$$Re = \frac{UL}{\nu}$$

where $U$ is a characteristic velocity, $L$ a characteristic length and $\nu$ the kinematic viscosity coefficient ($\nu = 0.15 \times 10^{-4}$ m$^2$ s$^{-1}$ for air at sea-level at 15°C, and $\nu = 0.11 \times 10^{-5}$ m$^2$ s$^{-1}$ for water at 15°C). The Reynolds number is a measure of the ratio of inertial and viscous forces in a flow. From the values of $\nu$ just quoted it follows that $Re \gg 1$ in most industrial flows. For example, $Re = 7 \times 10^4$ for flow of air at 1 m s$^{-1}$ past a flat plate 1 m long.

One of the most surprising and delightful features of fluid dynamics is the phenomenon that a rich variety of flows evolve as $Re \to \infty$. The intricate and intriguing flow patterns accurately rendered in masterful drawings by Leonardo da Vinci, or photographically recorded in Van Dyke (1982) are surprising, because the underlying physics (for small Mach numbers) is just a simple mass and momentum balance. A 'route to chaos', however, develops as $Re \to \infty$, resulting in turbulence.

Turbulence remains one of the great unsolved problems of physics, in the sense that accurate prediction of turbulent flows starting from first principles is out of the question, and other fundamentally sound prediction methods have not (yet) been found. The difficulty is that turbulence is both non-linear and stochastic. The strong dependence of flows on $Re$ complicates predictions based on scaled down experiments. At $Re = 10^7$ a flow may be significantly different from the flow at $Re = 10^5$, in the same geometry. A typical Reynolds number for a large aircraft is $Re = 10^7$ (based on wing chord). The impossibility of full-scale experiments means that computational fluid dynamics plays an important role in extrapolating to full scale. Ideally, one would like to simulate turbulent flows directly on the computer, solving the equations of motion that will be presented shortly. This involves solving the smallest scales of fluid motion that occur. The ratio of the length scales $\eta$ and $L$ of the smallest and largest turbulent eddies satisfies

$$\frac{\eta}{L} = O(Re^{-3/4})$$
Applications of multigrid methods in computational fluid dynamics

(Tennekes and Lumley 1972) with \( \text{Re} \) based on \( L \). The size of the flow domain will be bigger than \( L \), whereas the mesh size will need to be smaller than \( \eta \), so that the required number of cells in the grid will be at least

\[
\left( \frac{L}{\eta} \right)^3 = O(\text{Re}^{9/4}) \tag{9.1.3}
\]

Hence, direct simulation of turbulent flows is out of the question.

As far as accuracy is concerned, the next best thing is large eddy simulation. With this method large turbulent eddies are resolved, and small eddies are modelled heuristically. Their structure is to a large extent independent of the particular geometry at hand and largely universal. For large aircraft at \( \text{Re} = 10^7 \), Chapman (1979) has estimated a requirement of \( 8 \times 10^8 \) grid cells and \( 10^4 M \) words storage, assuming that large eddies are resolved only where they occur, namely in the thin boundary layer on the surface of aircraft, and in the wake. A crude estimate of the computational cost of a large eddy computation for a large aircraft may be obtained as follows. Taking as a rough guess for the cost per grid cell and per time step \( 10^3 \) flop (floating point operation), and assuming \( 10^2 \) time steps are required, we arrive at an estimate of \( 8 \times 10^4 G \) flop (1 \( G \) flop = \( 10^9 \) flop) for the computational cost. Such a computation is not feasible on present-day computers, but Teraflop (= \( 10^3 G \) flop) machines are expected to arrive during this decade, so that such computations will come within reach. Computations such as this would be of great technological value, and there are many other fluid mechanical disciplines where computations of similar scale would be very useful. As a consequence, the demands posed by CFD are a prime factor in stimulating the development of faster and larger computers, and more efficient algorithms.

In contemporary CFD technology simplified mathematical models are used to reduce storage and computing time requirements. In order of increasing complexity we have potential equations, Euler equations, Navier–Stokes equations (neglecting turbulence), Reynolds-averaged Navier–Stokes equations (crude turbulence modelling), large eddy simulation and direct simulation.

We will discuss the application of multigrid methods to the potential, Euler and Navier–Stokes equations. Table 9.1.1 (from Gentzsch et al. 1988) gives estimates of the required number of floating point operations for certain

<table>
<thead>
<tr>
<th>Model</th>
<th>Flop/cell/cycle</th>
<th>Number of cells</th>
<th>Number of cycles</th>
<th>Total Gflop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Potential, 3D</td>
<td>500</td>
<td>( 10^4 )</td>
<td>100–200</td>
<td>5–10</td>
</tr>
<tr>
<td>Euler, 2D</td>
<td>400</td>
<td>( 5 \times 10^3 )</td>
<td>500–1000</td>
<td>1–2</td>
</tr>
<tr>
<td>Euler, 3D</td>
<td>950</td>
<td>( 10^5 )</td>
<td>200–500</td>
<td>20–50</td>
</tr>
</tbody>
</table>

Table 9.1.1. Computing work for compressible inviscid flow computation
codes (by Jameson c.s.) to compute steady compressible inviscid flows, with the potential and Euler equations. Typical computations that one would like to carry out with the Euler or Navier–Stokes equations in three dimensions involve a computing task of the order of a Teraflop and a memory requirement of the order of a G word. Multigrid methods are a prime source of improvement in computing efficiency. We define a work unit (WU) as the number of operations involved in the definition of the discrete operator in one cell or grid point, times $N$: the total number of cells or grid points. A reasonable estimate of the minimum computing work required is thus a few WU. Multigrid methods make it possible to attain this lower bound, although this has not yet been completely achieved in many areas. Taking as a very rough guess $1 \text{ WU} = 500 \text{ N}$ for a typical fluid mechanics problem and assuming the work required to be 10 WU, we obtain the estimated lower bounds quoted in Table 9.1.2. Comparison of Tables 9.1.1 and 9.1.2 indicates that much is still to be gained from algorithmic improvements.

### 9.2. The governing equations

**Navier–Stokes equations**

Fluid dynamics is a classical discipline. The physical principles underlying the flow of simple fluids such as water and air have been understood since the time of Newton, and the mathematical formulation has been complete for a century and a half. The equations describing the flow of fluids are the Navier–Stokes equations. These give the laws of conservation of mass, momentum and energy. Let $p$, $\rho$, $T$, $e$ and $u_\alpha$ be the pressure, density, temperature, total energy and velocity components in a Cartesian reference frame with coordinates $x_\alpha$. The conservation laws have the form

$$\frac{\partial q}{\partial t} + F_{\beta,\beta} = 0$$

(9.2.1)

using Cartesian tensor notation and the summation convention: summation takes place over repeated Greek indices ($F_{\beta,\beta} = \Sigma_\delta \partial F_{\delta}/\partial x^\delta$). For the mass
conservation equation we have

\[ q = \rho, \quad F_\beta = \rho u_\beta \quad (9.2.2) \]

For the \( x^\alpha \)-momentum conservation equation we have

\[ q = q_\alpha = \rho u_\alpha, \quad F_\beta = F_{\alpha\beta} = \rho u_\alpha u_\beta + \rho \delta_{\alpha\beta} - \sigma_{\alpha\beta} \quad (9.2.3) \]

with \( \sigma_{\alpha\beta} \) the viscous stress tensor, given by

\[ \sigma_{\alpha\beta} = \mu (u_{\alpha,\beta} + u_{\beta,\alpha}) - \frac{2}{3} \mu \delta_{\alpha\beta} u_{\gamma,\gamma} \quad (9.2.4) \]

with \( \mu = \rho \nu \) the dynamic viscosity coefficient. For the energy conservation equation we have

\[ q = \rho e, \quad F_\beta = (\rho e + p) u_\beta - \sigma_{\beta\gamma} u_\gamma - \eta T,_{\beta} \quad (9.2.5) \]

with \( \eta \) the heat conduction coefficient. The system of equations is completed by the equation of state for a perfect gas: \( p = \rho RT \), with \( R \) a constant. The temperature \( T \) is related to \( e \) by \( c_v T = e - \frac{1}{2} u_\alpha u_\alpha \), with the coefficient \( c_v \) the specific heat at constant volume. Noting that \( R = c_p - c_v \) (\( c_p \) is the specific heat at constant pressure), elimination of \( T \) gives

\[ p = (\gamma - 1) \rho (e - \frac{1}{2} u_\alpha u_\alpha) \quad (9.2.6) \]

with \( \gamma \) the ratio of specific heats; \( \gamma = 7/5 \) for air.

The Navier–Stokes equations are of parabolic type. In the time-independent case they are elliptic. For the computation of time-dependent flows the time step should be small with respect to the timescale of the physical phenomena to be modelled. As a consequence, the result of the previous time step is usually a good approximation of the solution at the new time level, so that often relatively simple iteration methods suffice, and multigrid does not lead to such drastic efficiency improvements as in the time-independent case. Henceforth we shall consider only the latter case.

**Euler and potential equations**

Neglecting viscosity and heat conduction (\( \mu = \eta = 0 \)), equations (9.2.1) reduce to the *Euler equations*. These form a system that is hyperbolic in time.

From the Euler equations, the potential flow model is obtained by postulating

\[ u_\alpha = \phi,_{\alpha} \quad (9.2.7) \]

with \( \phi \) the velocity potential. Substitution of (9.2.7) in the mass conservation
equation gives, neglecting time dependence,

\[(\rho \Phi, a)_{,a} = 0\]  \hspace{1cm} (9.2.8)

which is the potential equation. It can be shown that (cf. Fletcher 1988, Section 14.3.1) in potential flow the density is related to the magnitude of the velocity by

\[\rho = \rho_{\infty} \left(1 + \frac{\gamma - 1}{2} M_0^2 (1 - q_z^2 / q^2)\right)^{\gamma / (\gamma - 1)}\]  \hspace{1cm} (9.2.9)

Here the subscript \(\infty\) denotes some reference state, for example upstream infinity, \(q_z^2 = u_z u_{\alpha}; M = q / c\), with \(c\) the speed of sound, is the Mach number. The potential equation is elliptic where the local velocity is subsonic, and hyperbolic where it is supersonic. Hence, in transonic flow it is of mixed type.

In order to distinguish (9.2.8) and (9.2.9) from more simplified models (used in classical aerodynamics) involving various approximations in (9.2.9), Equation (9.2.8) with \(\rho\) given by (9.2.9) is often called the full potential equation.

For more information on the basic equations and on the boundary conditions, see texts on fluid dynamics, such as Landau and Lifshitz (1959), or texts on computational fluid dynamics, such as Richtmyer and Morton (1967), Peyret and Taylor (1983), Fletcher (1988) or Hirsch (1988, 1990).

9.3. Grid generation

For the discretization of the governing equations a computational grid has to be chosen. One of the distinguishing features of present-day computational fluid dynamics is the geometric complexity of the domains in which flows of industrial interest take place. The generation of grids in complicated three-dimensional domains is a far from trivial affair, and is one of the major problem areas in computational fluid dynamics at present. Much research is going on. For a survey of the state-of-the-art in grid generation in computational fluid dynamics and introduction to the literature, see Sengupta et al. (1988), Thompson and Steger (1988), Thompson et al. (1985) and Thompson (1987).

Boundary conforming grids

There are various types of grids. This is not the place to discuss their relative merits; see Wesseling (1991). The present trend in computational fluid dynamics seems to favour structured boundary conforming grids. A mapping

\[x = x(\xi), \quad x \in \Omega, \quad \xi \in G\]  \hspace{1cm} (9.3.1)
is constructed, with $\Omega$ the physical domain and $G$ a cube. The boundary $\partial \Omega$ consists of segments on each of which we have $\xi^\alpha = 0$ for some $\alpha$, which is why the grid is called boundary conforming or boundary fitted. This feature facilitates the accurate implementation of boundary conditions. A uniform grid is chosen in $G$; its image is the computational grid in physical space, cf. Figure 9.3.1. The local topological structure (number of neighbouring cells, etc.) is uniform, this type of grid is called \textit{structured}. This feature simplifies the data structures required, and facilitates efficient vector and parallel computing. The coarse grids required for multigrid are constructed in the standard way by doubling the mesh size in $G$. Henceforth it is assumed that a structured boundary conforming grid is used.

\textbf{Some tensor analysis}

Since the $\xi^\alpha$ coordinates are arbitrary, it is convenient to express the equation in an \textit{invariant} (i.e. coordinate independent) form. The tool for this is \textit{tensor analysis}. The fundamentals of tensor analysis, especially in relation to continuum mechanics, may be found in Aris (1962), Sedov (1977) or Sokolnikoff (1964). We present some elementary facts. The \textit{covariant base vectors} $a_{(\alpha)}$ and \textit{metric tensor} $g_{\alpha\beta}$ are defined by

$$ a_{(\alpha)} = \frac{\partial x}{\partial \xi^\alpha}, \quad g_{\alpha\beta} = a_\alpha \cdot a_{(\beta)} \quad (9.3.2) $$

The determinant of $g_{\alpha\beta}$ is called $g$ and follows from

$$ g^{1/2} = a_{(1)} \cdot (a_{(2)} \wedge a_{(3)}) \quad (9.3.3) $$

In two dimensions this becomes

$$ g^{1/2} = a_{(1)}^1 a_{(2)}^2 - a_{(1)}^2 a_{(2)}^1 \quad (9.3.4) $$
where $a(\beta)$ are the Cartesian components of $a(\beta)$; here and in the following lower case letters indicate Cartesian components, whereas capitals indicate components in a general reference frame. The quantity $g^{1/2}$ equals the Jacobian of the mapping $x = x(\xi)$. The contravariant base vectors $a^{(\alpha)}$ and metric tensor $g^{\alpha\beta}$ are defined by

$$a^{(\alpha)} = \frac{\partial \xi^\alpha}{\partial x^\beta}, \quad g^{\alpha\beta} = a^{(\alpha)} \cdot a^{(\beta)} \tag{9.3.5}$$

The independent variables on the computational grid are $\xi^\alpha$, thus $a^{(\alpha)}$ is easily obtained by finite difference approximation, but $a^{(\alpha)}$ is not. $a^{(\alpha)}$ can, however, be obtained from $a(\alpha)$ by using

$$a^{(\alpha)} \cdot a(\beta) = \delta^\beta_\alpha \tag{9.3.6}$$

with $\delta$ the Kronecker delta. In two dimensions this gives

$$a^{(1)} = \frac{1}{g^{1/2}} (a_2^2, -a_2^1), \quad a^{(2)} = \frac{1}{g^{1/2}} (-a_1^2, a_1^1) \tag{9.3.7}$$

The covariant and contravariant components of a vector field $u$ are given by, respectively,

$$U^\alpha = u \cdot a^{(\alpha)}, \quad U_\alpha = u \cdot a(\alpha) \tag{9.3.8}$$

Equation (9.3.8) shows how the components of a vector field change under coordinate transformation. Superscripts may be raised or lowered by contraction (i.e. multiplication and summation) with $g^{\alpha\beta}$ or $g_{\alpha\beta}$, for example

$$U^\alpha = g^{\alpha\beta} U_\beta, \quad U_\alpha = g_{\alpha\beta} U^\beta \tag{9.3.9}$$

The divergence of a vector field $u$ is given by

$$\text{div } u = U^\alpha_{,\alpha} = \frac{1}{g^{1/2}} \frac{\partial}{\partial \xi^\alpha} (g^{1/2} U^\alpha) \tag{9.3.10}$$

For the definition of the covariant derivative $U^\alpha_{\beta}$, not needed here, the reader is referred to the literature. The covariant derivative of a scalar $\varphi$ is defined by

$$\varphi_{,\alpha} = \frac{\partial \varphi}{\partial \xi^\alpha} \tag{9.3.11}$$

**Generation of structured boundary conforming grids**

A widely used method to construct structured boundary conforming grids is *elliptic grid generation*. An introduction to this method is given by Thompson
et al. (1985). The mapping $\xi = \xi(x)$ is defined as the solution of a Poisson equation:

$$\frac{\partial^2 \xi^\alpha}{\partial x^\beta \partial x^\delta} = P^\alpha(\xi), \quad x \in \Omega$$

(9.3.12)

The functions $P^\alpha(\xi)$ and the boundary conditions are used to influence the position and the orientation of the grid lines. The boundary $\partial \Omega$ is divided in segments in a suitable way. On each of these segments a constant value is assigned to $\xi^\alpha$ for some $\alpha$; this makes the grid boundary conforming. A relation between $x$ and the remaining components of $\xi$ is chosen, which determines the position of the grid lines at $\partial \Omega$.

Since the grid is generated by specifying grid lines in the $\xi$-plane, the mapping $x = x(\xi)$ is required instead of $\xi = \xi(x)$. Therefore the dependent and the independent variables in (9.3.12) have to be reversed. This can be done as follows. Suppose we have a quantity $\varphi$ satisfying

$$\frac{\partial^2 \varphi}{\partial x^\alpha \partial x^\beta} = 0, \quad x \in \Omega$$

(9.3.13)

Changing to $\xi$-coordinates satisfying (9.3.12) one obtains

$$a_{\alpha}^{(\gamma)} \frac{\sigma}{\partial \xi^\gamma} \left( a_{\alpha}^{(\beta)} \frac{\partial \varphi}{\partial \xi^\beta} \right)$$

$$= a_{\alpha}^{(\beta)} a_{\gamma}^{(\alpha)} \frac{\partial^2 \varphi}{\partial \xi^\beta \partial \xi^\gamma} + a_{\gamma}^{(\beta)} \frac{\partial}{\partial \xi^\gamma} (a_{\alpha}^{(\beta)} \frac{\partial \varphi}{\partial \xi^\beta})$$

$$= g_{\beta \gamma} \frac{\partial^2 \varphi}{\partial \xi^\beta \partial \xi^\gamma} + \frac{\partial^2 \xi^\beta}{\partial x^\alpha \partial x^\alpha} \frac{\partial \varphi}{\partial \xi^\beta}$$

$$= g_{\beta \gamma} \frac{\partial^2 \varphi}{\partial \xi^\beta \partial \xi^\gamma} + P^\beta(\xi) \frac{\partial \varphi}{\partial \xi^\beta} = 0$$

(9.3.14)

Choosing $\varphi = x^\delta$ Equation (9.3.13) holds, and (9.3.14) gives (renaming $\delta$ by $\alpha$):

$$g_{\beta \gamma} \frac{\partial^2 x^\alpha}{\partial \xi^\beta \partial \xi^\gamma} + P^\beta(\xi) \frac{\partial x^\alpha}{\partial \xi^\beta} = 0, \quad \xi \in G.$$  

(9.3.15)

This, together with appropriate boundary conditions, defines the mapping $x = x(\xi)$. Choosing the boundary conditions and the control functions $P^\beta(\xi)$ such as to obtain a grid with the desired properties is quite an art. For further information, see the literature.

Equation (9.3.15) may be solved numerically as follows. A uniform grid is chosen in $G$, and (9.3.15) is discretized by standard central finite differences. The resulting non-linear algebraic system does not need to be solved accurately, since the sole aim is to obtain a reasonable distribution of grid points.
in $\Omega$. Multigrid methods are easily applied, and efficient. One possibility is to let $g^{B^2}$ lag behind in an iterative procedure, and to solve the resulting linear system approximately with a standard linear multigrid code. Another possibility is to apply a few non-linear multigrid iterations. A non-linear smoother is easily obtained by letting $g^{B^2}$ lag behind. In both cases, a start with nested iteration is to be recommended.

An example: generation of a grid around an airfoil

The geometrical situation is sketched in Figure 9.3.2. The domain is two-dimensional, and consists of the region exterior to an airfoil. The domain is

![Physical plane](https://via.placeholder.com/150)

![Computational plane](https://via.placeholder.com/150)

**Figure 9.3.2** Mapping from computational plane to physical plane.

![Part of the grid around an airfoil](https://via.placeholder.com/150)

**Figure 9.3.3** Part of the grid around an airfoil.
Applications of multigrid methods in computational fluid dynamics

made finite for numerical reasons by truncation at a large distance from the airfoil by some curve, for which we take a circle. The mapping \( x = x(\xi) \) maps a computational rectangle onto the physical domain, according to Figure 9.3.2.

The physical domain is doubly connected. It is made simply connected by a cut emanating from the trailing edge. A uniform grid is chosen in the computational rectangle. Figure 9.3.3 shows part of the grid (the image of the computational grid) in the physical plane. The outer boundary and the airfoil consist of curves \( \xi^2 = \text{constant} \); on the cut we have \( \xi^1 = \text{constant} \) with different constants on both sides of the cut.

9.4. The full potential equation

It is assumed that the flow is transonic. The first numerical method for the resulting nonlinear elliptic–hyperbolic problem appeared in 1971 (Murman and Cole 1971). It has been possible to reduce the required computing time drastically by means of multigrid. Many publications have appeared in this field; see the multigrid bibliography in McCormick (1987), and the papers by Becker (1988), Liu Chaoqun and McCormick (1988), Van der Wees et al. (1983) and Van der Wees (1984, 1985, 1986, 1989).

We will see that the treatment of the full potential equation involves in addition to standard techniques in the numerical approximation of partial differential equations some special considerations, which are typical for computational fluid dynamics.

Invariant formulation of the full potential equation

It is assumed that the flow is time independent. The invariant (i.e. coordinate independent) form of the continuity equation (9.2.2) is

\[
\text{div } \rho u = 0
\] (9.4.1)

using (9.3.10) this becomes

\[
\frac{1}{g^{1/2}} \frac{\partial}{\partial \xi^\alpha} (g^{1/2} U^\alpha) = 0
\] (9.4.2)

Equation (9.2.7) gives, using (9.3.9)

\[
U^\alpha = g^{\alpha\beta} \varphi, \beta
\] (9.4.3)

The density \( \rho \) is given by (9.2.9), with

\[
q^2 = U^\alpha U_\alpha = g^{\alpha\beta} \varphi, \alpha \varphi, \beta
\] (9.4.4)
We restrict ourselves to the two-dimensional case. The coordinate mapping and the grid are presented in Figures 9.3.2 and 9.3.3.

The boundary conditions

The flow must be tangential to the airfoil surface. On the airfoil we have \( \xi^2 = 0 \), hence

\[
\mathbf{u} \cdot \mathbf{n} \big|_{\xi^2=0} = 0
\]  

(9.4.5)

with the normal at the airfoil. Since \( \mathbf{n} \parallel \mathbf{a}^{(2)} \), equation (9.4.5) is equivalent to, using (9.3.8), \( U^2 \big|_{\xi^2=0} = 0 \), or

\[
\mathbf{x}^2 \mathbf{\varphi}_n \big|_{\xi^2=0} = 0
\]  

(9.4.6)

Assuming that at infinity the magnitude of the velocity is \( q_{\infty} \) and that the flow is parallel to the \( \mathbf{x}^1 \) axis in a suitably rotated Cartesian frame \( (\mathbf{x}^1, \mathbf{x}^2) \), the potential at the outer circle is prescribed as

\[
\varphi \big|_{\xi^2=1} = q_{\infty} \mathbf{x}^2
\]  

(9.4.7)

The fact that (9.4.7) is prescribed at a finite distance from the airfoil instead of at infinity (in which case one would work with \( \varphi^1 = \varphi - q_{\infty} \mathbf{x}^2 \) instead of with \( \varphi \), which becomes infinite of course) causes an inaccuracy, which may be diminished by employing an asymptotic expansion for the far field of potential flow. Assuming at infinity the flow is subsonic, a more accurate condition than (9.4.7) is (Ludford 1951)

\[
\varphi \big|_{\xi^2=1} = q_{\infty} \mathbf{x}^2 + \frac{\Gamma}{2\pi} \tan^{-1}((1 - M_{\infty}^2)^{1/2} \mathbf{x}^1/\mathbf{y})
\]  

(9.4.8)

Here \( \Gamma \) is the circulation around the airfoil, which has to be determined as part of the solution.

Determination of the circulation

A condition along the cut \( (\xi^1 = 0, 1, \text{ cf. Figure 9.3.2}) \) is obtained as follows. The pressure is continuous. In potential flow the magnitude of the velocity is a continuous function of the pressure. Assuming the velocity field to be non-singular this implies that the tangential velocity component at the cut is continuous, hence \( \varphi(0, \xi^2) - \varphi(1, \xi^2) = \text{constant} \). As suggested by (9.4.8), this constant equals \( \Gamma \):

\[
\varphi(0, \xi^2) - \varphi(1, \xi^2) = \Gamma
\]  

(9.4.9)
Of course, the mass conservation equation (9.4.2) must also be applied across the cut, taking (9.4.9) into account. This is done as follows. Assume point \( Z \) lies on the cut. Corresponding to \( Z \in \Omega \) there are two point \( Z', Z'' \in G \) with coordinates \((0, \xi Z')\) and \((1, \xi Z'')\). When differences of \( \phi \) are formed approximating (9.4.3), \( \phi Z' \) or \( \phi Z'' \) is used, such that differences across the cut are avoided. Next, \( \phi Z' \) is eliminated using (9.4.9).

The circulation \( \Gamma \) follows from the Kutta condition, which requires that the velocity field is smooth at a sharp trailing edge, i.e.

\[
\lim_{\xi \to 0} q(\xi, 0) = \lim_{\xi \to 1} q(\xi, 0) \quad (9.4.10)
\]

Finite volume discretization

Figure 9.4.1 shows part of the computational grid, with an \textit{ad hoc} numbering of the grid points. The potential \( \phi \) is approximated in the vertices of the grid (vertex-centred discretization). Equation (9.4.2) is integrated over a finite volume \( \Omega \) surrounding point 5, indicated by broken lines in Figure 9.4.1. This gives

\[
\int_{\Omega} \frac{\partial}{\partial \xi} \left( g^{1/2} \rho U x \right) d\xi d\xi' = \left( g^{1/2} \rho U^1 \right) \delta \xi + \left( g^{1/2} \rho U^2 \right) \delta \xi' \quad (9.4.11)
\]

When point 5 lies on the airfoil surface we apply boundary condition (9.4.6) by substituting \( (g^{1/2} \rho U^2)_{D} = -(g^{1/2} \rho U^2)_{B} \). The Kutta condition (9.4.10) is handled as follows. Let point 5 lie at the trailing edge. The corresponding control volume, consisting of two parts, is depicted in Figure 9.4.2. The Kutta condition is implemented as \( q_A = q_C \). We have

\[
q_A = \frac{u \cdot a_{(1)} | a_{(1)} |}{A} = \left\{ \varphi_1 \left[ \left( \frac{\partial X^1}{\partial \xi} \right)^2 + \left( \frac{\partial X^2}{\partial \xi} \right)^2 \right]^{-1/2} \right\}_A
\]

![Figure 9.4.1 Part of computational grid in \( \xi \) plane.](image-url)
hence, the Kutta condition gives

\[(\varphi_6 - \varphi_5) \left( \frac{\partial x^1}{\partial \xi^1} \right)^2 + \left( \frac{\partial x^2}{\partial \xi^1} \right)^2 \right]^{-1/2}_A = (\varphi_5 - \varphi_4) \left( \frac{\partial x^1}{\partial \xi^1} \right)^2 + \left( \frac{\partial x^2}{\partial \xi^1} \right)^2 \right]^{-1/2}_C \quad (9.4.12)\]

In addition to (9.4.12) we have the discretization over the finite volume of Figure 9.4.2.

A discrete system is obtained by substitution of (9.4.3) in (9.4.11), discretizing \( \varphi, \beta \) with central differences. In the interior, the nine-point stencil consisting of the points 1 to 9 in Figure 9.4.1 results. The circulation \( \Gamma \) may be determined as follows. Two values for the circulation \( \Gamma^* \) and \( \Gamma^{**} \) are chosen, and the corresponding solutions \( \varphi^* \) and \( \varphi^{**} \) are determined, neglecting (9.4.12). Then \( \omega \) is determined such that \( \varphi = \omega \varphi^* + (1 - \omega) \varphi^{**} \) satisfies (9.4.12). The new estimate for the circulation becomes \( \Gamma^* := \omega \Gamma^* + (1 - \omega) \Gamma^{**} \); a new \( \Gamma^{**} \) that does not differ much from \( \Gamma^* \) is chosen, and the process is repeated.

**Retarded density**

Before the discretization can be considered complete a final complication needs to be discussed. When \( M_a \) is sufficiently close to 1 a local supersonic zone appears adjacent to the airfoil, usually terminated at the downstream side by a shock. In the shock dissipation takes place, which is an irreversible thermodynamic process, resulting in an increase in entropy. The potential flow model is completely reversible (free from dissipation). As a consequence it allows not only (isentropic approximations of) compression shocks, but also expansion shocks, which are unphysical. To avoid these some irreversibility must be built in. One way to do this is to use the *retarded density* concept (Holst 1978, Hafez et al. 1979). In regions where the flow is locally supersonic the density \( \rho \) is not evaluated in the point where it should be according to
(9.4.11), but in the neighbouring grid point in the upstream direction. Our grid is shaped such that near the airfoil the flow is roughly aligned with the $\xi^1$ coordinate lines, so that it will suffice to displace the density in the $\xi^1$ direction; $\rho$ is replaced by $\tilde{\rho}$ defined by

$$\tilde{\rho} = \rho - \nu(M)D_1\rho$$  \hspace{1cm} (9.4.13)

with $D_1\rho$ the upwind undivided difference on the grid, and $\nu(M)$ the following smooth switching function

$$\nu = 0, \quad M/M_{\infty} < (1 + \epsilon)^{-1/2}$$

$$\nu = (M^2/M^2 - 1 - \epsilon)^2/4\epsilon, \quad (1 + \epsilon)^{-1/2} \leq M/M_{\infty} < (1 - \epsilon)^{-1/2}$$

$$\nu = 1, \quad M/M_{\infty} \geq (1 - \epsilon)^{-1/2}$$  \hspace{1cm} (9.4.14)

where $M_{\infty}$ and $\epsilon$ are parameters to be chosen: $M_{\infty}$ slightly less than 1. As a consequence of retarding the density, the accuracy of the discretization is only first order in supersonic zones.

**Multigrid method**

One way to solve the non-linear system of equations just described is to use Newton iteration on the global system, and to solve the resulting linear systems by a standard linear multigrid method, for example one of the codes discussed in Section 8.8. This approach has been followed by Nowak and Wesseling (1984), where it is found that multigrid solves the linear problems efficiently. The Newton process also converges rapidly for subsonic flow, but for transonic flow the convergence of the Newton process is erratic and requires many iterations, because the Fréchet derivative of the system is ill conditioned. This approach is not, therefore, to be recommended.

As has already been mentioned in Section 8.8, a very nice property of the non-linear multigrid algorithm is that global linearization is not required. Only in the smoother a local linearization is applied. This has been done by Nowak (1985). As a result non-linear multigrid converges fast, even though the global Fréchet derivative is ill conditioned. All that has to be done is to choose the coarse grids, the transfer operators $P^k$ and $R^k$, and the smoothing method. The coarse grids are constructed by successive doubling of the mesh size. $P^k$ and $R^k$ can be chosen using linear or bilinear interpolation according to Section 5.3 with $R^k = s(P^k)^*$, choosing $s$ such that the sum of the elements of $[R^k]$ equals 1. This gives us $m_P + m_R = 4 > 2m = 2$, satisfying rule (5.3.18). The choice of the smoothing method is less straightforward.

**Smoothing method**

In order to find out how the smoothing method should be chosen we study the *small disturbance limit* of (9.4.2) in Cartesian coordinates, that is, the
The full potential equation

Airfoil is assumed to be very thin and the flow is assumed to deviate little from the uniform flow field \( \mathbf{u}_\infty = (1, 0) \). Denoting the Cartesian components of the velocity disturbance by \( u^\alpha \) we have

\[
\mathbf{u} = (1 + u^1, u^2)
\]  
(9.4.15)

with \( |u^\alpha| \ll 1 \). From (9.2.9) it follows that

\[
\rho = \rho_\infty (1 - M^2_\infty u^1)
\]  
(9.4.16)

Taking \( \xi = x \), substituting (9.4.15) in (9.4.11) and writing \( U^\alpha = u^\alpha = \varphi, \alpha \) results in the following discretization, with \( \delta \xi^1 = \delta \xi^2 \)

\[
\{\rho (1 + \varphi, 1)\} \nabla + \{\rho \varphi, 2\} \frac{\mathbf{B}}{||B||} = 0
\]  
(9.4.17)

If \( M^2_\infty < 1 \) then \( \rho \) is not replaced by \( \bar{\rho} \) (cf. (9.4.13)). Equation (9.4.17) is approximated further by, using (9.4.16),

\[
(1 - M^2_\infty) \varphi, 1 \nabla + \varphi, 2 \frac{\mathbf{B}}{||B||} = 0
\]  
(9.4.18)

If \( M^2_\infty > 1 \) then \( \rho \) is replaced by \( \bar{\rho} \) according to (9.4.13) and we obtain

\[
(1 - M^2_\infty (D_1 - 1)) \varphi, 1 \nabla + \varphi, 2 \frac{\mathbf{B}}{||B||} = 0
\]  
(9.4.19)

Note that (9.4.18) and (9.4.19) correspond to an elliptic partial differential equation if \( M^2_\infty < 1 \), and to a hyperbolic equation if \( M^2_\infty > 1 \).

Discretizing the derivatives equation (9.4.19) becomes

\[
\nabla^2 \varphi |5 - M^2_\infty (\varphi_5 - 2\varphi_4 + \varphi_{11}) = 0
\]  
(9.4.20)

Discretization (9.4.20) is stable for \( M^2_\infty > 1 \), but (9.4.18) is not; this is another justification of the retarded density formula (9.4.13). For further discussion, see Hirsch (1990), Vol. 2 Section 5.1.

The smoother has to be chosen such that it works for (9.4.18) with \( M^2_\infty < 1 \) and for (9.4.20) with \( M^2_\infty > 1 \). Furthermore, in transonic flows \( M^2_\infty - 1 \ll 1 \). Equation (9.4.18) is equivalent to test problem (7.5.6) with \( \beta = 0 \) and \( \varepsilon \ll 1 \), so possible candidates are the smoothers discussed in Chapter 7 that work for this test problem. Smoothing analysis for (9.4.20) is carried out in Example 9.4.1, according to the principles set out in Chapter 7.

Example 9.4.1. Smoothing analysis for equation (9.4.20). A method that works for (9.4.18) is backward vertical line Gauss-Seidel. The amplification factor of this smoother applied to (9.4.20) is easily found to be

\[
\lambda(\theta) = \frac{(1 + 2M^2_\infty) e^{-i\theta_1} - M^2_\infty e^{-2i\theta_1}}{4 - 2 \cos \theta_1 - e^{-i\theta_2}}
\]  
(9.4.21)
Applications of multigrid methods in computational fluid dynamics

Table 9.4.1. Fourier smoothing factor \( p \) for equation (9.4.20). Forward vertical line Gauss–Seidel smoothing; \( n = 64 \)

<table>
<thead>
<tr>
<th>( M_\infty )</th>
<th>1.0</th>
<th>1.1</th>
<th>1.3</th>
<th>1.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p )</td>
<td>0.34</td>
<td>0.34</td>
<td>0.35</td>
<td>0.38</td>
</tr>
</tbody>
</table>

Hence \( | \lambda(\pi, 0) | = (3M_\infty^2 + 1)/5 \), so that \( | \lambda(\pi, 0) | = 0.8 \) for \( M_\infty = 1 \) and \( | \lambda(\pi, 0) | \geq 1 \) for \( M_\infty \geq (4/3)^{1/2} = 1.15 \), so that this is not a good smoother if \( M_\infty \geq 1 \). This is not surprising, since for \( M_\infty > 1 \) the underlying problem is hyperbolic, and information flows from left to right, so that we are sweeping in the wrong direction. Forward vertical line Gauss–Seidel (also a good smoother for (9.4.18)) sweeps in the right direction, and is found to be a good smoother. The derivation of the amplification factor is left to the reader. Table 9.4.1 presents some values of the smoothing factor. Clearly, this is a satisfactory smoother.

Essential ingredients for the numerical solution of the transonic potential equation are the use of different discretizations in the subsonic and supersonic parts of the flow (cf (9.4.13)), and the use of forward vertical line Gauss–Seidel iteration; this is the approach that led to the first successful numerical method for this problem (Murman and Cole 1971). Most multigrid methods applied to this problem, starting with South and Brandt (1976) include, therefore, some form of forward vertical line Gauss–Seidel smoothing. If in parts of the physical space the mesh is strongly stretched in the \( \xi^2 \)-direction (corresponding to \( \beta = \pi/2 \) and \( \epsilon \ll 1 \) in test problem (7.4.7)) then horizontal line Gauss–Seidel smoothing must also be incorporated. ILU smoothing can also be used (Nowak and Wesseling 1984, Van der Wees et al. 1983, Van der Wees 1984, 1985, 1986, 1989). Zebra smoothing has not been investigated, but is expected to work, provided damping is used, because of the hyperbolic nature in the supersonic zone; cf. the results of Fourier smoothing analysis of alternating zebra for the convection-diffusion equation in Section 7.10.

9.5. The Euler equations of gas dynamics

We consider the two-dimensional case only. Although the grid is curvilinear, it is not necessary to use general tensor notation. We will use Cartesian tensor notation. Putting \( \mu = \eta = 0 \), equations (9.2.1) reduce to the Euler equations. These can be written as

\[
\frac{\partial q}{\partial t} + g_{\beta, \beta} = s
\]  
(9.5.1)
The Euler equations of gas dynamics

with \( q = (\rho, \rho u_1, \rho u_2, \rho e)^T \), \( g_1 = (\rho u_1, \rho u_1 u_1 + p, \rho u_1 u_2, (e + p)u_1)^T \), \( g_2 = (\rho u_2, \rho u_1 u_2, \rho u_2 u_2 + p, (e + p)u_2)^T \). The system of equations is completed by (9.2.6). A known source term \( s \) has been added for generality. Even if \( s = 0 \), there will be a non-zero right-hand side on the coarse grids when a multigrid method is used.

For a discussion of the boundary conditions that should accompany the hyperbolic system (9.5.1) the reader is referred to Hirsch (1990, Chapter 19).

Finite volume discretization

Discretization of (9.5.1) may take place by means of the finite element method, or the finite volume method, or the finite difference method. There is not much difference between the last two methods. Finite difference methods of Lax–Wendroff type, especially the MacCormack variant (see Hirsch 1990) have long been popular and are still widely used, but are being superseded by finite volume methods. For brevity, we restrict ourselves to the finite volume method.

Equations (9.5.1) constitute a hyperbolic system. Solutions often exhibit discontinuities (shock waves, contact discontinuities). These discontinuities should be accurately represented in numerical approximations. It is desirable to have: (i) second-order accuracy; (ii) monotonicity; (iii) fulfillment of the entropy condition; (iv) crisp resolution of discontinuities. By monotonicity we mean that the numerical scheme produces no artificial extrema as time progresses, so that there are no numerical 'wiggles' near discontinuities. The entropy condition refers to a thermodynamic property of the dissipation process that occurs in shocks, and which is not modelled by the Euler equations, because all dissipation is neglected, since \( \mu = \eta = 0 \). The entropy condition states that the entropy should be non-decreasing, so that non-physical expansion shocks are ruled out. The entropy condition can be fulfilled by building in some form of irreversibility in the numerical scheme, as was done in the preceding section by retarding the density \( \rho \). For a fuller discussion of the entropy condition see Hirsch (1990) Chapter 21. Requirements (i) and (ii) can only be satisfied by non-linear numerical schemes, i.e. schemes that are non-linear, even if (1) is linear. This is because of the fact that linear monotone schemes are necessarily of at most first order accuracy (Harten et al. (1976)).

Figure 9.5.1 presents part of a computational grid. The unknowns \( q \) may be assigned to the vertices of the cells or finite volumes (such as A, B, C, D) or to the centres. For the former approach, see Hall (1986), Jameson (1988), Mavripilis (1988) and Morton and Paisley (1989). We proceed with the cell-centred approach, the fundamentals of which have been presented in Section 3.7.

Equation (9.5.1) is integrated over each of the cells separately. Integration
over the finite volume $\Omega_{ij} = ABCD$ gives

$$a_{ij} \frac{d}{dt} q_{ij} + \int_{S_{ij}} g_\beta \, dS_\beta = a_{ij} s_{ij} \quad (9.5.2)$$

with $a_{ij}$ the area of $\Omega_{ij}$, $q_{ij}$ the value of $q$ at the centre of $\Omega_{ij}$ and $S_{ij}$ the boundary of $\Omega_{ij}$. The contour integral in (9.5.2) is approximated by, taking the part AB as an example,

$$\int_{A}^{B} g_\beta \, dS_\beta = g_{\beta(AB)} n_\beta |AB| \quad (9.5.3)$$

where $n$ is the outward normal on $S_{ij}$ and $g_{\beta(AB)}$ is a suitable approximation of $g_\beta$ on AB, on which the properties of the discretization depend strongly. Central differences may be used:

$$g_{\beta(AB)} = \frac{1}{2} [g_\beta(q_{ij}) + g_\beta(q_{i+1,j})] \quad (9.5.4)$$

resulting in second-order accuracy. In order to satisfy requirements (ii)—(iv), artificial non-linear dissipation terms must be added. This approach is followed by Jameson c.s. is a widely used set of computer codes (Jameson et al. 1981, Jameson 1985a, 1985b, 1986, 1988, Jameson and Yoon 1986), and has been adopted by many authors.

**Flux splitting**

Another widespread approach, not requiring artificial parameters, is flux splitting. First, the rotational invariance of $g_\beta$ is exploited as follows. We have

$$g_{\beta(AB)}(q) n_\beta = Q^{-1} g_{1(AB)}(Qq) \quad (9.5.5)$$
The Euler equations of gas dynamics

with the rotation matrix $Q$ defined by

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & n_1 & n_2 & 0 \\ 0 & -n_2 & -n_1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (9.5.6)$$

Next, it is assumed that $g_{1(AB)}$ depends only on the two adjacent states:

$$g_{1(AB)}(Qq) = \tilde{g}_1(Qq_{ij}, Qq_{i+1,j}) \quad (9.5.7)$$

There are several good possibilities for choosing $\tilde{g}_1$. For a survey, see Harten et al. (1983), Van Leer (1984) and Hirsch (1990). One possibility is to introduce a splitting

$$g_1 = \tilde{g}_1^+ + \tilde{g}_1^- \quad (9.5.8)$$

such that the Jacobians $\partial \tilde{g}_1^+/\partial q$ and $\partial \tilde{g}_1^-/\partial q$ have non-negative and non-positive eigenvalues, respectively. There are various ways to do this; see the literature just cited. Next, we choose

$$\tilde{g}_1(Qq_{ij}, Qq_{i+1,j}) = g_1^+(Qq_{ij}) + g_1^-(Qq_{i+1,j}) \quad (9.5.9)$$

A crude intuitive motivation of this procedure is that, as in upwind discretization of scalar convection–diffusion equations, the main diagonal is enhanced in the resulting discrete system. (cf. Exercise 9.5.1). In the linear case the matrix is an $M$-matrix, ensuring monotonicity, and allowing simple effective iterative and smoothing methods. Another way of looking at (9.5.9) is that the physical direction of the flow of information is simulated numerically; this is especially clear if $\tilde{g}_1$ is derived from a (approximate) Riemann problem solution.

The scheme resulting from (9.5.9) has first-order accuracy, is monotone, and has crisp resolution of discontinuities that are approximately aligned with the grid lines. For sharp resolution of discontinuities with general orientation adaptive local grid refinement is required, as in Bassi et al. (1988). Furthermore, the entropy condition is satisfied: the ‘one-sidedness’ of (9.5.9) implies irreversibility. Second-order discretizations may be obtained by assuming linear distribution of $q$ in each finite volume; monotonicity has to be ensured by adding non-linear ‘limiters’ (Spekreijse 1987, 1987a, Sweby 1984, Van Albada et al. 1982, Van Leer 1977). Multigrid is not directly applicable to these second-order discretizations; defect correction (Section 4.6) can be used. This has been done by Hemker (1986), Hemker et al. (1986), Koren (1988) and Koren and Speckreijse (1987, 1988). We will describe the principles of multigrid applied to flux-splitting discretizations of the Euler equations, and of defect correction.

The discretization resulting from (9.5.2), (9.5.3), (9.5.5) and (9.5.9) looks
Applications of multigrid methods in computational fluid dynamics

as follows:

\[
\frac{d}{dt} q_{ij} = N(q)_{ij} + s_{ij} \tag{9.5.10}
\]

\[
N(q)_{ij} = -\frac{1}{a_{ij}} \left[ |AB| Q_{AB}^\top \left( g^+_l (Q_{AB} q_{ij}) + g^-_l (Q_{AB} q_{i+1,j}) \right) 
+ |BC| Q_{BC}^\top \left( g^+_l (Q_{BC} q_{ij}) + g^-_l (Q_{BC} q_{i,j+1}) \right) 
+ |CD| Q_{CD}^\top \left( g^+_l (Q_{CD} q_{ij}) + g^-_l (Q_{CD} q_{i-1,j}) \right) 
+ |AD| Q_{AD}^\top \left( g^+_l (Q_{AD} q_{ij}) + g^-_l (Q_{AD} q_{i,j-1}) \right) \right] \tag{9.5.11}
\]

where \( Q_{AB} \) is the rotation matrix for cell face AB, etc.

Boundary conditions

The numerical implementation of the boundary conditions has great influence on the accuracy. Artificial numerical reflections from the boundaries are to be avoided as much as possible. The simplest (but not the best) approach is to prescribe \( q \) on the whole boundary. Due to the asymmetric differencing of \( g^+_l \) the scheme automatically selects the appropriate information. For more accurate approaches, see Hirsch (1990) Chapter 19.

Time discretization

Let us assume that the aim is to obtain steady (time-independent) solutions of (9.5.10). One way to achieve this has been proposed by Jameson \textit{et al.} (1981), namely Runge–Kutta time stepping, as described in Section 7.11. Convergence to steady state is enhanced by choosing the Runge–Kutta coefficients such as to increase the stability domain, by choosing the maximum time step allowed by stability in each finite volume separately (since the transient behaviour of \( q_{ij} \) is not of interest), and by introducing a multigrid method: time stepping takes place alternating on coarser and finer grids, driving transient waves out rapidly by the large time steps allowed on coarse grids (Jameson 1983, 1985a, 1985b, 1986, 1988, 1988a, Jameson and Baker 1984, Hall 1986). The performance of Runge–Kutta time stepping as a smoothing method was analysed in Section 7.11.

Another approach is to discretize (9.5.10) with the backward Euler method:

\[
(q^n_{ij} - q^{n+1}_{ij})/\Delta t = N(q^{n+1}_{ij}) + s^{n+1}_{ij} \tag{9.5.12}
\]

Now \( \Delta t \) is unconstrained by stability, and one may step to \( t = \infty \) in very few steps. Equation (9.5.12) may be solved by the standard non-linear multigrid method described in Chapter 8. Some publications in which this approach is taken are: Anderson and Thomas (1988), Dick (1985, 1989, 1989b, 1990),
The Euler equations of gas dynamics


Multigrid method

The grid on which equation (9.5.12) is to be solved is called $G^K$, and is the finest in a sequence of grids $G^k$, $k = 1, 2, \ldots, K$, $G^K$ finer than $G^{k-1}$. Equation (9.5.12) can be rewritten as

$$L^K (u^K) = f^K$$

(9.5.13)

with $L^K = I - \Delta t N$, $u^K = q^{n+1}$ and $f^K = q^n + \Delta t s^{n+1}$. Equation (9.5.13) may be solved by the standard non-linear multigrid algorithms described in Sections 8.3 and 8.7.

Coarse grids are constructed by cell-centred coarsening (see Section 5.1). It is assumed that the only information available about the grid geometry is the location of the cell vertices. For the cell boundaries we take straight lines; this is implicit in the finite volume discretization discussed before. Figure 9.5.2 shows four fine cells and the corresponding coarse cell. Prolongation and restriction operators $P^k$ and $R^k$ are chosen as follows. Equation (9.5.11) constitutes a first-order system, thus it follows from (5.3.18) that $P^k$ and $R^k$ are sufficiently accurate if $m_p = m_R = 1$. Inspection of (9.5.11) shows that we have $\alpha = 0$ in the scaling rule (5.3.16); hence we should have

$$\sum_{m,n} R^k ((i,j), (m,n)) = 1$$

(9.5.14)

It follows that we may choose

$$[R^k] = \frac{1}{4} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad P^k = 4(R^{k-1})^*$$

(9.5.15)

Figure 9.5.2 Fine cells and coarse cell.
Applications of multigrid methods in computational fluid dynamics

that is,

\[(R^{k-1}u^k)_{ij} = \frac{1}{4} [u_{i+1,j}^k + u_{i,j+1}^k + u_{i+1,j+1}^k + u_{i,j}^k] \quad (9.5.16)\]

and

\[(P^k u^{k-1})_{2i,2j} = (P^k u^{k-1})_{2i-1,2j} = (P^k u^{k-1})_{2i,2j-1} = (P^k u^{k-1})_{2i-1,2j-1} = u_{ij}^k \quad (9.5.17)\]

The coarse grid operators are obtained by discretizing the differential equation on the coarse grids. The problem to be solved on the coarse grids \(G^k, k < K\) can be denoted as

\[L^k(u^k) = b^k \quad (9.5.18)\]

with \(L^k = I - \Delta t N^k; N^k\) is obtained by discretizing the differential equation on \(G^k; b^k\) follows from the non-linear multigrid algorithm.

Smoothing method

A suitable smoothing method is collective Gauss–Seidel smoothing. In finite volume \((i,j)\) Equation (9.5.18) gives a non-linear algebraic relation between the unknowns in neighbouring finite volumes, which we may denote as (deleting the superscript \(k\) for brevity)

\[A(u_{ij}, u_{i+1,j}, u_{i,j+1}, \ldots) = b_{ij} \quad (9.5.19)\]

The finite volumes are visited in a predetermined sequence. In each cell \(u_{ij}\) is updated, keeping \(u\) fixed in neighbouring cells. The update may consist of a single Newton iteration. This involves solution of a linear system for the unknowns represented by \(u_{ij}\) (in the two-dimensional Euler case, these are \(\rho, \rho u_1, \rho u_2, \rho e\), cf. (9.5.1)). The adjective ‘collective’ refers to the fact that these unknowns are updated simultaneously. It may happen that Newton iteration does not converge. In that case one may decrease \(\Delta t\), which is tantamount to damping the Newton process.

The order in which the finite volumes are visited can be any of the orderings for which point-wise iteration methods are found to be robust for the convection–diffusion equation with the Fourier smoothing analysis of Chapter 7. The convection–diffusion equation is the relevant test problem, because it simulates hyperbolic behaviour as \(\epsilon \downarrow 0\). Hence, suitable methods are: four-direction point Gauss–Seidel (Section 7.7), four-direction point Gauss–Seidel–Jacobi (Section 7.7), and alternating white–black Gauss–Seidel (Section 7.10). The first method can be vectorized/parallelized to a reasonable extent by using diagonal ordering (Section 4.3), because we have the five-point stencil, which is a special case of the seven-point stencil of Figure 3.4.2(b). The last two methods vectorize and parallelize in a natural way. These point-wise methods do not work for the anisotropic diffusion equation, and as a consequence the smoothing methods just discussed fail for...
the Euler equations on grids where cells with high mesh aspect ratios occur. Then one may apply semi-coarsening (Section 7.4), decreasing mesh aspect ratios on coarser grids. Or line Gauss–Seidel methods must be used, which means that rows or columns of finite volumes are updated simultaneously, that is, taking rows for example, in (9.5.19) $u_{ij}$ and $u_{i \pm 1,j}$ are updated simultaneously, letting the other arguments of $A$ lag behind. This leads to a more complicated nonlinear system to be solved, of course, but this approach is feasible in practice. The line versions of Sections 7.7 and 7.10 that work both for the convection–diffusion and anisotropic diffusion equation should be used, of course.

**Defect correction**

The smoothers discussed before work only for flux-splitting discretization of first order. In practice however second-order discretization is usually desirable. We will not discuss second order discretization here; see Hirsch (1990) Chapter 21 for an introduction. Using the multigrid method just described, second order accuracy may be obtained by means of defect correction, described in Section 4.6. This method has been used by Hemker (1986), Hemker et al. (1986), Koren (1988), Koren and Spekreijse (1987, 1988) and Hemker and Koren (1988).

Let a first-$(m = 1)$ and second-$(m = 2)$ order spatial discretization of the Euler equations be given by (cf. (9.5.12)):

$$
(q_i^m + 1 - q_i^m)/\Delta t = N^{(m)}(q_i^m + 1) + s_i^{m+1}, \quad m = 1, 2 \tag{9.5.20}
$$

Then, instead of solving (9.5.12), the following algorithm is carried out

\begin{verbatim}
Solve (q* - q^m)/\Delta t = N^{(1)}(q*)
for i = 1 step 1 until s do
    solve (\tilde{q} - q^m)/\Delta t = N^{(1)}(q) + N^{(2)}(q*) - N^{(1)}(q*)
    q* = \tilde{q}
od
q^{m+1} = q*
\end{verbatim}

This algorithm carries out $s$ defect corrections. Usually $s$ can be taken small. With one non-linear multigrid iteration (V-cycle with one symmetric collective Gauss–Seidel pre- and postsmoothing) per defect correction Koren (1988) obtains second-order engineering accuracy after about five defect corrections for the Euler equations for two-dimensional supercritical airfoil flows. This amounts to about 14 work units (one work unit is the cost of one symmetric Gauss–Seidel iteration on the finest grid). The savings in computing time due to the use of multigrid is large in this type of application.
Exercise 9.5.1. Show that in the case of one unknown flux-splitting is equivalent to upwind discretization, by applying flux splitting discretization to equation (7.5.7) with $\epsilon = 0$.

9.6. The compressible Navier–Stokes equations

The Navier–Stokes equations for compressible flows have been presented in Section 9.2. It is convenient to write them as

$$\frac{\partial q}{\partial t} + g_{\beta, \beta} + G_{\beta, \beta} = s$$  \hspace{1cm} (9.6.1)

with $g_{\beta}$ defined after equation (9.5.1), and $G_{\beta}$ defined by

$$G_1 = (0, \sigma_{11}, \sigma_{12}, -\sigma_{1\gamma}, -\eta T_{11})^T$$

$$G_2 = (0, \sigma_{21}, \sigma_{22}, -\sigma_{2\gamma}, -\eta T_{22})^T$$  \hspace{1cm} (9.6.2)

with $\sigma_{\alpha\beta}$ defined by Equation (9.2.4). Here $g_{\beta}$ is called the inviscid flux function and $G_{\beta}$ the viscous flux function. Equation (9.6.1) is a generalization of the Euler equations (9.5.1), and numerical methods for the compressible Navier–Stokes equations generally resemble those for the Euler equations, so that not much needs to be added compared to the preceding section.

Finite volume discretization

As in the preceding section, we restrict ourselves to finite volume discretization. The inviscid terms $g_{\beta, \beta}$ can be discretized as before. A slight complication may, however, arise. At solid walls, with the Euler equations the tangential velocity component is left free, whereas with the Navier–Stokes equations it is prescribed to be zero (no-slip condition). Suppose flux-splitting (9.5.8) is employed, using the method of Van Leer (1982). This flux-splitter has the property that the no-slip condition has the effect of bringing the tangential velocity down close to zero in the vicinity of the wall. This leads to large discretization errors, because the no-slip boundary condition should influence only the viscous terms, but not the inviscid terms. Schwane and Hänel (1989) have proposed a modification of Van Leer's Euler flux-splitting that removes this defect; other Euler flux-splittings do not need to be modified for use with Navier–Stokes.

Integration of (9.6.1) over the finite volume $\Omega_{ij} = ABCD$ (Figure 9.5.1) gives (cf. (9.5.2)):

$$a_{ij} \frac{dq_{ij}}{dt} + \int_{S_{ij}} g_{\beta} \, dS_{\beta} + \int_{S_{ij}} G_{\beta} \, dS_{\beta} = a_{ij} s_{ij}$$  \hspace{1cm} (9.6.3)
The treatment of the first integral is given in the preceding section. All that remains to be done is to discretize the second integral.

**Discretization of viscous terms**

The second contour integral in (9.6.3) is approximated by, taking the part AB as an example,

\[
\int_{A}^{B} G_{\beta} \, dS_{\beta} = G_{\beta(AB)} n_{\beta} |AB| \tag{9.6.4}
\]

where \(n\) is the outward normal on \(S_{ij}\) and \(G_{\beta(AB)}\) is a suitable approximation of \(G_{\beta}\) on \(AB\), which has to be obtained by further discretization, because \(G_{\beta}\) contains derivatives. We have

\[
G_{\beta(AB)} n_{\beta} |AB| = (G_{1}\Delta x^2 - G_{2}\Delta x^1)_{AB} \tag{9.6.5}
\]

where \(\Delta x_{AB}^2 = x_{B}^2 - x_{A}^2\). It suffices to show how to handle one of the terms occurring in \(G_{\beta}\), for example \(\mu u_{1,1}\). This term is approximated as a mean value over a suitably chosen secondary finite volume surrounding \(AB\), for example EFGH (cf. Figure 9.5.1), to be denoted as \(\Omega_{i+1/2,j}\), with boundary \(S_{i+1/2,j}\) and area \(a_{i+1/2,j}\). Then we can write

\[
u_{1,1}(AB) \approx \frac{1}{a_{i+1/2,j}} \int_{\Omega_{i+1/2,j}} u_{1,1} \, d\Omega = \frac{1}{a_{i+1/2,j}} \int_{S_{i+1/2,j}} u_{1} \, dx^2 = \frac{1}{a_{i-1/2,j}} \times [(u_{1}\Delta x^2)_{EF} + (u_{1}\Delta x^2)_{FG} + (u_{1}\Delta x^2)_{OH} + (u_{1}\Delta x^2)_{HE}] \tag{9.6.6}
\]

where \(u_{1}(EF)\) is a mean value of \(u_{1}\) on \(EF\), etc., and where \(\Delta x_{2(EF)} = x_{2(E)} - x_{2(E)}\). The following approximations complete the discretization of this term

\[
u_{1}(EF) = u_{1}(\bar{i+1,j})
\]

\[
u_{1}(FG) = \frac{1}{4} (u_{1}(\bar{j}) + u_{1}(i+1,j) + u_{1}(i,j+1) + u_{1}(i+1,j+1)) \tag{9.6.7}
\]

Repeating this type of procedure for the other terms in \(G_{\beta}\) completes the discretization of (9.6.1). The resulting stencil is of nine-point type, as depicted in Figure 3.4.2(c). A seven-point stencil as given by Figure 3.4.2(a) is obtained if the interpolation in (9.6.7) is changed to (without loss of accuracy)

\[
u_{1}(FG) = \frac{1}{2} (u_{1}(i+1,j) + u_{1}(i_{j}+1)) \tag{9.6.8}
\]

and using similar suitably chosen averages for the other terms in \(G_{\beta}\). A seven-point stencil as given by Figure 3.4.2(b) is obtained if, instead of (9.6.8), one uses

\[
u_{1}(FG) = \frac{1}{2} (u_{1}(i_{j}) + u_{1}(i_{j}+1)) \tag{9.6.9}
\]
and choosing averages for the other terms in $G_B$ in a similar appropriate way. These possibilities are analogous to the options for the discretization of the mixed derivative in the rotated anisotropic diffusion problem discussed in Section 7.5. As remarked in Section 4.3, in the case of seven-point stencils, diagonal ordering is equivalent to forward ordering in Gauss–Seidel iteration; of course, there is an equivalent diagonal ordering also for backward ordering and successive orderings in other corners. Because with diagonal ordering Gauss–Seidel vectorizes along diagonals, the seven-point discretization is more amenable to Gauss–Seidel iteration than the nine-point discretization. On the other hand, we saw in Chapter 7 that typical smoothing methods tend to work better for the nine-point version of the anisotropic diffusion test case. It may be expected that this will also be so in the Navier–Stokes case. In practice mixed derivatives arise due to the use of non-orthogonal coordinates, and their role becomes significant only when the grid is highly skewed. Grid generation methods try to avoid this for reasons of accuracy.

The way in which boundary conditions are accounted for in the discretization of the viscous terms is standard and will not be discussed here.

**Time discretization**

As in the preceding section we assume that the aim is to obtain steady solutions of (9.6.3). Again, Runge–Kutta time-stepping may be used as a smoother accelerated by multigrid. Usually this approach is combined with central discretization of the inviscid terms $g_{9,8}$ accompanied by artificial diffusion terms, thus leading to a viscous version of the Euler solution methods developed by Jameson *c.s.* (see the literature cited in the preceding section). This approach is developed in Haase *et al.* (1984), Martinelli *et al.* (1986), Martinelli and Jameson (1988), Jayaram and Jameson (1988) and used by many authors.

Also widespread is the approach just described, using flux splitting for the inviscid terms. Multigrid methods for the resulting discrete version of (9.6.3), using time-discretization of the type (9.5.12), have been developed by Shaw and Wesseling (1986), Hemker and Koren (1988), Koren (1989b, 1989c, 1990, 1990a), Hanel, *et al.* (1989) and Schwane and Häänel (1989).

Second-order accuracy may be obtained with defect correction (Hemker and Koren 1988, Koren (1989b, 1989c, 1990)).

Runge–Kutta time-stepping smoothing and collective Gauss–Seidel smoothing have been compared for several flux-splitting discretizations by Häänel *et al.* (1989).

**Turbulence**

A multigrid method for the three-dimensional compressible Navier–Stokes equations with $k−\varepsilon$ turbulence modelling (Launder and Spalding 1974) has been described by Yokota (1990). An ILU factorization smoother is used. The $k−\varepsilon$ turbulence model is included on the finest grid only.
Multigrid method

The multigrid method to be employed for the compressible Navier–Stokes equations can be the same as for the Euler equations, apart from one important modification: prolongation and/or restriction must be more accurate. The Navier–Stokes equations are of second order, thus rule (5.3.18) gives

\[ m_p + m_R > 2 \]  \hspace{1cm} (9.6.10)

P is, therefore, now chosen to be linear or bilinear interpolation. The implementation is a straightforward generalization to non-uniform grids of the cell-centred prolongations discussed in Chapter 5. Referring to Figure 9.5.2 and taking bilinear interpolation as an example, a bilinear function \( a_0 + a_1 x^1 + a_2 x^2 + a_3 x^1 x^2 \) is determined that interpolates grid function values in the coarse grid cell centres \((i, j), (i + 1, j), (i, j + 1), (i + 1, j + 1)\). This function is used to determine prolongated grid function values in the fine grid cell centres, and has \( m_p = 2 \). Restriction can be defined by (9.5.15), which gives \( m_R = 1 \).

9.7. The incompressible Navier–Stokes and Boussinesq equations

The governing equations

In the incompressible case \( \rho \) is constant along streamlines. As a consequence the energy equation (9.2.5) and equation (9.2.6) are no longer needed. Assuming that the streamlines emanate from a region of constant density we have

\[ \rho = \text{constant} \]  \hspace{1cm} (9.7.1)

With this simplification the equation of mass conservation follows from (9.2.1) and (9.2.2) as

\[ u_{\alpha,\alpha} = 0 \]  \hspace{1cm} (9.7.2)

For greater generality it is assumed that the temperature \( T \) is non-uniform, and that the density of the fluid is a decreasing function of the temperature only. The derivation of a suitable mathematical model when temperature variations are not large is one of the more subtle things in fluid dynamics. If the velocity of the flow is small compared to the speed of sound, and if the temperature differences are not too large (more precisely: \( \gamma \Delta T \ll 1 \), with \( \gamma \) the thermal expansion coefficient of the fluid), it can be shown (Rayleigh 1916)
that to a good degree of approximation the density can still be taken constant, except in the vertical momentum balance, assuming we have a vertical gravity force. As a result, vertical buoyancy forces will occur in the fluid when the temperature is non-uniform. The resulting equations are called the Boussinesq equations, and are given by (taking $\nu$ constant, although in reality $\nu$ varies with $T$)

$$\frac{\partial u_\alpha}{\partial t} + (u_\beta u_\alpha)_\beta = -p_\alpha + \nu u_{\alpha,\beta\beta} + g_\gamma T_\delta 2\alpha$$  \hspace{1cm} (9.7.3)

with $\gamma$ the thermal expansion coefficient of the fluid, $g$ the acceleration of gravity, and $\delta$ the Kronecker delta. It is assumed that gravity acts in the negative $x_2$ direction. The temperature is governed by the energy equation (9.2.5), which reappears in the following form:

$$\frac{\partial T}{\partial t} + (u_\alpha T)_\alpha = -\eta T_{,\alpha\alpha}$$  \hspace{1cm} (9.7.4)

with $\eta$ the heat diffusion coefficient, taken constant.

The equations may be made non-dimensional as follows. Let $U$ be a characteristic velocity, $L$ a characteristic length and $T_0$ a characteristic temperature, and define dimensionless variables by (not changing notation for convenience)

$$x_\alpha := x_\alpha/L, \quad u_\alpha := u_\alpha/U, \quad p := p/U^2, \quad T := T/T_0, \quad t := tL/U$$  \hspace{1cm} (9.7.5)

then the dimensionless form of (9.7.3) and (9.7.4) is obtained as

$$\frac{\partial u_\alpha}{\partial t} + (u_\beta u_\alpha)_\beta = -p_\alpha + Re^{-1} u_{,\alpha\alpha} + Gr Re^2 T_{\delta 2\alpha}$$  \hspace{1cm} (9.7.6)

$$\frac{\partial T}{\partial t} + (u_\alpha T)_\alpha = -\frac{1}{Re Pr} T_{,\alpha\alpha}$$  \hspace{1cm} (9.7.7)

where $Re = UL/\nu$ is the Reynolds number, $Gr = \gamma g L^3 T/\nu^2$ is the Grashof number and $Pr = \nu/\eta$ is the Prandtl number.

**The staggered grid**

As in the applications discussed before, the success of multigrid depends strongly on the properties of the discretization. We will, therefore, give a detailed discussion of a suitable discretization method.

There is an essential difference between the compressible and the incompressible case, arising from the fact that in the present case a time-derivative is lacking for one of the unknowns, namely $p$. If the space discretization employed in the previous section is used here, artificial checkerboard type
fluctuations may occur in the numerical solution for the pressure, due to a lack of coupling between velocity components and pressure in adjacent points. For discussions of this phenomenon, see Patankar (1980) or Hirsch (1990) Section 23.3.4. The problem may be remedied by the use of a staggered grid, as introduced by Harlow and Welch (1965). Unfortunately, staggered discretization in general coordinates is a complicated affair. Here we restrict ourselves to a uniform Cartesian grid in two dimensions. The unknowns $u_1$, $u_2$, $p$ and $T$ are assigned to different grid points, as shown in Figure 9.7.1. The physical domain $\Omega$, taken to be the unit square for simplicity, is uniformly divided into square cells or finite volumes with sides of length $h$. The $u_1$ variables are located in the centres of the vertical sides, the $u_2$ variables are located in the centres of the horizontal sides, and the $p$ and $T$ variables are located in the centres of the cells. The cell with centre at $((i-1/2)h, (j-1/2)h)$ is called $\Omega_{ij}$. The variables located in the centre of $\Omega_{ij}$ and the centres of the left and lower faces are labelled $ij$, so that for example $u_{1,ij}$ is located at $((i-1)h, (j-1/2)h)$.

Finite volume discretization

The mass conservation equation (9.7.2) is integrated over $\Omega_{ij}$. This gives in straightforward fashion

$$ (u_{1,i+1,j} - u_{1,ij} + u_{2,i,j+1} - u_{2,ij})h = 0 \quad (9.7.8) $$

The momentum equation (9.7.6) in $x_1$ direction ($\alpha = 1$) is integrated over a shifted finite volume, which is again a square with sides of length $h$ and centre at the $u_{1,ij}$ point, i.e. at $((i-1)h, (j-1/2)h)$. For the time being, the steady
Applications of multigrid methods in computational fluid dynamics

Figure 9.7.2 Shifted control volume for the $u_1$ momentum.

The shifted finite volume is given in Figure 9.7.2. The result is

$$h(u_1^2 | H + u_1 u_2 | E) = -h p | H + Re^{-1} h | u_{1,1} | H + u_{1,2} | E\) \quad (9.7.9)$$

Since $u_2$ is not given in E, F, G, H further approximations have to be made.

**Hybrid scheme for convective terms**

Central approximations of $u_{1,F}^2$ is given by

$$u_{1,F}^2 = \frac{1}{2}(u_{1,i,j} + u_{1,i+1,j}) \quad (9.7.10)$$

and similarly for $u_{1,H}^2$. One obtains

$$u_{1,H}^2 = \frac{1}{2}(u_{1,i+1,j} - u_{1,i-1,j}) \quad (9.7.11)$$

which is $h$ times the standard central difference approximation of $(u_1^2)_{,1}$. Because (9.7.6) resembles a convection–diffusion equation, central approximation of the convection term $(u_{2}/u_{1})_{,1}$ may lead to numerical wiggles in the solution and to deterioration of the smoothing method if the mesh Reynolds numbers exceed 2. For the approximation of $u_{1,F}^2$ the appropriate definition of the mesh Reynolds number is

$$Re_{1,i+1/2,j} = |u_{1,i+1/2,j}| h Re \quad (9.7.12)$$

where

$$u_{1,i+1/2,j} = \frac{1}{2}(u_{1,i,j} + u_{1,i+1,j}) \quad (9.7.13)$$

The problems just mentioned may be avoided by upwind discretization. To this end (9.7.10) is replaced by

$$u_{1,F}^2 = \frac{1}{2} \left( (1 + s_{1,i+1/2,j})u_{1,i,j} + (1 - s_{1,i+1/2,j})u_{1,i+1,j} \right) \quad (9.7.14)$$

where $s_{1,i+1/2,j} = \text{sign}(u_{1,i+1/2,j})$. 
A good strategy is to use upwind approximation of \( u_{1,F}^\tau \) according to (9.7.14) if \( \text{Re}_{i+1/2,j} > 2 \) and otherwise central approximation according to (9.7.10). Convergence of iterative methods is generally enhanced by making the switch between upwind and central approximation smooth, as follows

\[
u_{1,F}^\tau = \omega_{1,i+1/2,j} u_{1,Fu}^\tau + (1 - \omega_{1,i+1/2,j}) u_{1,Fc}^\tau
\]  

(9.7.15)

with \( u_{1,Fu}^\tau \) given by (9.7.14) and \( u_{1,Fc}^\tau \) given by (9.7.10). Note that (9.7.15) can be written as

\[
u_{1,F}^\tau = \frac{1}{2} \left[ u_{1,ij}^\tau + u_{1,i+1,j}^\tau + \omega_{1,i+1/2,j} \left( \left| u_{1,ij} \right| - \left| u_{1,i+1,j} \right| \right) \right]
\]  

(9.7.16)

Here \( \omega_{1,i+1/2,j} = \omega(\text{Re}_{i+1/2,j}) \), with \( \omega(r) \) a switching function which increases from 0 to 1 in the vicinity of \( r = 2 \), and may be given for example by

\[
\begin{align*}
\omega(r) & = 0, \quad 0 \leq r < 1.9 \\
\omega(r) & = (r - 1.9)/0.1, \quad 1.9 \leq r < 2 \\
\omega(r) & = 1, \quad r \geq 2
\end{align*}
\]  

(9.7.17)

The function \( \omega(r) \) does not need to be chosen precisely this way, and it is easy to think of different prescriptions, avoiding IF statements if one so desires for purposes of vectorized computing.

In cells where the scheme has switched to upwind discretization the numerical viscosity due to the discretization error exceeds the physical viscosity. To be more precise, the local discretization error in the upwind discretization of \( (u_{1}^\tau)_{1} \) is approximately \( \frac{1}{2} u_{1,i+1/2,j} u_{1,11} \), which exceeds the physical term \( \text{Re}^{-1} u_{1,11} \) if \( \text{Re} > 2 \). The term \( \text{Re}^{-1} u_{1,11} \) may as well, therefore, be deleted under these circumstances. This we will do by multiplying the discrete approximation of \( (u_{1,1})_{F} \) (which is still to be specified) by \( 1 - \omega_{1,i+1/2,j} \). The resulting scheme is often called the hybrid scheme, and has been introduced by Spalding (1972). It is further discussed by Patankar (1980). Needless to say, the physical flow is not approximated at the true value of \( \text{Re} \) with the hybrid scheme if \( \text{Re} < 2 \), \( \alpha = 1 \) or 2. Defect correction as described in Section 4.6 may be used to approximate the physical situation for \( \text{Re} > 1 \) more closely. A second-order discretization is immediately available by putting \( \omega_{\alpha} = 0 \).

The treatment of the term \( u_{1,i+1/2,j}^\tau \) is similar to that of \( u_{1,F}^\tau \). The term \( (u_{1} u_{2})_{G} \) has to be treated a little differently, because \( u_{1} \) and \( u_{2} \) are not given in the same point. The procedure is a straightforward adaptation of what was just done for \( u_{1,F}^\tau \). We write

\[
(u_{1} u_{2})_{G} = u_{2,i-1/2,j+1} u_{1,G}
\]  

(9.7.18)

where

\[
u_{2,i-1/2,j+1} = \frac{1}{2} (u_{C} + u_{D}) = \frac{1}{2} (u_{2,i,j+1} + u_{2,i-1,j+1})
\]  

(9.7.19)
and \( u_{1,G} \) is approximated with the hybrid scheme:

\[
\begin{align*}
    u_{1,Gc} &= \frac{1}{2} (u_{1,i,j} + u_{1,i,j+1}) \\
    u_{1,Gu} &= \frac{1}{2} \{ (1 + s_{2,i-1/2,j+1})u_{1,i,j} + (1 - s_{2,i-1/2,j+1})u_{1,i,j+1} \} 
\end{align*}
\]  

(9.7.20)

with \( s_{2,i-1/2,j+1} = \text{sign}(u_{2,i-1/2,j+1}) \). We define the following mesh Reynolds number:

\[
\text{Re}_{2,i-1/2,j+1} = |u_{2,i-1/2,j+1}| h \text{ Re} 
\]  

(9.7.21)

The resulting hybrid approximation of \((u_1u_2)_G\) can be written as

\[
(u_1u_2)_G = \frac{1}{2} \{ u_{2,i-1/2,j+1}(u_{1,i,j} + u_{1,i,j+1}) \\
    + \omega_{2,i-1/2,j+1} \{ u_{2,i-1/2,j+1} |(u_{1,i,j} - u_{1,i,j+1})| \} \} 
\]  

(9.7.22)

where \( \omega_{2,i-1/2,j+1} = \omega(\text{Re}_{2,i-1/2,j+1}) \). The viscous flux \((u_1)_G\) is multiplied by \( \omega - \omega_{2,i-1/2,j+1} \), if the hybrid scheme is applied.

Note that upwind approximation is not applied to \( u_2 \), but to \( u_1 \). This is as it should be, since in the convection–diffusion-like Equation (9.7.6) with \( \alpha = 1 \), \( u_1 \) is to be regarded as unknown, and \( u_2 \) is to be regarded as (and will be in the iterative method to be described) a known coefficient.

De Henau et al. (1989) have proposed a method to improve the accuracy of the pressure when upwind discretization is used.

**Linearization of convection terms**

In iterative solution methods the convection terms are to be linearized. In the framework of the non-linear multigrid algorithm a natural way to do this is as follows. Before smoothing starts an approximate solution \( \tilde{u}_\alpha \) has already been generated by the non-linear multigrid algorithm. Equations (9.7.16) and (9.7.22) are replaced by

\[
\begin{align*}
    u_{1,F}^I &= \frac{1}{2} \{ \tilde{u}_{1,i,j}u_{1,i,j} + \tilde{u}_{1,i,j+1}u_{1,i,j+1} \\
    &\quad + \omega_{1,i+j/2,j}(\tilde{u}_{1,i,j} |u_{1,i,j} - |\tilde{u}_{1,i,j+1}|u_{1,i,j+1}) \} 
\end{align*}
\]  

(9.7.23)

and

\[
\begin{align*}
    (u_1u_2)_G &= \frac{1}{2} \{ \tilde{u}_{2,i-1/2,j+1}(u_{1,i,j} + u_{1,i,j+1}) \\
    &\quad + \omega_{2,i-1/2,j+1} |\tilde{u}_{2,i-1/2,j+1}|(u_{1,i,j} - u_{1,i,j+1}) \} 
\end{align*}
\]  

(9.7.24)

and \( \text{Re}_\alpha \) is evaluated using \( \tilde{u}_\alpha \).
Approximation of the remaining terms

The pressure term in (9.7.9) can be maintained as it stands:

\[ -hp \big|_{H} = -h(p_{ij} - p_{i-1,j}) \]  

(9.7.25)

For \((u_{1,1})_F\) one takes of course

\[ (u_{1,1})_F = (1 - \omega_{1,i+1/2,j})(u_{1,i+1,j} - u_{1,ij})/h \]  

(9.7.26)

and similarly for the remaining viscous terms.

The equation for \(u_2\) \((\alpha = 2\) in (9.7.6)) can be discretized in the same way. Now finite volume integration takes place over a control volume that is shifted vertically, with centre at the point where \(u_{2,ij}\) is located. An additional buoyancy term \(\frac{1}{2} \text{Gr} \Re^{-2}(T_{ij} + T_{i,j-1})h^2\) appears in the right-hand side. Space discretization of the temperature Equation (9.7.7) takes place by integration over the control volumes \(\Omega_{ij}\) defined for the mass conservation equation. The convection term is again approximated by the hybrid scheme, according to the principles just discussed. Details are left to the reader.

Boundary conditions

This not being a text on computational fluid dynamics, it would lead too far to discuss all possible boundary conditions that occur in practice. For brevity it is assumed that the velocity is prescribed on the boundary. Let the \(u_1\) cell \(ABCD\) of Figure 9.7.2 lie at the lower boundary, i.e. \(AB\) is part of the boundary, where \(u_a\) is given. Where the interior scheme asks for \(u_{1,\partial}\), this is eliminated using

\[ u_{1,\partial} = 2u_{1,E} - u_{1,ij} \]  

(9.7.27)

The \(u_2\) equation is handled similarly. The temperature may be either prescribed at the wall (Dirichlet condition), or the wall may be thermally insulated:

\[ \partial T/\partial n = 0 \]  

(9.7.28)

(homogeneous Neumann condition). Other cases will not be considered. In the Dirichlet case one proceeds in the same way as for the \(u_1\) equation. In the Neumann case one has to approximate \(T,2\) at the boundary, which is simply replaced by 0, of course.
Time discretization

Introductions to methods suitable for the approximation of time-dependent solutions may be found in Fletcher (1988) and Hirsch (1990). Here we will restrict ourselves to the steady case, where the pay-off of multigrid is greatest.

Summary of the discrete equations

The discretized Boussinesq equations can be summarized as follows. The system of equations can be written as

\[
\begin{align*}
Q_{(\alpha)}(\bar{u})u_{\alpha,ij} + G_{\alpha}p_{ij} - F_{\alpha}T_{ij} &= s_{\alpha} \quad \alpha = 1, 2 \quad (9.7.29) \\
Q_{(3)}(\bar{u})T_{ij} &= s_{3} \quad (9.7.30) \\
G_{\alpha}^*u_{\alpha,ij} &= s_{4} \quad (9.7.31)
\end{align*}
\]

where the source terms \( s_{\alpha} \), \( s_{3} \) and \( s_{4} \) arise from the boundary conditions, and the notation \( (\alpha) \) indicates that the summation convention does not apply. The operators in these equations are defined as follows. The equations resulting from the finite volume procedure are scaled with appropriate powers of \( h \), such that the operators in (9.7.29) to (9.7.31) approximate the differential operators occurring in the Boussinesq equations. We have, according to (9.7.9) (after scaling by \( 1/h^2 \)):

\[
G_{1}p_{ij} = (p_{ij} - p_{i-1,j})/h, \quad G_{2}p_{ij} = (p_{i,j} - p_{i,j-1})/h \quad (9.7.32)
\]

As already suggested by the notation, \( G_{\alpha}^* \) is the adjoint (transpose) of \( G_{\alpha} \) (to show this is left to the reader) and is given by

\[
G_{1}^*u_{1,ij} = (u_{1,ij} - u_{1,i+1,j})/h, \quad G_{2}^*u_{2,ij} = (u_{2,ij} - u_{2,i+1,j})/h \quad (9.7.33)
\]

Furthermore,

\[
F_{\alpha}T_{ij} = \frac{h^2}{2} \delta_{\alpha2} \text{Gr Re}^{-2}(T_{ij} + T_{i,j-1}) \quad (9.7.34)
\]

and

\[
Q_{(\alpha)}(\bar{u})u_{\alpha,ij} = C_{(\alpha)}(\bar{u})u_{\alpha,ij} + D_{(\alpha)}(\bar{u})u_{\alpha,ij} \quad (9.7.35)
\]
The incompressible Navier-Stokes and Boussinesq equations

The temperature equation (9.7.30) can be similarly split in a convection part and a diffusion part. The convection part is given by

$$
C(α)(u)u_{1,i,j} = \frac{1}{2h} \left( \left( \tilde{u}_{1,i,j} \right)^{1/2} + \omega_{1,i+1/2,j}(\tilde{u}_{1,j} / u_{1,j})^{1/2} \right) \\
+ \omega_{1,i-1/2,j}(\tilde{u}_{1,j} / u_{1,j})^{1/2} \\
+ \tilde{u}_{2,i-1/2,j+1}(u_{1,j} + u_{1,i,j+1}) - \tilde{u}_{2,i-1/2,j}(u_{1,j} + u_{1,i,j-1}) \\
+ (\omega_{2} / \tilde{u}_{2,j})(\tilde{u}_{1,j} + 1 / u_{1,j}) + (\omega_{2} / \tilde{u}_{2,j})(\tilde{u}_{1,j})^{1/2} (9.7.36)
$$

$$
C(2)(u)u_{2,i,j} = \frac{1}{2h} \left( \tilde{u}_{1,i+1,j-1/2}(u_{2,j} + u_{2,i,j}) \\
- \tilde{u}_{1,i,j-1/2}(u_{2,j} + u_{2,i-j}) + (\omega_{1} / \tilde{u}_{1,j})(\tilde{u}_{1,j})^{1/2} \\
+ (\omega_{1} / \tilde{u}_{1,j})(\tilde{u}_{1,j})^{1/2} + (\omega_{2}^{2})(\tilde{u}_{1,j})^{1/2} \\
+ \omega_{2,i,j+1/2}(\tilde{u}_{2,j} / u_{1,j})^{1/2} + \omega_{2,i,j-1/2}(\tilde{u}_{2,j} / u_{1,j})^{1/2} (9.7.37)
$$

The diffusion terms are represented by $D(α)(u)u_{α,i,j}$ with

$$
D(1)(u)u_{1,i,j} = \frac{1}{h^{2} Re}(1 - \omega_{1,i+1/2,j})u_{1,j}^{1/2} + (1 - \omega_{1,i,j-1/2})u_{1,j}^{1/2} \\
+ (1 - \omega_{2,i+1/2,j})u_{1,i,j+1}^{1/2} + (1 - \omega_{2,i,j-1/2})u_{1,i,j-1}^{1/2} (9.7.38)
$$

$$
D(2)(u)u_{2,i,j} = \frac{1}{h^{2} Re}(1 - \omega_{1,i+1,j-1/2})u_{2,j}^{1/2} + (1 - \omega_{1,i,j-1/2})u_{2,j}^{1/2} \\
+ (1 - \omega_{2,i+1,j})u_{2,i,j+1}^{1/2} + (1 - \omega_{2,i,j-1/2})u_{2,i,j-1}^{1/2} (9.7.39)
$$

The temperature equation (9.7.30) can be similarly split in a convection part and a diffusion part. The convection part is given by

$$
C(3)(u)T_{ij} = \frac{1}{2h} \left( \tilde{u}_{1,i+1,j}(T_{ij} + T_{i+1,j}) - \tilde{u}_{1,i,j}(T_{ij} + T_{i-1,j}) + (\omega_{1} / \tilde{u}_{1,j})(T_{ij} + T_{i+1,j}) \\
+ (\omega_{1} / \tilde{u}_{1,j})(T_{ij} + T_{i,j+1}) + \tilde{u}_{2,i,j+1}(T_{ij} + T_{i,j}) \\
- \tilde{u}_{2,i,j}(T_{ij} + T_{i,j-1}) + (\omega_{2} / \tilde{u}_{2,j})(T_{ij} + T_{i,j+1}) + (\omega_{2} / \tilde{u}_{2,j})(T_{ij})^{1/2} (9.7.40)
$$

The derivation of the diffusion part $D(3)$ is left to the reader.

Further remarks on the discretization of the incompressible Navier–Stokes equations

The main advantages of the hybrid scheme and the staggered grid just described are accuracy, stability, suitability for various iteration methods.
including multigrid, and the fact that this discretization is free of artificial parameters. A disadvantage of the staggered grid is that we have no unknown vector quantities in the grid points, but only components of vectors. This encumbers the formulation in general coordinates, which is why we have specialized to a Cartesian grid here. Work is, however, in progress on staggered grid formulations in general coordinates; see for example Demirdzic et al. (1987), Rosenfeld et al. (1988), Katsuragi and Ukai (1990), and Mynett et al. (1991). Discretization in general coordinates is easier if all unknowns are assigned to the same grid points (colocated approach). A colocated approach can be followed by introducing artificial compressibility, modifying (9.7.2) to

\[ \beta^2 \frac{\partial p}{\partial t} + u_{\alpha,\alpha} = 0 \quad (9.7.41) \]

For a discussion of this method, see Fletcher (1988) and Hirsch (1990). The temporal behaviour of the solution makes no physical sense if \( \beta \neq 0 \), but when steady state is reached a physical solution is approximated. Unfortunately, the convergence of methods to iterate to steady state depends strongly on \( \beta \). Furthermore, when steady state is reached the solution may contain unphysical fluctuations. With \( \beta = 0 \) the colocated approach may still be followed if certain derivatives are approximated by one-sided differences, or artificial averaging terms are added. Publications where this approach is compared with the staggered formulation are Fuchs and Zhao (1984) and Peric et al. (1988). The price paid is loss of accuracy, and dependence on artificial parameters.

Another approach has been proposed by Dick (1988, 1988a, 1989a) and Dick and Linden (1990), consisting of a flux-splitting discretization on a colocated grid, in the spirit of the compressible case. This discretization is stable and allows efficient iterative solution methods, but is only first-order accurate.

There are many publications using the staggered and the colocated formulations; we refrain from giving a survey. Both approaches are in widespread use.

Of course, it would be very attractive to be able to handle both the incompressible and the compressible case by a unified method. A recent attempt in this direction is described by Demirdzic et al. (1990); see this paper for further references to the literature. The staggered formulation is used. We will not go into this further.

**Distributive iteration**

We will now turn to multigrid methods for solving (9.7.29) to (9.7.31). The special mathematical nature of the incompressible Navier–Stokes equations, which led us to the use of the staggered grid formulation, also necessitates the use of special smoothing methods, for example of the distributive iteration type introduced in Section 4.6. As a consequence, we will have more to say about smoothing methods than in the compressible case.
The incompressible Navier-Stokes and Boussinesq equations

The system of discrete equations (9.7.29) to (9.7.31) can be presented as

\[
\begin{pmatrix}
Q_{(1)} & 0 & 0 & G_1 \\
0 & Q_{(2)} & -F_2 & G_2 \\
0 & 0 & Q_{(3)} & 0 \\
G_1^* & G_2^* & 0 & 0
\end{pmatrix}
\begin{pmatrix}
u_1 \\
u_2 \\
T \\
p
\end{pmatrix}
= 
\begin{pmatrix}
s_1 \\
s_2 \\
s_3 \\
s_4
\end{pmatrix}
\text{ (9.7.42)}
\]

The system (9.7.42) may be further abbreviated as

\[
Ay = b
\text{ (9.7.43)}
\]

If the unknowns are ordered linearly the operator \( A \) may be identified with its matrix representation, but where convenient \( A \) will also be regarded as a (finite difference) operator, so that it is meaningful to say, for example, \( A \) equals zero in the interior.

Clearly, a classical splitting \( A = M - N \) with \( M \) regular and easy to invert is not possible, because of the occurrence of a zero block on the main diagonal in (9.7.42). Therefore smoothing methods for (9.7.42) cannot be of the basic iterative type discussed in Chapter 4. The smoothers that have been proposed are of the distributive type discussed in Section 4.6, that is, the system (9.7.43) is postconditioned by a matrix \( B \) and the resulting system is split:

\[
AB = M - N \text{ (9.7.44)}
\]

As shown in Section 4.6 the iterative method becomes

\[
y^{m+1} = y^m + BM^{-1}(b - Ay^m)
\text{ (9.7.45)}
\]

The matrices \( B \) in (9.7.44) and (9.7.45) need not be the same.

The distributive smoothers that have appeared in the literature are usually presented in various \textit{ad hoc} ways, but fit in the framework given by (9.7.44) and (9.7.45), as shown by Hackbusch (1985) and Wittum (1986, 1989b, 1990, 1990a, 1990b). The advantage of the formulation in terms of splitting of a postconditioned operator is that this creates a common framework for the various methods, facilitates analysis, makes the consistency of these methods obvious, and makes it easy to introduce modifications that do not violate consistency. However, identifying the operators \( B \) and \( M \) corresponding to the methods proposed in the literature can be somewhat of a puzzle. We will, therefore, do this for several methods. Most of these have been formulated for simplified versions of (9.7.29), such as the Stokes or Navier–Stokes equations, but generalization to the Boussinesq equations (9.7.42) is straightforward.
Distributive Gauss–Seidel smoothing method

This method has been introduced by Brandt and Dinar (1979) and formulated in the form (9.7.45) by Hackbusch (1985). We choose the following postconditioning operator:

\[
B = \begin{pmatrix}
1 & 0 & 0 & G_1 \\
0 & 1 & 0 & G_2 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & G_α^*G_α
\end{pmatrix}
\]  
(9.7.46)

This gives

\[
AB = \begin{pmatrix}
Q_{(1)} & 0 & 0 & Q_1 G_1 + G_1 G_α^*G_α \\
0 & Q_{(2)} - F_2 & 0 & Q_2 G_2 + G_2 G_α^*G_α \\
0 & 0 & Q_{(3)} & 0 \\
G_1^* & G_2^* & 0 & G_α^*G_α
\end{pmatrix}
\]  
(9.7.47)

Note that the zero diagonal block has disappeared.

For the Stokes equations (obtained by deleting the unknown \( T \) and the convection terms) the first two elements of the last column vanish in the interior; the proof is left as an exercise. This suggests the following splitting \( AB = M - N \):

\[
M = \begin{pmatrix}
P_1 & 0 & 0 & 0 \\
0 & P_2 & -F_2 & 0 \\
0 & 0 & P_3 & 0 \\
G_1^* & G_2^* & 0 & R
\end{pmatrix}
\]  
(9.7.48)

where \( P_1, P_3 \) and \( R \) define further splittings of \( Q_1, Q_{(3)} \) and \( G_α^*G_α \) such that \( My = c \) is easily solvable. For clarity we present a possible method in full. The basic algorithm is given by (9.7.45). We have

\[
b - Ay^m = \begin{pmatrix}
s_1 \\
s_2 \\
s_3 \\
s_4
\end{pmatrix} - \begin{pmatrix}
Q_{(1)} & 0 & 0 & G_1 \\
0 & Q_{(2)} - F_2 & 0 & G_2 \\
0 & 0 & Q_{(3)} & 0 \\
G_1^* & G_2^* & 0 & 0
\end{pmatrix} \begin{pmatrix}
u_1^m \\
u_2^m \\
T^m \\
p^m
\end{pmatrix} = \begin{pmatrix}
r_1 \\
r_2 \\
r_3 \\
r_4
\end{pmatrix}
\]  
(9.7.49)

A temperature correction \( δT \) is computed by solving

\[
P_3 δT = r_3 \]  
(9.7.50)
Preliminary velocity corrections $\delta \tilde{u}_\alpha$ are computed by solving

$$
\begin{pmatrix}
P_1 & 0 \\
0 & P_2
\end{pmatrix}
\begin{pmatrix}
\delta \tilde{u}_1 \\
\delta \tilde{u}_2
\end{pmatrix} =
\begin{pmatrix}
r_1 \\
r_2 + F_2 \delta T
\end{pmatrix}
$$

(9.7.51)

Next, a preliminary pressure correction $\delta \tilde{p}$ is computed by solving

$$
R \delta \tilde{p} = r_4 - G^*_\alpha \delta \tilde{u}_\alpha
$$

(9.7.52)

As prescribed by (9.7.45) and (9.7.46) the final velocity and pressure corrections are obtained as

$$
\delta u_\alpha = \delta \tilde{u}_\alpha + G_\alpha \delta \tilde{p}
$$

$$
\delta p = G^*_\alpha G_\alpha \delta \tilde{p}
$$

(9.7.53)

The iteration step is completed by $u_\alpha^{m+1} = u_\alpha^m + \delta u_\alpha$, $T^{m+1} = T^m + \delta T$, $p^{m+1} = p^m + \delta p$.

In the distributive Gauss–Seidel method of Brandt and Dinar (1979) $P_\alpha$ corresponds to point Gauss–Seidel iteration, and $R = \text{diag}(G^*G)$, corresponding to point Jacobi, but other choices are possible, of course. Fourier smoothing analysis of the distributive Gauss–Seidel method is described by Brandt and Dinar (1979). The smoothing factor is found to be $\tilde{\rho} = 1/2$ for the Stokes equation.

**Distributive ILU smoothing method**

This method has been introduced by Wittum (1986, 1989b). The postconditioning operator $B$ is the same as for the preceding method, but the splitting of $AB$ (given by (9.7.47) is provided by ILU factorization:

$$
AB = LU - N
$$

(9.7.54)


**SIMPLE method**

The SIMPLE method (Semi-Implicit Method for Pressure-Linked Equations) has been introduced by Patankar and Spalding (1972) and is discussed in
Applications of multigrid methods in computational fluid dynamics

detail in Patankar (1980). This method is obtained by choosing

\[ \mathbf{B} = \begin{pmatrix}
I & 0 & 0 & -S_1^{-1}G_1 \\
0 & I & 0 & -S_2^{-1}G_2 \\
0 & 0 & I & 0 \\
0 & 0 & 0 & I
\end{pmatrix} \quad (9.7.55) \]

where \( S_\alpha^{-1} \) is an easy to evaluate approximation of \( Q(\alpha) \). This yields

\[ \mathbf{A}\mathbf{B} = \begin{pmatrix}
Q(1) & 0 & 0 & G_1 - Q(1)S_1^{-1}G_1 \\
0 & Q(2) & -F_2 & G_2 - Q(2)S_2^{-1}G_2 \\
G_1^* & G_2^* & 0 & -G_1^*S_1^{-1}G_1 - G_2^*S_2^{-1}G_2
\end{pmatrix} \quad (9.7.56) \]

An appropriate splitting \( \mathbf{AB} = \mathbf{M} - \mathbf{N} \) is defined by (9.7.48) where now \( \mathbf{R} \) is an appropriate splitting of \( -G_1^*S_1^{-1}G_1 - G_2^*S_2^{-1}G_2 \). Depending on the choice of \( P_\alpha, S_\alpha, P_3 \) and \( \mathbf{R} \) various variants of the SIMPLE method are obtained. The algorithm proceeds as follows. First, \( \delta T, \delta \tilde{u}_\alpha \) and \( \delta \tilde{p} \) are computed as before, except that \( \mathbf{R} \) is different. In the original SIMPLE method one chooses \( S_\alpha = \text{diag}(Q(\alpha)) \). This makes \( G_1^*S_1^{-1}G_1 + G_2^*S_2^{-1}G_2 \) easy to determine; it has a five-point stencil to which a suitable iteration may be applied, such as point- or line Gauss–Seidel, thus determining \( \mathbf{R} \). The iteration is completed with the distribution step, according to

\[ \delta u_\alpha = \delta \tilde{u}_\alpha - \omega_\alpha S_\alpha^{-1}G_\alpha \delta \tilde{p} \quad (\text{no summation}) \quad (9.7.57) \]

\[ \delta p = \omega_p \delta \tilde{p} \quad (9.7.58) \]

where \( \omega_\alpha \) and \( \omega_p \) are relaxation parameters.

The Fourier smoothing factor of this type of smoothing method has been determined by Shaw and Sivaloganathan (1988) for the Navier–Stokes equations. For \( \text{Re} = 1 \), which is close to the Stokes equations, they find \( \rho = 0.62 \). On the basis of multigrid experiments, Sivaloganathan and Shaw (1988) advise to take \( \omega_\alpha = 0.5, \omega_p = 1 \).

An ILU variant is obtained by using ILU factorization for (9.7.56). This has been explored by Wittum (1990b), who finds, however, that distributive ILU based on (9.7.47) is more efficient.

Symmetric coupled Gauss–Seidel method

This smoothing method has been proposed by Vanka (1986). The symmetric coupled Gauss–Seidel (SCGS) method is best explained without using the framework of distributive iteration (but see Wittum (1990) for a description
as a 'local' distributive method). Each cell is visited in turn in some prescribed order. The six unknowns associated with \( Q_{ij} \), namely \( u_{1,ij}, u_{1,i+1,j}, u_{2,ij}, u_{2,i,j+1}, T_{ij} \) and \( p_{ij} \) are updated simultaneously. Hence, at a given stage during the course of an iteration, some variables have already been updated, others not, similar to Gauss–Seidel iteration. Note that the velocity variables are associated with two cells (for example, \( u_{1,ij} \) belongs to \( Q_{i-1,j} \) and \( Q_{ij} \)). Hence they are updated twice during an iteration. Let the residual before the update of \( Q_{ij} \) be given by

\[
\begin{pmatrix}
    r_1 \\
    r_2 \\
    r_3 \\
    r_4 \\
\end{pmatrix} =
\begin{pmatrix}
    s_1 \\
    s_2 \\
    s_3 \\
    s_4 \\
\end{pmatrix} -
\begin{pmatrix}
    Q^{(1)} & 0 & 0 & G_1 \\
    0 & Q^{(2)} & -F_2 & G_2 \\
    0 & 0 & Q^{(3)} & 0 \\
    G_1^* & G_2^* & 0 & 0 \\
\end{pmatrix}
\begin{pmatrix}
    \delta u_1 \\
    \delta u_2 \\
    \delta T \\
    \delta p \\
\end{pmatrix}
\]  

where \( (\delta u_1, \delta u_2, \delta T, \delta p)^T \) represents the current approximate solution. The correction \( (\delta u_1, \delta u_2, \delta T, \delta p) \) required to obtain the final solution satisfies

\[
\begin{pmatrix}
    Q^{(1)} & 0 & 0 & G_1 \\
    0 & Q^{(2)} & -F_2 & G_2 \\
    0 & 0 & Q^{(3)} & 0 \\
    G_1^* & G_2^* & 0 & 0 \\
\end{pmatrix}
\begin{pmatrix}
    \delta u_1 \\
    \delta u_2 \\
    \delta T \\
    \delta p \\
\end{pmatrix} =
\begin{pmatrix}
    r_1 \\
    r_2 \\
    r_3 \\
    r_4 \\
\end{pmatrix}
\]  

In Gauss–Seidel fashion, the correction is put zero in all cells except \( Q_{ij} \). This results in a local \( 6 \times 6 \) system for the six unknowns associated with \( Q_{ij} \) which may be denoted by

\[
\begin{pmatrix}
    a_{1,ij} & a_{1,i+1,j} & 0 & 0 & 0 & h^{-1} \\
    a_{2,ij} & a_{2,i+1,j} & 0 & 0 & 0 & h^{-1} \\
    0 & 0 & a_{3,ij} & a_{3,i,j+1} & b_{1,ij} & h^{-1} \\
    0 & 0 & a_{4,ij} & a_{4,i,j-1} & b_{1,i,j+1} & h^{-1} \\
    h^{-1} & h^{-1} & h^{-1} & h^{-1} & 0 & 0 \\
\end{pmatrix}
\begin{pmatrix}
    \delta u_{1,ij} \\
    \delta u_{1,i+1,j} \\
    \delta u_{2,ij} \\
    \delta u_{2,i,j+1} \\
    \delta T_{ij} \\
    \delta p_{ij} \\
\end{pmatrix} =
\begin{pmatrix}
    r_{1,ij} \\
    r_{1,i+1,j} \\
    r_{2,ij} \\
    r_{2,i,j+1} \\
    r_{3,ij} \\
    r_{4,ij} \\
\end{pmatrix}
\]  

The values of the coefficients in (9.7.61) are easily deduced from (9.7.32) to (9.7.40). The system is simplified by dropping some terms, and damping is
Applications of multigrid methods in computational fluid dynamics

introduced by dividing the diagonal elements by damping factors. The system replacing (9.7.61) is given by, solving $\delta T_{ij}$ from (9.7.61),

$$
\begin{pmatrix}
1 + a_{1,ij}/\sigma_1 & 0 & 0 & 0 \\
0 & 1 + a_{2,ij}/\sigma_1 & 0 & 0 \\
0 & 0 & 1 + a_{3,ij}/\sigma_2 & 0 \\
(-h^{-1}) & (-h^{-1}) & (-h^{-1}) & 1 + a_{4,ij}/\sigma_2
\end{pmatrix}
\begin{pmatrix}
\delta u_{1,ij} \\
\delta u_{1,i+1,j} \\
\delta u_{2,ij} \\
\delta u_{2,i+1,j+1}
\end{pmatrix}
= 
\begin{pmatrix}
r_{1,ij} \\
r_{1,i+1,j} \\
r_{2,ij} - b_{1,ij} \delta T_{ij} \\
r_{2,i+1,j+1} - b_{1,i+1,j+1} \delta T_{ij} \\
r_{4,ij}
\end{pmatrix}
(9.7.62)
$$

This system can be written in the following partitioned form

$$
\begin{pmatrix}
A_1 & A_2 \\
A_2^T & 0
\end{pmatrix}
\begin{pmatrix}
U_1 \\
U_2
\end{pmatrix}
= 
\begin{pmatrix}
b_1 \\
b_2
\end{pmatrix}
(9.7.63)
$$

and is solved by the following explicit formula:

$$
U_1 = A_1^{-1}(b_1 - A_2 U_2) \quad U_2 = (A_2^T A_1^{-1} b_1 - b_2)/A_2^T A_1^{-1} A_2
(9.7.64)
$$

We put $\hat{u}_{1,ij} := u_{1,ij} + \delta u_{1,ij}$ etc., recompute the elements of $r_1, r_2, r_3, r_4$ that are affected by the update of $\hat{u}_\alpha$ and $T$, and proceed with the next cell. The method is called symmetric because the four velocity variables associated with a cell are treated the same way, it is called a coupled method because the six unknowns associated with a cell are updated simultaneously, and it is called a Gauss–Seidel method because the cells are visited sequentially in Gauss-Seidel fashion.

Suitable values for the underrelaxation factors $\sigma_\alpha$ must be determined empirically. Usually one can take $\sigma_1 = \sigma_2$, and optimum values are found to vary between 0.5 and 0.8 (Vanka 1986), decreasing with increasing Reynolds number. Wittum (1990) finds, however, that with $\sigma_\alpha = 1$ SCGS is still an acceptable smoother. This paper also shows that it does not pay to solve (9.7.61) instead of (9.7.62).

Fourier smoothing analysis results for the SCGS method are presented by Shah et al. (1990). For $\text{Re} = 1$ they find $p = 0.32$ with $\sigma_1 = \sigma_2 = 0.7$. These authors also present a more efficient version, in which the pressure variables in rows or columns of cells are solved in a coupled manner (but not the velocities) by solving tridiagonal systems. Wittum (1990) gives numerical multigrid results comparing distributive ILU variants with SCGS, finding that ILU is a little more efficient. Sivaloganathan et al. (1988) find SCGS to be more efficient than SIMPLE smoothing.
Further remarks on smoothing methods

The temperature equation is a convection-diffusion equation, but also the momentum equations are basically of this type. This is reflected in the iterative methods just discussed. For example, the operators $\mathbf{P}_\alpha, \mathbf{P}_3$ in (9.7.48) correspond to an iteration method for a single convection-diffusion equation, so that the smoothing analysis presented in Chapter 7 carries over directly. Keeping in mind that flow direction is variable and that mesh Péclet numbers are often large in fluid dynamics, it follows that $\mathbf{P}_\alpha, \mathbf{P}_3$ should correspond to a robust smoothing method for the convection-diffusion equation, a number of which have been identified in Chapter 7. When large mesh aspect ratios occur $\mathbf{P}_\alpha, \mathbf{P}_3$ should also be robust for the anisotropic diffusion equation, unless semi-coarsening is used.

In Vanka’s method the equations remain coupled, and no single convection-diffusion is operated on during an iteration. The lessons learned in Chapter 7, however, carry over qualitatively. For example, the order in which the cells are visited with the SCGS method should be such that when this order is used in a point Gauss–Seidel method for the convection-diffusion test problem, we have a smoother. When large mesh aspect ratios occur all unknowns (not just the pressure, as in the SCGS version proposed by Shah et al. (1990)) in rows and/or columns of cells must be updated simultaneously; for distributive ILU no change is required. This leads to simple tridiagonal systems, except in the case of SCGS, where the system for the unknowns in rows or columns is more involved; however, Thompson and Ferziger (1989) report an increase of only 50% in computing time per sweep as compared to cell-wise SCGS.

The remarks made in Chapters 4 and 7 on vectorized and parallelized computing also carry over to the present case, at least qualitatively. Vector and parallel implementation of the SCGS method is discussed by Vanka and Misegades (1986), who propose to visit the cells in the white–black Gauss–Seidel order given in Section 4.3.

Coarse grid approximation

It suffices to consider only one coarse grid. Coarse grid quantities are denoted by an overbar. The coarse grid cells are obtained by taking unions of fine grid cells (cell-centred coarsening, cf. Section 5.1), as follows:

$$\bar{\Omega}_{ij} = \Omega_{2i,2j} \cup \Omega_{2i-1,2j} \cup \Omega_{2i,2j-1} \cup \Omega_{2i-1,2j-1}$$  \hspace{1cm} (9.7.65)$$

cf. Figure 9.7.3. The coarse grid equations are obtained by discretizing the differential equations on $\bar{G}$ in the same way as on $G$. 
The transfer operators are assumed to be of block-diagonal form. That is, for example, with $r = (r_1, r_2, r_3, r_4)^T$ defined by (9.7.49), we have

$$Rr = (R_1r_1, R_2r_2, R_3r_3, R_4r_4)^T \quad (9.7.66)$$

and similarly for prolongation, writing $y = (u_1, u_2, T, p)^T$:

$$Py = (P_1u_1, P_2u_2, P_3T, P_4p)^T \quad (9.7.67)$$

The accuracy rule for the transfer operators (5.3.18) generalizes to our system as follows:

$$m_p + m_r > 2m_s, \quad s = 1, 2, 3, 4 \quad (9.7.68)$$

where $2m_s$ is the order of differential equation number $s$:

$$2m_1 = 2m_2 = 2m_3 = 2, \quad 2m_4 = 1 \quad (9.7.69)$$

The theory developed by Wittum (1990a) assumes accuracy of higher order than prescribed by (9.7.68), but (9.7.68) is found to suffice in practice.

**Restriction**

Since the residuals may be regarded as integrals over finite volumes, a natural way to define $R_s r_s, \ s = 1, 2, 3$ or 4 is to add contributions from the appropriate fine grid cells, followed by scaling following the scaling rule (5.3.16). Let us call the shifted finite volume for the $u_{1,j}$ momentum equation $\bar{\Omega}_{u_{1/2,j}}$ (cf. Figure 9.7.2). This consists of the unions of the following shifted fine grid

---

**Figure 9.7.3** Coarse and fine grid cells.
The incompressible Navier–Stokes and Boussinesq equations

finite volumes: $\Omega_{2i-3/2,2j}$, $\Omega_{2i-3/2,2j-1}$ and half of $\Omega_{2i-5/2,2j}$, $\Omega_{2i-5/2,2j-1}$, $\Omega_{2i-1/2,2j}$ and $\Omega_{2i-1/2,2j-1}$. This gives

\[(R_1r_1)_{ij} = \frac{1}{4} \{ r_{1,2i-1,2j} + r_{1,2i-1,2j-1} + \frac{1}{2} (r_{1,2i-2,2j} + r_{1,2i-2,2j-1} + r_{1,2i,2j} + r_{1,2i,2j-1}) \} \quad (9.7.70)\]

Similarly, one obtains

\[(R_2r_2)_{ij} = \frac{1}{4} \{ r_{2,2i-1,2j-1} + r_{2,2i,2j-1} + \frac{1}{2} (r_{2,2i-1,2j-2} + r_{2,2i-1,2j} + r_{2,2i,2j}) \} \quad (9.7.71)\]

\[(R_3r_3)_{ij} = \frac{1}{4} (r_{3,2i,2j} + r_{3,2i-1,2j} + r_{3,2i,2j-1} + r_{3,2i-1,2j-1}) \quad (9.7.72)\]

for $s = 3, 4$. This defines the restriction operator in the non-linear two-grid algorithm TG presented in Section 8.2. If the approximate coarse grid solution $\tilde{u}$ occurring in TG is obtained by (8.2.5) then an additional restriction operator $\tilde{R}$ is required, operating not on the residuals but on the unknowns. $\tilde{R}$ may be defined by interpolation:

\[(\tilde{R}_1u_1)_{ij} = \frac{1}{4} (u_{1,2i-1,2j} + u_{1,2i-1,2j-1}) \quad (9.7.73)\]

\[(\tilde{R}_2u_2)_{ij} = \frac{1}{4} (u_{2,2i,2j-1} + u_{2,2i-1,2j-1}) \quad (9.7.74)\]

\[(\tilde{R}_3T)_{ij} = \frac{1}{4} (T_{2i,2j} + T_{2i-1,2j} + T_{2i,2j-1} + T_{2i-1,2j-1}) \quad (9.7.75)\]

and $\tilde{R}_4p$ is defined similar to $\tilde{R}_3T$.

Prolongation

Prolongation may be defined by bilinear interpolation. This gives

\[(P_1\tilde{u}_1)_{2i,2j} = \frac{1}{8} (3\tilde{u}_{1,i,j} + 3\tilde{u}_{1,i+1,j} + \tilde{u}_{1,i,j+1} + \tilde{u}_{1,i+1,j+1}) \quad (9.7.76)\]

\[(P_1\tilde{u}_1)_{2i-1,2j-1} = \frac{1}{8} (3\tilde{u}_{1,i,j} + 3\tilde{u}_{1,i+1,j} + \tilde{u}_{1,i,j+1} + \tilde{u}_{1,i+1,j+1}) \quad (9.7.76)\]

\[(P_1\tilde{u}_1)_{2i,j} = \frac{1}{8} (3\tilde{u}_{1,i,j} + \tilde{u}_{1,i,j+1}) \quad (9.7.76)\]

\[(P_1\tilde{u}_1)_{2i-1,j-1} = \frac{1}{8} (3\tilde{u}_{1,i,j} + \tilde{u}_{1,i,j+1}) \quad (9.7.76)\]

Determining $P_2\tilde{u}_2$ is left to the reader. For $P_3\tilde{T}$ one obtains

\[(P_3\tilde{T})_{2i,2j} = \frac{1}{16} (9\tilde{T}_{i,j} + 3\tilde{T}_{i+1,j} + 3\tilde{T}_{i,j+1} + \tilde{T}_{i+1,j+1}) \quad (9.7.77)\]

\[(P_3\tilde{T})_{2i-1,2j-1} = \frac{1}{16} (9\tilde{T}_{i,j} + 3\tilde{T}_{i+1,j} + 3\tilde{T}_{i,j+1} + \tilde{T}_{i+1,j+1}) \quad (9.7.77)\]

\[(P_3\tilde{T})_{2i,2j-1} = \frac{1}{16} (9\tilde{T}_{i,j} + 3\tilde{T}_{i+1,j} + 3\tilde{T}_{i,j+1} + \tilde{T}_{i+1,j+1}) \quad (9.7.77)\]

\[(P_3\tilde{T})_{2i-1,2j} = \frac{1}{16} (9\tilde{T}_{i,j} + 3\tilde{T}_{i+1,j} + 3\tilde{T}_{i,j+1} + \tilde{T}_{i+1,j+1}) \quad (9.7.77)\]

$P_4\tilde{p}$ is defined similar to $P_3\tilde{T}$. 
Applications of multigrid methods in computational fluid dynamics

These transfer operators satisfy

\[ m_{p_i} = 2, \quad m_{R_i} = 1 \quad (9.7.78) \]

Hence, the accuracy rule (9.7.68) is satisfied. For \( P_4 \) one could also use

\[ (P_4 \tilde{\rho})_{2i,2j} = (P_4 \tilde{\rho})_{2i-1,2j} = (P_4 \tilde{\rho})_{2i,2j-1} = (P_4 \tilde{\rho})_{2i-1,2j-1} = \tilde{\rho}_{ij}. \quad (9.7.79) \]

which gives \( m_{P_4} = 1 \), still satisfying (9.7.68). Niestegge and Witsch (1990) present Fourier two-grid analysis results for the Stokes equations, comparing various smoothing methods and transfer operators, confirming that the transfer operators defined above will work.

**Application to a free convection flow**

We will now describe the application of the multigrid method just described to the computation of a flow problem described by Roux (1990, 1990a). The domain is a rectangular cavity, see Figure 9.7.4. The height of the cavity is taken as the unit of length. The aspect ratio is \( r \). The temperature equals \( T_1 \) at \( x_1 = 1 \) and \( T_1 + T_0 \) at \( x_1 = 0 \). Taking advantage of the fact that the Boussinesq equations leave the velocity field invariant under addition of a constant to the temperature (cf. Exercise 9.7.3) we define the dimensionless temperature by

\[ T := (T - T_1)/T_0 \quad (9.7.80) \]

The horizontal walls are perfectly conducting, so that there \( T \) varies linearly. This gives the following Dirichlet boundary conditions for \( T \)

\[ T(0, x_2) = 1, \quad T(r, x_2) = 0, \quad T(x_1, 0) = T(x_1, 1) = (r - x_1)/r \quad (9.7.81) \]

The walls are at rest, so that \( u = 0 \) at the boundary. The other physical quan-

**Figure 9.7.4** Rectangular cavity for free convection flow.
The incompressible Navier–Stokes and Boussinesq equations

The quantities that determine the problem are (cf. (9.7.3) (9.7.4)) $v$, $g$ and $\mu$. The unit of length $L$ is the height of the cavity. This specifies the Grashof number:

$$Gr = \frac{\gamma g L^3 T_0}{\nu^2} \quad (9.7.82)$$

The flow is completely driven by the buoyancy force, represented by the last term of (9.7.3). A reasonable velocity unit $U$ is, therefore, such that this term has coefficient 1 in the dimensionless form (6). This implies $Re = Gr^{1/2}$, or

$$U = (\gamma g T_0)^{1/2} \quad (9.7.83)$$

The resulting dimensionless equations are given by (9.7.2) (9.7.6) and (9.7.7), with $Re = Gr^{1/2}$. Note that in Roux (1990, 1990a) the Grashof number is defined a little differently, and equals $Gr/r$. We take $r = 4$, and $Pr \ll 1$.

Physical characteristics of the flow

The resulting flow has the following interesting features. For $Gr < 10^5$ a steady flow results. This flow is centro-symmetric and consists of a main central cell and two adjacent small cells (called the $S_{12}$ state). For $Gr \geq 10^5$ the steady flow becomes unstable, and bifurcates to a laminar unsteady flow. At $Gr = 1.2 \times 10^5$ the flow is periodic and centro-symmetric, but after several tens of periods suddenly changes to a quasi-periodic flow which is no longer centro-symmetric. At $Gr = 1.2 \times 10^5$ and $Gr = 2 \times 10^5$ a steady flow also exists, which is centro-symmetric and has two cells; this is called the $S_2$ state. At $Gr = 1.6 \times 10^5$ also an oscillating solution exists, which after many periods suddenly switches to the $S_2$ state. These features, described by Roux (1990, 1990a) have been found by a number of investigators by numerical means. Their reproduction is a demanding test for numerical methods. For example, the hybrid scheme misses the transition from the $S_{12}$ state to the periodic state, because the numerical diffusion is too great, unless a very fine mesh is used, such that the hybrid scheme is switched to the central scheme.

Application of a multigrid method

The work to be described here has been carried out in cooperation with Zeng Shi (Tsinghua University, Beijing). Both time-dependent and time-independent calculations have been carried out. Denoting the nonlinear stationary discrete equations described before by

$$A(y) = b \quad (9.7.84)$$

the time-dependent discrete equations are chosen as follows

$$(y^{n+1} - y^{(n)})/\Delta t + \theta A(y^{n+1}) + (1 - \theta)A(y^n) = \theta b^{n+1} + (1 - \theta)b^n \quad (9.7.85)$$
where the superscript \( n \) denotes the time level. We take \( \theta = 1/2 \) (Crank–Nicolson scheme). Hence, for every time step one has to solve

\[
y^{n+1} + \theta \Delta t A (y^{n+1}) = y^n - (1 - \theta) \Delta t A (y^n) + \theta \Delta t b^{n+1} + (1 - \theta) \Delta t b^n \tag{9.7.86}
\]

Both (9.7.84) and (9.7.86) are solved with the non-linear multigrid algorithm with adaptive schedule, of which the structure diagram is given in Figure 8.7.2. First, nested iteration is applied, which gives us \( \tilde{y}^k \). Next multigrid iteration is applied to (9.7.84), taking \( \tilde{y}^k \) from the preceding iteration. In the time-dependent case the solution of (9.7.84) is taken as initial solution; \( \tilde{y}^k \) is obtained from the preceding iteration or the preceding time level, as the case may be. Of course, solving (9.7.86) with the same method as used for (9.7.84) may not be the most efficient; for special multigrid methods for parabolic initial value problems see Hackbusch (1984a, 1985) and Murata et al. (1991).

The multigrid method is further specified as follows. One pre- and one post-smoothing is carried out with the SCGS method with \( \alpha_\sigma = 0.6 \). On the coarsest grid 11 smoothings are carried out. The parameters governing the multigrid schedule are \( \text{tol} = 10^{-8}, \delta = 0.2 \). The residual norm is defined by

\[
\| r \| = \left\{ \frac{4}{s=1} \| r_s \|^2 \right\}^{1/2}, \quad \| r_s \|^2 = N_s^{-1} \sum_{i,j} r^2_{s,i,j}, \quad s \neq 3
\tag{9.7.87}
\]

\[
\| r_3 \|^2 = Pr^2 N_3^{-1} \sum_{i,j} r^2_{3,i,j}
\]

where \( N_s \) is the number of grid points associated with \( r_s \). We scale \( \| r_3 \|^2 \) with \( P^2 \) in order to balance the residuals for \( Pr \ll 1 \), thus avoiding to demand much more accuracy for \( T \) than for the other unknowns. After the multigrid method has converged defect correction may or may not be applied.

The stationary case

Solutions are computed for \( Gr = 10^{-5} \) and \( Pr = 0.15 \times 10^{-11} \), on grids with 64 \( \times \) 16, 128 \( \times \) 32 and 256 \( \times \) 64 cells. The coarsest grid has 8 \( \times \) 2 cells. Defining the work unit (WU) as the cost of one smoothing on the finest grid, the solution on the three grids with one defect correction is obtained in about 140 WU (counting only smoothing work), showing a mesh-size-independent rate of convergence. Note that with \( \text{tol} = 10^{-8} \) we converge well beyond engineering accuracy, and probably also well beyond discretization accuracy, which we did not try to estimate. The solution on the 256 \( \times \) 64 grid is shown in Figure 9.7.5(a). It is centro-symmetric, and in close agreement with the results of other authors as presented in Roux (1990, 1990a). This is the \( S_{12} \) state referred to earlier. The solution on the 128 \( \times \) 32 grid is quite similar to the one on the 256 \( \times \) 64 grid, but on the coarser grids the two smaller vortices are not resolved. Without defect correction the solution is quite similar to the one
The incompressible Navier–Stokes and Boussinesq equations

with defect correction on the $256 \times 64$ grid, presumably because the hybrid scheme is largely switched to the central scheme on this fine grid. On the $128 \times 32$ grid, however, the hybrid scheme does not resolve the two small vortices.

Although the savings in computing time due to multigrid are very great in this case, it is hard to estimate these savings, because SCGS is not a very good

\[ \text{Figure 9.7.5 Streamline patterns for free convection problem. For further information, see the text.} \]
single grid iteration method. In the first 100 work units SCGS as a single grid method drives down the residual norm by about a factor 10, but the next factor 10 requires about 1000 WU.

In computing Navier—Stokes solutions for the flow in a driven cavity, of which no further details will be given here, it was found that the ratios of the computing times using the adaptive multigrid schedule used here, the W-cycle and the V-cycle was roughly 1:1.3:2.7; these figures are approximate and problem-dependent. It is found that the adaptive schedule expends relatively more effort on the coarser grids than the W-cycle and, a fortiori, than the V-cycle.

The non-stationary case

Non-stationary calculations are carried out on the 128 × 32 grid. We take \( Gr = 1.2 \times 10^{5} \), \( Pr = 0.15 \times 10^{-11} \) and \( \Delta t = 2 \), which is about 1/8 of the period of the oscillations that should occur. Without defect correction no oscillations occur, which is thought to be due to the damping effect of the numerical viscosity inherent in the hybrid scheme. With defect correction periodic oscillations occur with flow patterns closely resembling those found by other authors (Roux 1990, 1990a). The cost of a time-step is about 23 WU. The flow pattern is centro-symmetric. The computations were not continued long enough to observe the transition to quasi-periodic oscillations which should occur.

Choosing \( \Delta t = 2 \) and \( Gr = 1.6 \times 10^{5} \) periodic oscillations with period about 20 are found, followed by a transition lasting from \( t = 200 \) until \( t = 280 \) to the \( S_2 \) state, which persists. The \( S_2 \) state is shown in Figure 9.7.5(b). Figures 9.7.5(c, d) give flow patterns, half a period apart, which occur during the periodic oscillations preceding transition. With \( \Delta t = 1 \) transition takes place from \( t = 240 \) until \( t = 280 \). These results and the observed flow patterns are in agreement with the results presented in Roux (1990, 1990a).

It could have been thought that the multigrid method might have trouble computing this kind of flow, because on the coarse grids with the hybrid scheme the correct solution branch cannot be found. We have, however, seen that this difficulty does not materialize, and that it is sufficient to drive the non-linear multigrid algorithm with the correct residual on the finest grid.

We may conclude that the non-linear multigrid method combined with defect correction is a dependable, robust and efficient method to solve complicated problems from computational fluid dynamics.

Literature

There is a rapidly growing literature on the application of multigrid methods to the numerical solution of the incompressible Navier—Stokes equations. A (no doubt incomplete) list of recent publications using the staggered formulation is: Arakawa et al. (1988), Becker et al. (1989), Bruneau and

The colocated formulation is employed by Barcus et al. (1988), Majumdar et al. (1988), Michelsen (1990), Orth and Schöning (1990) Dick (1988, 1988a, 1989a), and Dick and Linden (1990). Compared with single grid methods, large speed-up factors are found that increase when the grid is refined to 100 and beyond.

**Exercise 9.7.1.** Prove that for the Stokes equations the first two elements of the last column in (9.7.47) vanish in the interior.

**Exercise 9.7.2.** Show that $G_\alpha^*$ (defined in (9.7.33)) is the adjoint of $G_\alpha$ (defined in (9.7.32)).

**Exercise 9.7.3.** Show that the Boussinesq equations (9.7.2), (9.7.6) and (9.7.7) have the following property: if $u_\alpha, p, T$ is a solution, then $u_\alpha, p + Gr \cdot Re^{-2} \Delta T x_2, T + \Delta T$ is also a solution (provided the boundary conditions for $T$ are adjusted accordingly).

### 9.8. Final remarks

An introduction has been given to the application of multigrid methods in computational fluid dynamics. The subject has been only partially covered. No mention has been made of computation of flow in porous media (reservoir engineering), where the use of multigrid methods is also developing (see for example Behie and Forsyth 1982, Schmidt and Jacobs 1988, Schmidt 1990). We have also neglected the subject of grid generation, where multigrid methods are evolving rapidly, especially for the purpose of adaptive discretization. In the application areas discussed multigrid methods have been investigated thoroughly, generating enough confidence to stimulate widespread use, permitting large gains in computing time and bringing larger scale models within reach.