Chapter 4
An Implicit Finite-Difference Algorithm

4.1 Introduction

A numerical solution algorithm for the Navier-Stokes equations converts the original system of partial differential equations (PDEs) to a much larger system of algebraic equations, which is then solved. Many such algorithms discretize space and time independently, such that the PDEs are first reduced to ordinary differential equations (ODEs) through the discretization of the spatial terms in the governing equations. This semi-discrete ODE system is then converted to a system of ordinary difference equations (OΔEs) through a time-marching method. This assumes that the PDE system is time-dependent. If one is interested only in the steady solution of the Navier-Stokes equations, then the time-derivative terms can be dropped, and there is no intermediate ODE system. In this case, the spatial discretization directly reduces the original nonlinear PDE system to a system of nonlinear algebraic equations. Being nonlinear, this algebraic system cannot be solved directly and must be solved using an iterative method. It can often be useful to retain the time-dependent terms even if one is interested only in the steady solution, as a time-marching method that follows a quasi-physical path to the steady solution can be an effective iterative method.

Both the implicit algorithm presented in this chapter and the explicit algorithm presented in the next chapter retain the time-derivative terms in the Navier-Stokes equations even when solving for steady flows. Moreover, both algorithms involve independent discretization of space and time, and hence an intermediate semi-discrete ODE form. In principle, the spatial and temporal components of the algorithms could be presented independently. However, in these two algorithms the two are quite closely linked. In other words, the time-marching methods are particularly effective with the specific spatial discretization used. Nonetheless, the reader should be aware that it is of course possible and reasonable to develop an explicit finite-difference algorithm or an implicit finite-volume algorithm.

T. H. Pulliam and D. W. Zingg, *Fundamental Algorithms in Computational Fluid Dynamics*, Scientific Computation, DOI: 10.1007/978-3-319-05053-9_4,

The key characteristics of the algorithm presented in this chapter are as follows:

- node-based data storage; the numerical solution for the state variables is associated with the nodes of the grid
- second-order finite-difference spatial discretization; centered with added numerical dissipation; a simple shock-capturing device
- transformation to generalized curvilinear coordinates; applicable to structured grids
- implicit time marching based on approximate factorization of the resulting matrix operator

All of these terms will be explained in this chapter. Key contributions to this algorithm were made by Beam and Warming [1], Steger [2], Warming and Beam [3], Pulliam and Steger [4], Pulliam and Chaussee [5], and Pulliam [6].

The exercises at the end of the chapter provide an opportunity to write a computer program to apply this algorithm to several one-dimensional problems. Neither approximate factorization nor the coordinate transformation will enter into this program, but the exercise will enable the reader to develop a greater understanding of most other aspects of the algorithm.

4.1.1 Implicit Versus Explicit Time-Marching Methods

As discussed in Chap. 2, time-marching methods can be classified as implicit or explicit, and the two types have significantly different properties with respect to stability and cost. A simple characterization of implicit and explicit methods states that implicit methods have a much higher computing cost per time step, but their stability properties permit much larger time steps to be used. Depending on the nature of the problem, specifically its *stiffness*, either method can be more efficient. Implicit methods become relatively more efficient with increasing problem stiffness.

In computational fluid dynamics, stiffness has many sources, both physical and numerical. Physical stiffness comes from varying scales and speeds associated with different physical processes contained in the PDEs. For example, if the computation includes chemical reactions that proceed at rates much higher than those associated with the basic fluid dynamics, and time-accurate resolution of the chemical reactions is not required, then this will lead to a stiff system. Figure 2.2 shows one way in which numerical stiffness is introduced. There exist many modes in the system at high wavenumbers that are completely inaccurate. Such modes are inherently parasitic. This means that resolving them accurately in time will not improve the accuracy of the solution, because the spatial discretization is not accurate for these components of the solution. Thus these modes and their associated eigenvalues must lie within the stable region of the time-marching method, but need not lie within its region of accuracy (see Fig. 2.6). Furthermore, in many computations, very small grid spacings are needed in some regions of the flow, such as boundary layers, while much larger spacings are sufficient elsewhere. This too can cause stiffness, as the

time taken for information to pass through a small cell is much shorter than that taken to pass through a large cell, introducing widely different time scales from a numerical point of view. Moreover, if gradients are much higher in one direction than another, then it is efficient to use small grid spacings in the direction of large gradient and larger spacings in the smaller gradient direction, leading to grid cells with high aspect ratios. As the time taken for waves to traverse the cell in one direction is thus much different from the other direction, multiple time scales and hence stiffness can again be introduced.

One way to understand the choice between implicit and explicit methods is to consider the limiting factor in the choice of the time step. Accuracy considerations place one bound on the maximum allowable time step. In other words, the time step must be small enough that the time accuracy of the solution is sufficient. Stability considerations place another bound on the time step. If the accuracy bound is smaller than the stability bound, then the time step is said to be *accuracy limited*. If the stability bound is smaller, then it is said to be *stability limited*. In a simulation where the time step is accuracy limited, there is little point in using an implicit method, as the same time step must be used in either case, so the extra cost per time step of an implicit method is not worthwhile. Conversely, if the stability bound is much smaller than the accuracy bound, then the explicit method will require a much smaller time step than an unconditionally stable implicit method, and hence the latter can be more efficient.

In the context of the numerical solution of ODEs, it is straightforward to categorize a method as explicit or implicit. In the context of PDEs, it is more accurate to classify methods according to a spectrum ranging from fully explicit to fully implicit. At the fully explicit end of the spectrum lies a method such as the explicit Euler method, without any additional convergence acceleration techniques, such as multigrid or implicit residual smoothing (which the reader will learn about in the next chapter). A multi-stage method, such as an explicit Runge-Kutta method, is still officially explicit, but generally has a larger stability bound at the expense of an increased cost per time step and can therefore be considered to have moved slightly toward the implicit end of the spectrum. Similarly, convergence acceleration techniques such as implicit residual smoothing and multigrid move the resulting "explicit" algorithm further in the implicit direction. This is typically associated with increased transfer of information across the mesh during a time step, which is a characteristic of implicit methods, an increased stability bound, and an increased cost per time step. At the fully implicit end of the spectrum lies the implicit Euler method with a direct solution of the linear problem at each time step. As this is usually infeasible and inefficient, for reasons to be discussed in this chapter, the linear problem is usually solved inexactly using an iterative method, which moves the algorithm slightly in the explicit direction. Alternatively, the linear problem can be approximated in a manner that makes it easier to solve, as in the approximate factorization algorithm that is the subject of this chapter. This reduces the cost per time step but can also reduce the optimal time step for convergence; in other words, it moves the algorithm somewhat further away from the fully implicit end of the spectrum.

Both the extreme explicit and the extreme implicit ends of the spectrum lead to inefficient algorithms for large problems. Therefore, all practical algorithms

in use today for large-scale problems, including the algorithms described in this and the following chapter, lie somewhere between these two extremes, with the choice depending on the stiffness of the particular problem under consideration. It is interesting to note that, although this chapter's algorithm is nominally classified as implicit, while next chapter's algorithm is nominally classified as explicit, their cost per time step is quite comparable.

4.2 Generalized Curvilinear Coordinate Transformation

Finite-difference formulas are most naturally implemented on rectilinear meshes, as described in Chap. 2. On such meshes, the mesh lines are orthogonal, and it is straightforward to align the mesh such that each mesh line is associated with a specific coordinate direction. The derivative in a given coordinate direction can then be easily approximated based on finite differences along the corresponding mesh line. On the other hand, implementation of boundary conditions is simplified if the mesh is *body-fitted*, in other words the mesh conforms to the boundary of the geometry under consideration. If the boundary is curved, as is the case for most geometries of interest, this precludes the use of a mesh that is both rectilinear and body-fitted. In the present algorithm, this issue is addressed by transforming the physical space in which the mesh has curved, potentially non-orthogonal mesh lines into a computational space in which the mesh is rectilinear through a generalized curvilinear coordinate transformation. Such a transformation enables the straightforward application of finite-difference formulas on a body-fitted mesh. Our exposition will be in two dimensions, but extension to three dimensions should not present the reader with any conceptual difficulties.

An example of a mesh about an airfoil is shown in Fig. 4.1, and the corresponding curvilinear coordinate transformation is shown schematically in Fig. 4.2. In this case, the body is an airfoil, and the flow domain is bounded by an outer boundary. In the physical space defined by the Cartesian coordinates x, y, one set of mesh lines forms a "C" and hence such a mesh is known as a "C-mesh." The innermost "C" conforms to the airfoil surface and a *wake cut* along which two mesh lines correspond to a single line in physical space. The outermost "C" corresponds to the curved portion of the outer boundary. This set of lines is defined to be the one along which the curvilinear coordinate ξ varies, and the curvilinear coordinate η is constant. The second set of mesh lines is roughly orthogonal to the first and emanates from the body or the wake cut toward the outer boundary. Along these lines, η varies, and ξ is constant. The coordinate transformation is chosen such that the mesh is mapped to a computational space where the mesh lines are orthogonal, and the spacings $\Delta\xi$ and $\Delta\eta$ are unity in both directions. Therefore, standard finite-difference formulas can be easily applied. The computational space is a rectangle, where the bottom side includes the grid line lying on the airfoil and the wake cut, the top is the curved portion of the outer boundary, the left side is the portion of the back boundary below the wake cut, and the right side is the portion of the back boundary above the wake cut. Although

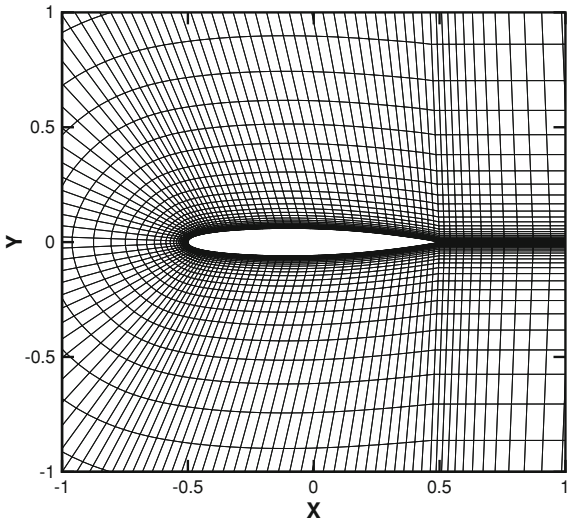

Fig. 4.1 A sample airfoil grid with a "C" topology showing only the region near the airfoil

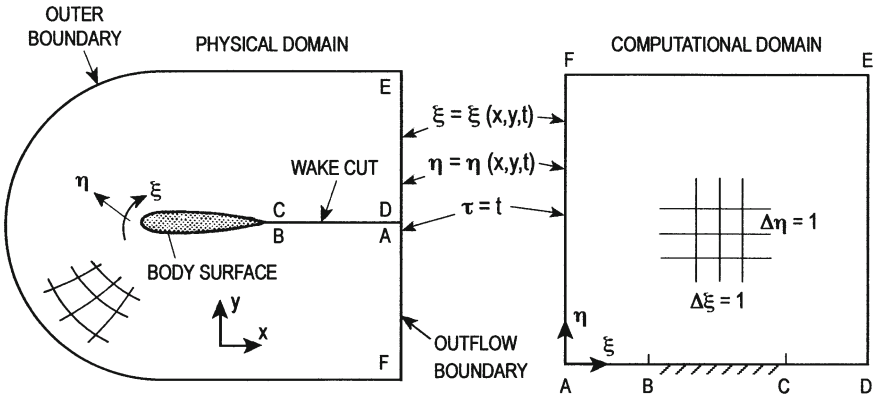

Fig. 4.2 An example of a generalized curvilinear coordinate transformation for a C-mesh

meshes can be defined by an analytical transformation for simple geometries, they are typically defined solely by the Cartesian coordinates of their nodes, and the underlying transformation to computational space is not known explicitly.

It is important to note that the mesh topology shown in Figs. 4.1 and 4.2 is just one possible topology. Another possibility, an "O" mesh, is shown in Fig. 4.3. The key property of such meshes, known as *structured meshes* is that the nodes are aligned along coordinate directions. This contrasts with *unstructured meshes*, which have no such constraint. An interior node in a two-dimensional structured mesh must have four neighbors (six in three dimensions), while a node in an unstructured mesh can have an arbitrary number of neighbors. This characteristic of a structured mesh sim-

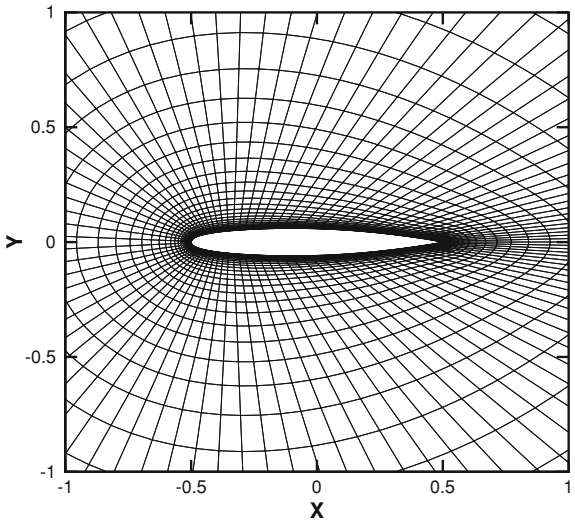

Fig. 4.3 A sample airfoil grid with an "O" topology showing only the region near the airfoil

plifies its storage. In two dimensions, a structured mesh is defined by a set of x and y coordinates that are assigned indices j and k, where j corresponds to the index in the ξ direction, and k corresponds to the η direction. The four immediate neighbors of node (j, k) are the nodes with indices $(j+1, k)$, $(j-1, k)$, $(j, k+1)$, $(j, k-1)$; the connectivity is implied by the indices. For more complex geometries, it can be impossible to define a mesh such that a single, simply connected, rectangular computational space exists. For such cases, block-structured meshes can be defined such that multiple rectangular computational domains are produced by the transformation. These domains can be interfaced in a number of different ways, including overlapping and abutting blocks.

In order to make use of finite-difference formulas defined in computational space, the governing equations must be transformed such that derivatives with respect to the Cartesian coordinates x and y are replaced by derivatives with respect to computational coordinates ξ and η. The coordinate transformation introduced here follows the development of Viviand [7] and Vinokur [8]. The Navier-Stokes equations can be transformed from Cartesian coordinates to generalized curvilinear coordinates where

$$\begin{aligned} \tau &= t \\ \xi &= \xi(x, y, t) \\ \eta &= \eta(x, y, t). \end{aligned} \tag{4.1}$$

If the grid does not deform over time, then $\xi = \xi(x, y)$ and $\eta = \eta(x, y)$. Typically there will be a one to one correspondence between a physical point in space and a computational point, except for regions where there are singularities or cuts due to

the topology, such as the wake cut in the C-mesh example above. In those cases it may be necessary to map one physical point to more than one computational point.

The present coordinate transformation differs from some in that only the independent variables are transformed. The dependent variables remain defined in the Cartesian space, e.g. in terms of the Cartesian velocity components u and v. Chain-rule expansions are used to represent the derivatives in Cartesian space, ∂_t, ∂_x, and ∂_y of (3.1), in terms of the curvilinear derivatives, as follows:

$$\begin{aligned}\frac{\partial}{\partial x} &= \frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta} \\ \frac{\partial}{\partial y} &= \frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta} \\ \frac{\partial}{\partial t} &= \frac{\partial}{\partial \tau} + \frac{\partial \xi}{\partial t}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial t}\frac{\partial}{\partial \eta}.\end{aligned} \tag{4.2}$$

Introducing the notation

$$\partial_x \equiv \frac{\partial}{\partial x} \quad \text{and} \quad \xi_x \equiv \frac{\partial \xi}{\partial x}, \tag{4.3}$$

these can be written in matrix form as

$$\begin{bmatrix} \partial_t \\ \partial_x \\ \partial_y \end{bmatrix} = \begin{bmatrix} 1 & \xi_t & \eta_t \\ 0 & \xi_x & \eta_x \\ 0 & \xi_y & \eta_y \end{bmatrix} \begin{bmatrix} \partial_\tau \\ \partial_\xi \\ \partial_\eta \end{bmatrix}. \tag{4.4}$$

Applying these chain-rule expansions to the Navier-Stokes equations (3.1), we obtain

$$\begin{aligned}&\partial_\tau Q + \xi_t \partial_\xi Q + \eta_t \partial_\eta Q + \xi_x \partial_\xi E + \eta_x \partial_\eta E + \xi_y \partial_\xi F + \eta_y \partial_\eta F \\ &\quad = Re^{-1}\left(\xi_x \partial_\xi E_\mathrm{v} + \eta_x \partial_\eta E_\mathrm{v} + \xi_y \partial_\xi F_\mathrm{v} + \eta_y \partial_\eta F_\mathrm{v}\right).\end{aligned} \tag{4.5}$$

4.2.1 Metric Relations

In (4.5), derivatives with respect to t, x, and y have been replaced by derivatives with respect to τ, ξ, and η. Since the computational space is rectilinear and equally spaced, the latter can be easily approximated using finite-difference expressions—these will be presented in a subsequent section. The coefficients introduced ($\xi_t, \xi_x, \xi_y, \eta_t, \eta_x, \eta_y$) are known as grid metrics. Since in most cases the transformation from physical space to computational space is not known analytically, the metrics must be determined numerically. That is, we usually are provided with just the x, y coordinates of the grid points and must numerically generate the

metrics $(\xi_t, \xi_x, \xi_y, \eta_t, \eta_x, \eta_y)$ using finite differences. This introduces a difficulty in that these are derivatives with respect to the original Cartesian coordinates.

In order to address this, consider the inverse of the transformation given in (4.1):

$$\begin{aligned} t &= \tau \\ x &= x(\xi, \eta, \tau) \\ y &= y(\xi, \eta, \tau). \end{aligned} \tag{4.6}$$

Reversing the role of the independent variables in the chain rule formulas (4.3), we have,

$$\partial_\tau = \partial_t + x_\tau \partial_x + y_\tau \partial_y, \quad \partial_\xi = x_\xi \partial_x + y_\xi \partial_y, \quad \partial_\eta = x_\eta \partial_x + y_\eta \partial_y, \tag{4.7}$$

which can be written in matrix form as

$$\begin{bmatrix} \partial_\tau \\ \partial_\xi \\ \partial_\eta \end{bmatrix} = \begin{bmatrix} 1 & x_\tau & y_\tau \\ 0 & x_\xi & y_\xi \\ 0 & x_\eta & y_\eta \end{bmatrix} \begin{bmatrix} \partial_t \\ \partial_x \\ \partial_y \end{bmatrix}. \tag{4.8}$$

Comparing (4.4) and (4.8), it is immediately clear that

$$\begin{bmatrix} 1 & \xi_t & \eta_t \\ 0 & \xi_x & \eta_x \\ 0 & \xi_y & \eta_y \end{bmatrix} = \begin{bmatrix} 1 & x_\tau & y_\tau \\ 0 & x_\xi & y_\xi \\ 0 & x_\eta & y_\eta \end{bmatrix}^{-1} \tag{4.9}$$

$$= J \begin{bmatrix} (x_\xi y_\eta - y_\xi x_\eta) & (-x_\tau y_\eta + y_\tau x_\eta) & (x_\tau y_\xi - y_\tau x_\xi) \\ 0 & y_\eta & -y_\xi \\ 0 & -x_\eta & x_\xi \end{bmatrix}, \tag{4.10}$$

where $J = (x_\xi y_\eta - x_\eta y_\xi)^{-1}$ is defined as the metric Jacobian. This yields the following metric relations:

$$\begin{aligned} \xi_t &= J(-x_\tau y_\eta + y_\tau x_\eta), \quad \xi_x = J y_\eta, \quad \xi_y = -J x_\eta \\ \eta_t &= J(x_\tau y_\xi - y_\tau x_\xi), \quad \eta_x = -J y_\xi, \quad \eta_y = J x_\xi. \end{aligned} \tag{4.11}$$

Using these relations, the metrics $(\xi_t, \xi_x, \xi_y, \eta_t, \eta_x, \eta_y)$ can be determined from $(x_\tau, x_\xi, x_\eta, y_\tau, y_\xi, y_\eta)$, where the latter are easily found using finite differences, since they are derivatives in computational space. Finite-difference formulas for these terms will be presented later in this chapter.

4.2.2 Invariants of the Transformation

At this point we notice that the transformed equations (4.5) are in a weak conservation law form. That is, even though none of the flow variables (or functions of the flow variables) occur as coefficients in the differential equations, the metrics, which are spatially varying, lie outside of the derivative operators. There is some argument in the literature which advocates the use of the so called "chain rule form," since it should still have good shock capturing properties and in some ways is a simpler form. Here, though, we shall restrict ourselves to the strong conservation law form which will be derived below.

To simplify our derivation, we will consider the inviscid terms only. This reduces (4.5) to

$$\partial_\tau Q + \xi_t \partial_\xi Q + \eta_t \partial_\eta Q + \xi_x \partial_\xi E + \eta_x \partial_\eta E + \xi_y \partial_\xi F + \eta_y \partial_\eta F = 0. \quad (4.12)$$

To produce the strong conservation law form we first multiply (4.12) by J^{-1} and apply the product rule to all terms. For example, the fourth term on the left-hand side can be expanded as

$$\left(\frac{\xi_x}{J}\right)\partial_\xi E = \partial_\xi\left(\frac{\xi_x}{J}E\right) - E\partial_\xi\left(\frac{\xi_x}{J}\right). \quad (4.13)$$

Each term can thus be rewritten as the difference between a term in the form we are looking for, with no coefficient outside the derivative operator, and a second term that is the product of a function of Q and a derivative of a quantity that is strictly a function of the grid. Collecting all the terms into two groups, with Term_1 representing the first group of terms and Term_2 the second, we obtain

$$\text{Term}_1 + \text{Term}_2 = 0,$$

where

$$\begin{aligned}
\text{Term}_1 = {} & \partial_\tau(Q/J) + \partial_\xi[(\xi_t Q + \xi_x E + \xi_y F)/J] + \partial_\eta[(\eta_t Q + \eta_x E + \eta_y F)/J] \\
\text{Term}_2 = {} & -Q[\partial_\tau(J^{-1}) + \partial_\xi(\xi_t/J) + \partial_\eta(\eta_t/J)] \\
& -E[\partial_\xi(\xi_x/J) + \partial_\eta(\eta_x/J)] - F[\partial_\xi(\xi_y/J) + \partial_\eta(\eta_y/J)].
\end{aligned} \quad (4.14)$$

The expressions from Term_2,

$$\begin{aligned}
& \partial_\tau(J^{-1}) + \partial_\xi(\xi_t/J) + \partial_\eta(\eta_t/J) \\
& \partial_\xi(\xi_x/J) + \partial_\eta(\eta_x/J) \\
& \partial_\xi(\xi_y/J) + \partial_\eta(\eta_y/J),
\end{aligned} \quad (4.15)$$

are defined as invariants of the transformation. Substituting the metric relations (4.11) into the invariant expressions gives

$$\partial_\tau(x_\xi y_\eta - y_\xi x_\eta) + \partial_\xi(-x_\tau y_\eta + y_\tau x_\eta) + \partial_\eta(x_\tau y_\xi - y_\tau x_\xi)$$
$$\partial_\xi(y_\eta) + \partial_\eta(-y_\xi) \tag{4.16}$$
$$\partial_\xi(-x_\eta) + \partial_\eta(x_\xi). \tag{4.17}$$

Analytically, differentiation is commutative, and the above terms sum to zero. This eliminates Term_2 of (4.15), and the resulting equations are in strong conservation law form.

There is an important issue associated with these invariants. It is not true in general that finite-difference approximations are commutative. Consequently, when numerical differencing is applied to these equations (as developed in the Sect. 4.4), the finite-difference formulas used to evaluate the spatial derivatives of the fluxes and the finite-difference formulas used to calculate the metrics do not necessarily satisfy the commutative law. Second-order central differences commute, but mixed second-order and fourth-order formulas do not. This is further discussed in Sect. 4.4.1.

4.2.3 Navier-Stokes Equations in Generalized Curvilinear Coordinates

The Navier-Stokes equations written in strong conservation law form are

$$\partial_\tau \widehat{Q} + \partial_\xi \widehat{E} + \partial_\eta \widehat{F} = Re^{-1}[\partial_\xi \widehat{E}_\text{v} + \partial_\eta \widehat{F}_\text{v}], \tag{4.18}$$

with

$$\widehat{Q} = J^{-1}\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ e\end{bmatrix}, \widehat{E} = J^{-1}\begin{bmatrix}\rho U\\ \rho u U + \xi_x p\\ \rho v U + \xi_y p\\ U(e+p) - \xi_t p\end{bmatrix}, \widehat{F} = J^{-1}\begin{bmatrix}\rho V\\ \rho u V + \eta_x p\\ \rho v V + \eta_y p\\ V(e+p) - \eta_t p\end{bmatrix},$$

where

$$U = \xi_t + \xi_x u + \xi_y v, \quad V = \eta_t + \eta_x u + \eta_y v \tag{4.19}$$

are known as the contravariant velocity components—see Sect. 4.2.4 for more details. The viscous flux terms are $\widehat{E}_\text{v} = J^{-1}(\xi_x E_\text{v} + \xi_y F_\text{v})$ and $\widehat{F}_\text{v} = J^{-1}(\eta_x E_\text{v} + \eta_y F_\text{v})$.The viscous stress and heat conduction terms must also be transformed using the chain rule such that they are written in terms of ξ and η derivatives, giving

$$\begin{aligned}
\tau_{xx} &= \mu(4(\xi_x u_\xi + \eta_x u_\eta) - 2(\xi_y v_\xi + \eta_y v_\eta))/3 \\
\tau_{xy} &= \mu(\xi_y u_\xi + \eta_y u_\eta + \xi_x v_\xi + \eta_x v_\eta) \\
\tau_{yy} &= \mu(-2(\xi_x u_\xi + \eta_x u_\eta) + 4(\xi_y v_\xi + \eta_y v_\eta))/3 \\
f_4 &= u\tau_{xx} + v\tau_{xy} + \mu Pr^{-1}(\gamma - 1)^{-1}(\xi_x \partial_\xi a^2 + \eta_x \partial_\eta a^2) \\
g_4 &= u\tau_{xy} + v\tau_{yy} + \mu Pr^{-1}(\gamma - 1)^{-1}(\xi_y \partial_\xi a^2 + \eta_y \partial_\eta a^2).
\end{aligned} \tag{4.20}$$

The above discussion of metric invariants suggests a useful test for a finite-difference formulation. A minimum requirement of any finite-difference formulation is that a steady uniform flow be a valid solution of the discrete equations. If the chain-rule form (4.5) is evaluated for a steady uniform flow defined by

$$\begin{aligned}
\rho &= 1, \\
u &= M_\infty, \\
v &= 0, \\
e &= \frac{1}{\gamma(\gamma - 1)} + \frac{1}{2} M_\infty^2,
\end{aligned} \tag{4.21}$$

it is clearly satisfied, since all terms must equal zero given that the solution has no spatial or temporal variation. We would also like this steady uniform flow to satisfy (4.18) after the various derivatives have been replaced by finite-difference approximations. If the discrete form of (4.18) is not satisfied by a steady uniform flow, this can reveal a multitude of possible errors, including possibly a choice of difference operators for which the metric invariants are not zero.

4.2.4 Covariant and Contravariant Components in Curvilinear Coordinates

In Sect. 4.2.3 we introduced the contravariant velocity components associated with the curvilinear coordinate system. Since we will continue to work with Cartesian velocity components, a detailed knowledge of covariant and contravariant components is not necessary to understand the rest of the algorithm description. However, we will later need, for example, expressions for velocity components tangential and normal to a boundary in terms of the Cartesian components, so it is helpful to have a sufficient understanding to be able to derive such expressions.

We will assume a steady mesh in two dimensions, so we have $x(\xi, \eta)$, $y(\xi, \eta)$ and the inverse transformation $\xi(x, y)$, $\eta(x, y)$. First, define the vector

$$r = x\hat{i} + y\hat{j}. \tag{4.22}$$

In curvilinear coordinates, two sets of basis vectors can be defined. The covariant basis vectors are tangent to the ξ and η axes and are not required to be orthogonal.

They are given by

$$b_1 = \frac{\partial r}{\partial \xi}, \quad b_2 = \frac{\partial r}{\partial \eta}. \tag{4.23}$$

It can be more convenient to scale these such that they are unit vectors, giving

$$\hat{e}_1 = \frac{\frac{\partial r}{\partial \xi}}{\left|\frac{\partial r}{\partial \xi}\right|}, \quad \hat{e}_2 = \frac{\frac{\partial r}{\partial \eta}}{\left|\frac{\partial r}{\partial \eta}\right|}. \tag{4.24}$$

Note that these vectors are defined locally. The contravariant basis vectors are normal to the η and ξ axes and are defined by

$$B_1 = \nabla \xi, \quad B_2 = \nabla \eta, \tag{4.25}$$

where ∇ is the gradient operator. The contravariant basis vectors can also be scaled such that their length is unity:

$$\hat{E}_1 = \frac{\nabla \xi}{|\nabla \xi|}, \quad \hat{E}_2 = \frac{\nabla \eta}{|\nabla \eta|}. \tag{4.26}$$

With these bases, an arbitrary vector A can be defined in the following ways:

$$\begin{aligned} A &= A_1 \hat{e}_1 + A_2 \hat{e}_2 = a_1 \hat{E}_1 + a_2 \hat{E}_2 \\ &= C_1 b_1 + C_2 b_2 = c_1 B_1 + c_2 B_2. \end{aligned} \tag{4.27}$$

Here C_1 and C_2 are the contravariant components of A, i.e. $C_1 = B_1 \cdot A$ and $C_2 = B_2 \cdot A$, and c_1 and c_2 are the covariant components of A, i.e. $c_1 = b_1 \cdot A$ and $c_2 = b_2 \cdot A$. Note that $B_i \cdot b_j = \delta_{ij}$, where δ_{ij} is the Kronecker delta.

For example, let A represent the velocity vector $u\hat{i} + v\hat{j}$. From (4.25) we have

$$B_1 = \xi_x \hat{i} + \xi_y \hat{j}, \quad B_2 = \eta_x \hat{i} + \eta_y \hat{j}. \tag{4.28}$$

Therefore, we obtain for the contravariant components of velocity

$$C_1 = B_1 \cdot A = \xi_x u + \xi_y v, \quad C_2 = B_2 \cdot A = \eta_x u + \eta_y v, \tag{4.29}$$

consistent with the definitions of U and V in (4.19) when the coordinate transformation is time invariant.

In the application of boundary conditions, one often needs expressions for the velocity components normal and tangential to the boundary in terms of the Cartesian velocity components. In this case, we must work with unit basis vectors to preserve the magnitude of the velocity. We assume that the boundary is a grid line of constant η, such as the airfoil surface in Figs. 4.1 and 4.2, but the result is easily generalized

to other boundaries. Recall that $\hat{e}_1$ is tangent to the ξ axis, and $\hat{E}_2$ is normal to the η axis. Therefore, we can write

$$u\hat{i} + v\hat{j} = V_t\hat{e}_1 + V_n\hat{E}_2, \tag{4.30}$$

where V_t and V_n are the tangential and normal velocity components, respectively. The two unit vectors are given by

$$\begin{aligned}\hat{e}_1 &= \frac{x_\xi\hat{i} + y_\xi\hat{j}}{\sqrt{x_\xi^2 + y_\xi^2}} = \frac{\eta_y\hat{i} - \eta_x\hat{j}}{\sqrt{\eta_x^2 + \eta_y^2}} \\ \hat{E}_2 &= \frac{\eta_x\hat{i} + \eta_y\hat{j}}{\sqrt{\eta_x^2 + \eta_y^2}},\end{aligned} \tag{4.31}$$

where the metric relations are used to obtain the second expression for $\hat{e}_1$. Noting that

$$\hat{e}_1 \cdot \hat{E}_2 = 0, \quad \hat{e}_1 \cdot \hat{e}_1 = \hat{E}_2 \cdot \hat{E}_2 = 1, \tag{4.32}$$

we find the following expressions for the tangential and normal velocity components:

$$\begin{aligned}V_t &= \hat{e}_1 \cdot (u\hat{i} + v\hat{j}) = \frac{\eta_y u - \eta_x v}{\sqrt{\eta_x^2 + \eta_y^2}} \\ V_n &= \hat{E}_2 \cdot (u\hat{i} + v\hat{j}) = \frac{\eta_x u + \eta_y v}{\sqrt{\eta_x^2 + \eta_y^2}}.\end{aligned} \tag{4.33}$$

These are the velocity components tangential and normal to a grid line of constant η at a specific point in space.

As a second example, consider the derivative of pressure in a direction normal to a surface which again corresponds to a grid line of constant η. The gradient of pressure can be expressed in terms of the basis vectors $\hat{e}_1$ and $\hat{E}_2$ as follows:

$$\nabla p = \frac{\partial p}{\partial x}\hat{i} + \frac{\partial p}{\partial y}\hat{j} = \frac{\partial p}{\partial t}\hat{e}_1 + \frac{\partial p}{\partial n}\hat{E}_2, \tag{4.34}$$

where here t refers to the tangential coordinate. The normal derivative can be isolated by taking the dot product with $\hat{E}_2$ (which is identical to $\hat{n}$):

$$\frac{\partial p}{\partial n} = \hat{E}_2 \cdot \nabla p = \frac{\eta_x \frac{\partial p}{\partial x} + \eta_y \frac{\partial p}{\partial y}}{\sqrt{\eta_x^2 + \eta_y^2}}. \tag{4.35}$$

The chain rule gives

$$\frac{\partial p}{\partial x} = \eta_x \frac{\partial p}{\partial \eta} + \xi_x \frac{\partial p}{\partial \xi}, \qquad \frac{\partial p}{\partial y} = \eta_y \frac{\partial p}{\partial \eta} + \xi_y \frac{\partial p}{\partial \xi}, \tag{4.36}$$

from which we obtain the final expression for the normal derivative:

$$\frac{\partial p}{\partial n} = \frac{(\eta_x \xi_x + \eta_y \xi_y)\frac{\partial p}{\partial \xi} + (\eta_x^2 + \eta_y^2)\frac{\partial p}{\partial \eta}}{\sqrt{\eta_x^2 + \eta_y^2}}. \tag{4.37}$$

4.3 Thin-Layer Approximation

We introduce the thin-layer approximation [9] here only to simplify the treatment of the viscous terms in the exposition of the algorithm. It is not of fundamental importance and is applicable only if the following criteria are satisfied:

- The Reynolds number is high; the geometry is streamlined and at a modest angle of incidence with respect to the flow direction. Consequently, boundary layers remain attached or mildly separated, and both boundary layers and wakes are thin relative to the characteristic dimension of the geometry.
- The mesh is body fitted, and mesh lines are at least close to orthogonal to the surface, as depicted in Fig. 4.4. Moreover, lines of constant η are reasonably well aligned with wakes. As a result of this last constraint, a C-mesh is a better choice than an O-mesh when the thin-layer approximation is used.

Under these conditions, boundary-layer theory shows that streamwise gradients of viscous and turbulent stresses are small compared to normal gradients in boundary layers and wakes, and viscous and turbulent stresses are negligible outside of boundary layers and wakes. Therefore, mesh resolution requirements typically dictate a smaller mesh spacing in the direction normal to the surface in boundary layers, leading to meshes with cells having high aspect ratios near the surface, as in Fig. 4.4. Moreover, streamwise gradients of viscous and turbulent stresses can often be neglected with little impact on solution accuracy, leading to the thin-layer Navier-Stokes equations. It is important to recognize that although the rationale for the thin-layer Navier-Stokes equations is closely related to that for the boundary-layer equations, unlike the latter, the thin-layer Navier-Stokes equations retain all inviscid terms in full. Hence they are applicable both within boundary layers and wakes and outside these regions, where the flow is effectively inviscid.

We will assume that mesh lines along which η varies are nearly normal to the surface, as shown in Fig. 4.4. Applying the thin-layer approximation to (4.18) then involves neglecting the term $\partial_\xi \widehat{E}_v$ as well as all derivatives with respect to ξ in $\widehat{F}_v$, leading to

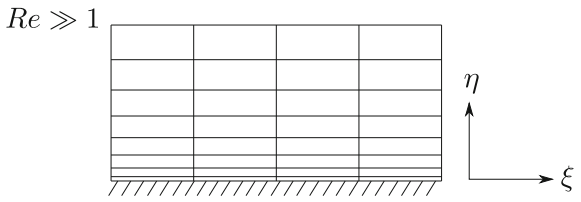

Fig. 4.4 Mesh near body surface

$$\partial_\tau \widehat{Q} + \partial_\xi \widehat{E} + \partial_\eta \widehat{F} = Re^{-1} \partial_\eta \widehat{S}, \tag{4.38}$$

where

$$\widehat{S} = J^{-1} \begin{bmatrix} 0 \\ \eta_x m_1 + \eta_y m_2 \\ \eta_x m_2 + \eta_y m_3 \\ \eta_x (u m_1 + v m_2 + m_4) + \eta_y (u m_2 + v m_3 + m_5) \end{bmatrix}, \tag{4.39}$$

with

$$\begin{aligned} m_1 &= \mu(4\eta_x u_\eta - 2\eta_y v_\eta)/3 \\ m_2 &= \mu(\eta_y u_\eta + \eta_x v_\eta) \\ m_3 &= \mu(-2\eta_x u_\eta + 4\eta_y v_\eta)/3 \\ m_4 &= \mu Pr^{-1}(\gamma - 1)^{-1} \eta_x \partial_\eta (a^2) \\ m_5 &= \mu Pr^{-1}(\gamma - 1)^{-1} \eta_y \partial_\eta (a^2). \end{aligned} \tag{4.40}$$

Although the thin-layer approximation was quite popular in the early days of CFD, it is important for the reader to understand that the algorithms presented here do not depend on this approximation and are applicable to the full Navier-Stokes equations. We proceed with the thin-layer approximation only because it simplifies our presentation of the algorithms while retaining their key features.

4.4 Spatial Differencing

We will now present an algorithm for the numerical solution of the transformed Navier Stokes equations (4.18), which in turn will provide a solution to the original equations in Cartesian coordinates (3.1). The algorithm will follow the semi-discrete approach described in Chap. 2 in which the spatial derivatives are approximated first to produce a system of ODEs.

Whether we are interested in a steady solution or a time-accurate solution to an unsteady problem, the first step is to take the continuous differential operators ∂_ξ and ∂_η and approximate them with finite-difference operators on a discrete mesh. This is facilitated by the use of the generalized curvilinear coordinate transformation

described in Sect. 4.2. A structured mesh is defined by a set of coordinate pairs $x(j,k)$, $y(j,k)$, where j and k are integer indices. If one defines $\xi \equiv j$ and $\eta \equiv k$, then the grid spacing in the computational space is unity in both directions, that is

$$\Delta\xi = 1, \quad \Delta\eta = 1. \tag{4.41}$$

Since the mesh is rectilinear and uniform in computational space, one can apply finite-difference formulas in a straightforward manner. We will use subscripts to indicate the coordinates of a flow variable in computational space, i.e.

$$Q_{j,k} := Q(j\Delta\xi, k\Delta\eta). \tag{4.42}$$

We can use second-order centered difference operators for the inviscid flux derivatives $\partial_\xi \widehat{E}$ and $\partial_\eta \widehat{F}$ as follows:

$$\delta_\xi \widehat{E}_{j,k} = \frac{\widehat{E}_{j+1,k} - \widehat{E}_{j-1,k}}{2\Delta\xi}, \quad \delta_\eta \widehat{F}_{j,k} = \frac{\widehat{F}_{j,k+1} - \widehat{F}_{j,k-1}}{2\Delta\eta}. \tag{4.43}$$

Similarly, second-order centered differences can be used for the metric terms, such as

$$\left(x_\xi\right)_{j,k} = \frac{x_{j+1,k} - x_{j-1,k}}{2\Delta\xi}. \tag{4.44}$$

Since $\Delta\xi = \Delta\eta = 1$ as a result of the transformation to computational space, we omit these terms for the remainder of this presentation.

For the viscous derivatives, the terms take the form

$$\partial_\eta \left(\alpha_{j,k} \partial_\eta \beta_{j,k}\right), \tag{4.45}$$

where $\alpha_{j,k}$ represents a spatially varying coefficient, such as a grid metric or the fluid viscosity, and $\beta_{j,k}$ is a velocity component or the square of the sound speed. Such a term can be approximated by differencing $\partial_\eta \beta_{j,k}$ using a second-order centered difference at each node, multiplying by the spatially varying coefficient, and applying the centered first-derivative approximation again. However, this leads to a five-point stencil involving values from $k-2$ to $k+2$ in the evaluation of (4.45). In the interest of retaining a compact three-point form, the term $\partial_\eta \beta_{j,k}$ can instead be evaluated at intermediate locations $k - \frac{1}{2}$ and $k + \frac{1}{2}$ using the following centered difference formulas:

$$\left(\frac{\partial \beta}{\partial \eta}\right)_{k+1/2} = \beta_{j,k+1} - \beta_{j,k}$$
$$\left(\frac{\partial \beta}{\partial \eta}\right)_{k-1/2} = \beta_{j,k} - \beta_{j,k-1}. \tag{4.46}$$

To second-order accuracy, the values of the spatially varying coefficient at the intermediate nodes can be found by averaging as follows:

$$\alpha_{j,k+1/2} = \frac{1}{2}\left(\alpha_{j,k} + \alpha_{j,k+1}\right)$$
$$\alpha_{j,k-1/2} = \frac{1}{2}\left(\alpha_{j,k-1} + \alpha_{j,k}\right). \tag{4.47}$$

A compact three-point finite-difference approximation to (4.45) can be obtained by applying a centered difference approximation at j, k using the intermediate points $j, k + \frac{1}{2}$ and $j, k - \frac{1}{2}$, as follows:

$$\frac{\left(\alpha_{j,k+1} + \alpha_{j,k}\right)}{2}\left(\beta_{j,k+1} - \beta_{j,k}\right) - \frac{\left(\alpha_{j,k} + \alpha_{j,k-1}\right)}{2}\left(\beta_{j,k} - \beta_{j,k-1}\right). \tag{4.48}$$

We will consider only second-order schemes in this chapter, but higher-order operators, as described in Sect. 2.2, can offer improved efficiency in certain contexts. If higher-order differencing operators are used, the metric terms should also be evaluated using the same first-derivative operator, boundary schemes of appropriate order[1] should be used, and the accuracy of other approximations in the algorithm, such as numerical integration to obtain forces, should also be raised to a consistent order.

At this point it is reasonable to ask whether the second-order centered difference formula remains second order on a nonuniform mesh when a curvilinear coordinate transformation is used. To address this question, consider a nonuniform mesh in one dimension, for which the coordinate transformation gives for a first derivative

$$\frac{\partial f}{\partial x} = \xi_x \frac{\partial f}{\partial \xi} = \frac{1}{x_\xi}\frac{\partial f}{\partial \xi}. \tag{4.49}$$

Application of second-order centered difference formulas to both $\partial f/\partial \xi$ and x_ξ at node j gives

$$(\delta_x f)_j = \frac{f_{j+1} - f_{j-1}}{x_{j+1} - x_{j-1}}. \tag{4.50}$$

[1] The order of a boundary scheme can often be one less than that of the interior scheme, and the global accuracy of the interior operator is still achieved (see Gustafsson [10]). The stability of boundary schemes for higher-order methods is also an important consideration but is beyond the scope of this book.

Denoting the mesh spacing immediately to the right of node j as

$$\Delta x_+ = x_{j+1} - x_j, \tag{4.51}$$

and that to the left as

$$\Delta x_- = x_j - x_{j-1}, \tag{4.52}$$

a Taylor series expansion of the derivative operator gives the following error term:

$$\frac{1}{2}\left(\frac{\partial^2 f}{\partial x^2}\right)_j (\Delta x_+ - \Delta x_-) + \frac{1}{6}\left(\frac{\partial^3 f}{\partial x^3}\right)_j \left(\frac{\Delta x_+^3 + \Delta x_-^3}{\Delta x_+ + \Delta x_-}\right) + \cdots . \tag{4.53}$$

The second term is clearly second order, but, at first glance, the first term appears to be first order. However, it is important to recall that the notion of the order of accuracy relates to the behavior of the error when a smooth mesh is refined uniformly.

For our present example, we can define a mesh function $x(\xi) = g(\xi/M) = g(\xi D)$, where M is the number of cells in the one-dimensional mesh, and $D = 1/M$ is a nominal mesh spacing parameter. For example, if the number of nodes M is doubled, then D is halved. With this mesh function, Taylor series expansions for Δx_+ and Δx_- give

$$\Delta x_+ = x_{j+1} - x_j = Dg'_j + \frac{1}{2}D^2 g''_j + \frac{1}{6}d^3 g'''_j + \cdots \tag{4.54}$$

and

$$\Delta x_- = x_j - x_{j-1} = Dg'_j - \frac{1}{2}D^2 g''_j + \frac{1}{6}d^3 g'''_j - \cdots . \tag{4.55}$$

Taking the difference gives

$$\Delta x_+ - \Delta x_- = D^2 g''_j + \cdots = O(D^2), \tag{4.56}$$

and we see that the error term remains second order, even on a nonuniform mesh. It is important to note that the error (4.53) contains a term proportional to $\partial^2 f/\partial x^2$, so, although it is second order, this approximation is not exact for a quadratic function, as is the case on a uniform mesh, where $\Delta x_+ - \Delta x_-$ is zero. One can easily define a finite-difference scheme on a nonuniform mesh that is exact for a quadratic function, but this approach extends to multiple dimensions in a straightforward manner only if the mesh is rectangular.

In order to make the above discussion more concrete, consider the one-dimensional mesh function

$$x(\xi) = \frac{e^{\xi/M} - 1}{e - 1}. \tag{4.57}$$

This function produces a uniform stretching ratio given by

$$\frac{\Delta x_+}{\Delta x_-} = \frac{\mathrm{e}^{1/M} - 1}{1 - \mathrm{e}^{-1/M}}. \tag{4.58}$$

With $M = 10$, the stretching ratio is roughly 1.105; if M is increased to 100, the stretching ratio is reduced to roughly 1.010. With each increase in M, not only does the mesh spacing decrease in proportion to $1/M$, but the stretching ratio also decreases. Consequently, the difference $\Delta x_+ - \Delta x_-$ is of order $(1/M)^2$. If mesh refinement is performed such that the stretching ratio is constant, this is not a suitable refinement and the second-order behaviour of the difference operator (4.50) will not be observed .

4.4.1 Metric Differencing and Invariants

The second-order centered difference formulas used in two dimensions naturally produce consistent metric invariants, but in three dimensions some additional measures must be taken to ensure this property. Examining one of these terms in two dimensions, $\partial_\xi(y_\eta) + \partial_\eta(-y_\xi)$, using second-order centered differences both to form the metric terms and to approximate the flux derivatives, we obtain

$$\begin{aligned}
\delta_\xi \delta_\eta y_{j,k} - \delta_\eta \delta_\xi y_{j,k} &= \delta_\xi (y_{j,k+1} - y_{j,k-1})/2 - \delta_\eta (y_{j+1,k} - y_{j-1,k})/2 \\
&= [y_{j+1,k+1} - y_{j-1,k+1} - y_{j+1,k-1} + y_{j-1,k-1}]/4 \\
&\quad -[y_{j+1,k+1} - y_{j+1,k-1} - y_{j-1,k+1} + y_{j-1,k-1}]/4 \\
&= 0,
\end{aligned} \tag{4.59}$$

as desired.

In three dimensions, there are several different ways to ensure that these terms are zero. For example, consider the metric ξ_x, which is given by[2]

$$\xi_x = J(y_\eta z_\zeta - y_\zeta z_\eta), \tag{4.60}$$

where z and ζ are the third coordinate directions in Cartesian and computational space, respectively. One approach is to form ξ_x through the following formula:

$$\xi_x = J\left[(\mu_\zeta \delta_\eta y)(\mu_\eta \delta_\zeta z) - (\mu_\eta \delta_\zeta y)(\mu_\zeta \delta_\eta z)\right], \tag{4.61}$$

where δ is the second-order centered difference operator, and μ is an averaging operator defined, for example, by $\mu_\eta x_{j,k,l} = (x_{j,k+1,l} + x_{j,k-1,l})/2$, where l is the

[2] See Sect. 4.7.1

index in the ζ direction. If all of the metric terms are calculated in this manner, then the metric invariants will be satisfied.

An alternative approach in three dimensions that extends to higher order involves writing the expression for ξ_x as [11]:

$$\xi_x = J((y_\eta z)_\zeta - (y_\zeta z)_\eta), \tag{4.62}$$

which is analytically equivalent to (4.60). Analogous expressions can be written for the other metrics of the transformation. If consistent centered difference formulas are used for both the derivatives in such expressions for the metric terms as well as the flux derivatives, e.g. $\delta_\xi \widehat{E}$, then the metric invariants will be zero (within the limits of round-off error).

In (4.59) we saw that second-order centered differencing of both the metric relations and the flux derivatives leads to satisfaction of the invariant relations in two dimensions. However, consider the case of centered differencing to form the metrics combined with first-order one-sided backward differencing for the fluxes. We obtain

$$\begin{aligned}\nabla_\xi \delta_\eta y - \nabla_\eta \delta_\xi y = {} & [y_{j,k+1} - y_{j-1,k+1} - y_{j,k-1} + y_{j-1,k-1}]/2 \\ & + [-y_{j+1,k} + y_{j+1,k-1} + y_{j-1,k} - y_{j-1,k-1}]/2 \neq 0.\end{aligned} \tag{4.63}$$

The error associated with not satisfying the invariant relations is a truncation error that corresponds to the order of the lowest-order-accurate operator used or higher.

4.4.2 Artificial Dissipation

The concept of numerical dissipation was introduced in Sect. 2.5. Numerical dissipation can be added to a spatial discretization for three distinct purposes:

- to eliminate high-frequency modes that are not resolved and can contaminate the solution;
- to enhance stability and convergence to steady state;
- to prevent oscillations at discontinuities, such as shock waves.

The idea is to achieve these three purposes by introducing a level of numerical dissipation that does not significantly increase the overall numerical error.

In linear problems, such as the linear convection equation, the frequencies or wavenumbers present in the solution are dictated by the initial and boundary conditions. In the numerical solution of such equations, the components with wavenumbers that are not well resolved (see Fig. 2.2) are essentially spurious. They will not be handled accurately by the numerical scheme in terms of either convection or diffusion. Therefore, it can be worthwhile to remove them through numerical dissipation or filtering.

In solutions of the Euler and Navier-Stokes equations, nonlinear interactions occur between waves as a result of the nonlinearity in the convection terms of the momentum equations. If scale is represented by wavelength or frequency, it can be shown that two waves interact as products to form a wave of higher frequency (the sum of the original two) and one of lower frequency (the difference). In a physical system, this can lead to turbulence and the formation of shock waves. As a result of viscosity, there is a limit to the smallest length scales that arise. Numerically, if all scales are well resolved, for example in a well-resolved direct numerical simulation of a turbulent flow or a well-resolved simulation of a laminar flow, then numerical dissipation is not needed. However, in most flow computations, these smallest scales are typically not resolved. As a result, the true physical mechanism that puts an upper bound on the frequencies present in the solution is not accurately represented in the numerical solution. The lower frequencies do not cause a problem, but the continual cascading into higher and higher frequencies can lead to instabilities. These can be addressed through numerical dissipation. Even in linear problems, instabilities can arise from numerical implementation of boundary conditions and other approximations that might cause some eigenvalues of the semi-discrete operator matrix to lie slightly in the right half-plane. Numerical dissipation can address such instabilities as well and speed up convergence to a steady state.

The Euler equations support discontinuities such as shock waves, slip lines, and contact surfaces. Across these discontinuities, the differential form of the PDEs does not apply, so the appropriate jump conditions must be determined from the integral form. In essence, shock waves are a limiting case of the frequency cascade described in the previous paragraph. The Euler equations contain no mechanism to limit the minimum length scale, so a shock wave is a true discontinuity in an inviscid flow. In a real viscous flow, shock waves have a finite thickness, but it is so small that it is rarely practical to resolve a shock, and in any case it is not clear that the continuum hypothesis would be applicable within a shock wave. Therefore, although the Navier-Stokes equations do not support discontinuities, the issue of the numerical treatment of shock waves is present even in computations of viscous flows. Without a careful treatment, oscillations will occur at and near shock waves and other discontinuities.

Historically, the numerical treatment of shock waves has been divided into two approaches, shock fitting and shock capturing. In shock fitting, one must know the location of the shock and apply the jump conditions across it. While this is an inherently elegant approach, in practice it is very difficult to track the precise location of shock waves. As a result, shock capturing, in which the shock wave is smoothed out by numerical dissipation and the flow is treated as if it were continuous, has become the predominant approach.

A substantial amount of research has gone into the development of numerical methods for capturing shocks. We will cover such methods in more detail in Chap. 6. For our purpose here it suffices to say that in order to prevent oscillations, first-order numerical dissipation is needed in the vicinity of discontinuities. However, use of first-order numerical dissipation throughout the flow domain would lead to very large numerical errors, or, alternatively, the need for a very fine mesh to reduce numerical errors to the desired levels. Consequently, the numerical dissipation added

to a spatial discretization of the Euler or Navier-Stokes equations generally consists of the following three components:

- a high-order component for smooth regions of the flow field,
- a first-order component for shock capturing,
- a means of sensing shocks and other discontinuities so that the appropriate dissipation operator can be selected in different regions of the flow field.

Before continuing, the reader may wish to review Sect. 2.5, which introduced the basic concepts underlying numerical dissipation. The dissipation is associated with the symmetric part of the difference operator and can be added either explicitly through artificial dissipation or through one-sided or upwind schemes that inherently include a symmetric component. In this chapter and the next we will concentrate on centered schemes with added artificial dissipation, while Chap. 6 discusses upwind schemes in more detail. The close relationship between the two approaches is clear from Sect. 2.5.

4.4.3 A Nonlinear Artificial Dissipation Scheme

Recalling Sect. 2.5, numerical dissipation can be added to a centered differencing scheme by adding a symmetric component to the difference operator approximating the first derivatives in the inviscid flux terms. For a constant-coefficient, linear hyperbolic system of equations in the form

$$\frac{\partial u}{\partial t} + \frac{\partial f}{\partial x} = \frac{\partial u}{\partial t} + A\frac{\partial u}{\partial x} = 0, \tag{4.64}$$

where $f = Au$, the dissipation can be added in the following manner:

$$\delta_x f = \delta_x^{\mathrm{a}} f + \delta_x^{\mathrm{s}}(|A|u), \tag{4.65}$$

where δ_x^{a} and δ_x^{s} are antisymmetric and symmetric difference operators, X is the matrix of right eigenvectors of A, Λ is a diagonal matrix containing the eigenvalues of A, and $|A| = X|\Lambda|X^{-1}$. The antisymmetric operator is simply the centered difference scheme, and the symmetric operator introduces the dissipation.

An antisymmetric or centered difference operator for a first derivative has an even order of accuracy, while the symmetric term has an odd order of accuracy. For smooth regions of the flow, the symmetric operator should be at least third order, since a first-order term is generally too dissipative and will add too much numerical error. With second-order centered differences, a third-order dissipation term is thus a good choice for regions where the flow variables behave smoothly, i.e. away from discontinuities.

Consequently, the following symmetric operator is often used together with second-order centered differences:

$$\left(\delta_x^s u\right)_j = \frac{\epsilon_4}{\Delta x}(u_{j-2} - 4u_{j-1} + 6u_j - 4u_{j+1} + u_{j+2}) \propto \epsilon_4 \Delta x^3 \frac{\partial^4 u}{\partial x^4}, \quad (4.66)$$

where ϵ_4 is a user defined constant. This operator is sufficient to damp unwanted high-frequency modes and provide stability while generally adding an error that is smaller than the second-order error associated with the centered difference scheme. However, it is not sufficient to prevent oscillations at discontinuities. For this purpose, the following first-order symmetric operator is typically used:

$$\left(\delta_x^s u\right)_j = \frac{\epsilon_2}{\Delta x}(-u_{j-1} + 2u_j - u_{j+1}) \propto -\epsilon_2 \Delta x \frac{\partial^2 u}{\partial x^2}. \quad (4.67)$$

The artificial dissipation scheme used in the implicit finite-difference algorithm of this chapter combines the above two operators using a pressure sensor to detect shock waves [6, 12]. This approach is intended for flows with shock waves, where the pressure is discontinuous; it will not sense a discontinuity such as a contact surface across which the pressure is continuous. Before presenting the operator, we note that

$$\nabla \Delta \nabla \Delta u_j = u_{j-2} - 4u_{j-1} + 6u_j - 4u_{j+1} + u_{j+2} \quad (4.68)$$

and

$$\nabla \Delta u_j = u_{j-1} - 2u_j + u_{j+1}, \quad (4.69)$$

where $\nabla u_j = u_j - u_{j-1}$ and $\Delta u_j = u_{j+1} - u_j$ are undivided differences.

Before moving to the two-dimensional equations in curvilinear coordinates, let us first consider the one-dimensional Euler equations (3.24):

$$\frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} = 0, \quad (4.70)$$

where $E = AQ$ as a result of the homogeneous property of the Euler equations (see [13], Appendix C). A natural application of (4.65) and (4.66) gives a fourth-difference dissipative term in the following form:

$$D_j = \nabla \Delta \nabla \Delta |A_j| Q_j. \quad (4.71)$$

In the constant-coefficient, linear case, $|A|$ is constant, but that is no longer true in the nonlinear case, and hence its position in the above equation can have a significant effect. For example, the choice

$$D_j = |A_j| \nabla \Delta \nabla \Delta Q_j \quad (4.72)$$

is not conservative. The preferred choice, motivated by analogy to flux-difference splitting, is

$$D_j = \nabla |A_{j+1/2}| \Delta \nabla \Delta Q_j, \tag{4.73}$$

where $A_{j+1/2}$ is some sort of average, such as a simple average or a Roe average (see Sect. 6.3).

Now consider the strong conservation law form of the Navier-Stokes equations in generalized curvilinear coordinates (4.18) with the spatial derivatives replaced by second-order centered differences, as in (4.43), and all of the spatial terms moved to the right-hand side:

$$\partial_\tau \widehat{Q} = -\delta_\xi \widehat{E} - \delta_\eta \widehat{F} + Re^{-1}[\delta_\xi \widehat{E}_\mathrm{v} + \delta_\eta \widehat{F}_\mathrm{v}], \tag{4.74}$$

where the compact three-point form (4.48) is assumed for the viscous derivatives. Let us restrict our interest for now to the inviscid term in the ξ direction, giving

$$\partial_\tau \widehat{Q} = -\delta_\xi \widehat{E}. \tag{4.75}$$

This can be written in the *conservation form*

$$\partial_\tau \widehat{Q} = -(f_{j+1/2} - f_{j-1/2}), \tag{4.76}$$

where

$$f_{j+1/2} = \frac{1}{2}(\widehat{E}_j + \widehat{E}_{j+1}). \tag{4.77}$$

Thus the discrete form applied to the conservation law form of the equation preserves the conservative property of the original PDE. It is important that the artificial dissipation scheme maintain this property.

We now introduce an artificial dissipation term $(D_\xi)_{j,k}$ in the ξ direction into (4.75) as follows:

$$(\partial_\tau \widehat{Q})_{j,k} = -(\delta_\xi \widehat{E})_{j,k} + (D_\xi)_{j,k}, \tag{4.78}$$

where

$$\begin{aligned}(D_\xi)_{j,k} = {} & \nabla_\xi \left(\epsilon^{(2)} |\widehat{A}| J^{-1}\right)_{j+1/2,k} \Delta_\xi Q_{j,k} \\ & -\nabla_\xi \left(\epsilon^{(4)} |\widehat{A}| J^{-1}\right)_{j+1/2,k} \Delta_\xi \nabla_\xi \Delta_\xi Q_{j,k}.\end{aligned} \tag{4.79}$$

Analogous terms are used in the η direction. There are many aspects to this expression; these will be explained one at a time. The first term on the right-hand side is the second-difference term, which is first order, that is needed near shock waves. The second term is the fourth-difference term, which is third order, that is used in smooth regions of the flow field. Their relative contributions are controlled by the two coefficients $\epsilon^{(2)}$ and $\epsilon^{(4)}$, which are defined below. Next, $\widehat{A}$ is the flux Jacobian

in the ξ direction defined as follows:

$$\widehat{A} = \frac{\partial \widehat{E}}{\partial \widehat{Q}}. \tag{4.80}$$

This is given in Sect. 4.5.

Notice that the dissipation operates on Q, not $\widehat{Q}$; J^{-1} is moved together with $|\widehat{A}|$. This ensures that no dissipation is generated for a uniform flow. On a nonuniform mesh, $\widehat{Q}$ is not constant in space, even if Q is constant, as a result of the spatial variation of J^{-1}. Consequently, nonzero dissipation would arise in a uniform flow if the dissipation were to operate on $\widehat{Q}$.

The location of the terms $\epsilon^{(2)}|\widehat{A}|J^{-1}$ and $\epsilon^{(4)}|\widehat{A}|J^{-1}$ is consistent with (4.73). These can be evaluated through simple averages, e.g.

$$\left(\epsilon^{(2)}|\widehat{A}|J^{-1}\right)_{j+1/2,k} = \frac{1}{2}\left[\left(\epsilon^{(2)}|\widehat{A}|J^{-1}\right)_{j,k} + \left(\epsilon^{(2)}|\widehat{A}|J^{-1}\right)_{j+1,k}\right] \tag{4.81}$$

$$\left(\epsilon^{(4)}|\widehat{A}|J^{-1}\right)_{j+1/2,k} = \frac{1}{2}\left[\left(\epsilon^{(4)}|\widehat{A}|J^{-1}\right)_{j,k} + \left(\epsilon^{(4)}|\widehat{A}|J^{-1}\right)_{j+1,k}\right], \tag{4.82}$$

or a Roe average can be used for $\widehat{A}_{j+1/2,k}$.

The contribution of the second-difference term is controlled by a pressure sensor that detects shock waves [6, 12]. It is defined as follows:

$$\begin{aligned}
\epsilon^{(2)}_{j,k} &= \kappa_2 \max(\Upsilon_{j+1,k}, \Upsilon_{j,k}, \Upsilon_{j-1,k}) \\
\Upsilon_{j,k} &= \left|\frac{p_{j+1,k} - 2p_{j,k} + p_{j-1,k}}{p_{j+1,k} + 2p_{j,k} + p_{j-1,k}}\right| \\
\epsilon^{(4)}_{j,k} &= \max(0, \kappa_4 - \epsilon^{(2)}_{j,k}),
\end{aligned} \tag{4.83}$$

where typical values of the constants are $\kappa_2 = 1/2$ and $\kappa_4 = 1/50$. The switch is based on a normalized undivided second difference of pressure, which is much larger at shock waves than in smooth regions. The logic turns off the fourth-difference dissipation when the second-difference coefficient is large. The max function spreads out the contribution of the second-difference dissipation to ensure that it is not switched off in the interior of the shock.

Consistent with (4.76), the dissipative term can be written as

$$(D_\xi)_{j,k} = (d_\xi)_{j+1/2,k} - (d_\xi)_{j-1/2,k}, \tag{4.84}$$

where

$$(d_\xi)_{j+1/2,k} = \left(\epsilon^{(2)}|\widehat{A}|J^{-1}\right)_{j+1/2,k} \Delta_\xi Q_{j,k} - \left(\epsilon^{(4)}|\widehat{A}|J^{-1}\right)_{j+1/2,k} \Delta_\xi \nabla_\xi \Delta_\xi Q_{j,k}. \tag{4.85}$$

This ensures that the dissipation is conservative.

In order to reduce the cost of the dissipation model, one can replace the matrix $|\widehat{A}|$ with the spectral radius of $\widehat{A}$, which is its largest eigenvalue by absolute value. The spectral radius of $\widehat{A}$ is given by

$$\sigma = |U| + a\sqrt{\xi_x^2 + \xi_y^2}. \tag{4.86}$$

The spectral radius of $\widehat{B}$ is used for the η dissipation term. This approach, known as scalar artificial dissipation, leads to an inexpensive artificial dissipation scheme that is robust but can be excessively dissipative in certain contexts.

The astute reader may be wondering where the Δx terms in (4.66) and (4.67) have gone. These are implicit in $\widehat{A}$ and the spectral radius σ through the metric terms ξ_x and ξ_y, which scale with the inverse of the mesh spacing. This ensures that the two dissipation operators in (4.79) are first order and third order as desired.

Let us consider the fourth-difference dissipation term in more detail. Temporarily ignoring the coefficient term, we have

$$\begin{aligned}(D_\xi^{(4)})_{j,k} &= -\nabla_\xi \Delta_\xi \nabla_\xi \Delta_\xi Q_{j,k} \\ &= -Q_{j-2,k} + 4Q_{j-1,k} - 6Q_{j,k} + 4Q_{j+1,k} - Q_{j+2}.\end{aligned} \tag{4.87}$$

This operator involves values of Q from $j-2, k$ to $j+2, k$, i.e. a five-point stencil, in contrast to the finite-difference approximations to the inviscid and viscous flux derivatives, which involve data from $j-1$ to $j+1$ only, i.e. a three-point stencil. As we shall see in Sect. 4.5, this has significant implications for an implicit time-marching method. Here we are concerned with its implications near the boundaries of the grid. Boundary conditions are discussed later in this chapter. For now we will assume that the values of Q at the boundary are known, so the governing equations are not solved at the boundary. At the first interior node, the three-point operators for the inviscid and viscous fluxes as well as the second-difference dissipation can be applied without modification. However, the five-point operator cannot be applied, as either $Q_{j-2,k}$ or $Q_{j+2,k}$ is unavailable, depending on the boundary.

In developing a boundary scheme for the fourth-difference dissipation operator, one must ensure that the resulting scheme is conservative, dissipative, stable, and sufficiently accurate globally. First, we will consider conservation. The operator in (4.87) can be rewritten as

$$(D_\xi^{(4)})_{j,k} = (d_\xi^{(4)})_{j+1/2,k} - (d_\xi^{(4)})_{j-1/2,k}, \tag{4.88}$$

where

$$(d_\xi^{(4)})_{j+1/2,k} = Q_{j-1,k} - 3Q_{j,k} + 3Q_{j+1,k} - Q_{j+2,k}. \tag{4.89}$$

Without loss of generality, we will consider a boundary located at $j = 0$. The operator at $j = 1$ must be modified because the node $j - 2$ does not exist. Since the operator at $j = 2$ is not modified, conservation dictates that the term $(d_\xi^{(4)})_{j+1/2,k}$ at $j = 1$ cannot be modified. In any case, this term does not involve $Q_{j-2,k}$, so it need not be modified. There are several different ways to proceed; one is to define $(d_\xi^{(4)})_{j-1/2,k}$ at node $j = 1$ as

$$(d_\xi^{(4)})_{j-1/2,k} = -Q_{j-1,k} + 2Q_{j,k} - Q_{j+1,k}. \tag{4.90}$$

This leads to the following operator for the node at $j = 1$:

$$\begin{aligned}(D_\xi^{(4)})_{j,k} &= (Q_{j-1,k} - 3Q_{j,k} + 3Q_{j+1,k} - Q_{j+2,k}) \\ &\quad -(-Q_{j-1,k} + 2Q_{j,k} - Q_{j+1,k}) \\ &= 2Q_{j-1,k} - 5Q_{j,k} + 4Q_{j+1,k} - Q_{j+2,k}.\end{aligned} \tag{4.91}$$

Similar formulas are used at other boundaries. This approach has been shown to be dissipative and stable [14] and is therefore popular, although other options are also used. This boundary operator is first-order accurate locally and consistent with second-order global accuracy. If the interior scheme has an order of accuracy greater than two, then a higher order boundary operator should be used for the fourth-difference dissipation. Similarly, if better than third-order global accuracy is desired, then an artificial dissipation scheme of higher order is needed.

We conclude this section with a brief discussion of the application of this artificial dissipation scheme to the quasi-one-dimensional Euler equations, which are the subject of the exercises at the end of this chapter. The problems are to be solved on a uniform grid using the scalar artificial dissipation scheme. The spectral radius of the one-dimensional flux Jacobian matrix is

$$\sigma = |u| + a. \tag{4.92}$$

Since the mesh is uniform, no coordinate transformation is needed. The dissipation terms thus become

$$\begin{aligned}D_j = &\frac{1}{\Delta x}\nabla\left(\epsilon^{(2)}(|u| + a)\right)_{j+1/2}\Delta Q_j \\ &-\frac{1}{\Delta x}\nabla\left(\epsilon^{(4)}(|u| + a)\right)_{j+1/2}\Delta\nabla\Delta Q_j,\end{aligned} \tag{4.93}$$

where ∇ and Δ denote undivided differences. Note in particular the $1/\Delta x$ scaling.

4.5 Implicit Time Marching and the Approximate Factorization Algorithm

After application of the above spatial discretization to (4.18), we obtain the following semi-discrete equation at each interior node in the mesh:

$$\partial_\tau \widehat{Q} = -\delta_\xi \widehat{E} + D_\xi - \delta_\eta \widehat{F} + D_\eta + Re^{-1}[\delta_\xi \widehat{E}_\mathrm{v} + \delta_\eta \widehat{F}_\mathrm{v}], \tag{4.94}$$

where δ represents the spatial difference operator, in this case second-order centered differences, and D_ξ and D_η the artificial dissipation terms, e.g. (4.79). Collecting these into a single equation, we obtain the following coupled system of nonlinear ODEs:

$$\frac{\mathrm{d}\widehat{\mathbf{Q}}}{\mathrm{d}t} = \mathbf{R}(\widehat{\mathbf{Q}}), \tag{4.95}$$

where $\widehat{\mathbf{Q}}$ is a column matrix containing $\widehat{Q}_{j,k}$ at each node of the mesh, $\mathbf{R}$ is a column matrix containing $R_{j,k}$ at each node, where

$$R(\widehat{Q}) = -\delta_\xi \widehat{E} + D_\xi - \delta_\eta \widehat{F} + D_\eta + Re^{-1}[\delta_\xi \widehat{E}_\mathrm{v} + \delta_\eta \widehat{F}_\mathrm{v}], \tag{4.96}$$

and we have replaced τ with t. In order to obtain a time-accurate solution for an unsteady flow problem, this system of ODEs must be solved using a time-marching method. Alternatively, if the flow under consideration is steady, one seeks the solution to the following coupled system of nonlinear algebraic equations:

$$\mathbf{R}(\widehat{\mathbf{Q}}) = \mathbf{0}. \tag{4.97}$$

In the steady case, $\mathbf{R}(\widehat{\mathbf{Q}})$ is referred to as the residual vector, or simply the residual. As a result of the nonlinear nature of the residual vector, this system cannot be solved directly: an iterative method is required.

For the numerical solution of a large system of nonlinear algebraic equations such as (4.97), it is natural to consider the Newton method, which produces the following linear system:

$$\mathbf{A}_n \Delta\widehat{\mathbf{Q}}_n = -\mathbf{R}(\widehat{\mathbf{Q}}_n), \tag{4.98}$$

where

$$\mathbf{A}_n = \frac{\partial \mathbf{R}}{\partial \widehat{\mathbf{Q}}} \tag{4.99}$$

is the Jacobian evaluated at state $\widehat{\mathbf{Q}}_n$, and $\Delta\widehat{\mathbf{Q}} = \widehat{\mathbf{Q}}_{n+1} - \widehat{\mathbf{Q}}_n$. This linear system must be solved iteratively until a converged solution is obtained that satisfies (4.97).

The degree to which a given iterate $\widehat{\mathbf{Q}}_n$ is a solution to (4.97) can be measured through the norm of $\mathbf{R}(\widehat{\mathbf{Q}})$. In finite precision arithmetic, it is typically not possible to reduce the norm of the residual below machine zero, so a solution for which this norm is on the order of machine zero can be considered fully converged. However, with single precision arithmetic, this level of convergence may not be sufficient.

Application of the Newton method to the large systems of nonlinear algebraic equations arising from the spatial discretization of the Euler or Navier-Stokes equations in multiple dimensions leads to two principal challenges. First, the Newton method converges only from an iterate that is within a finite region of convergence near the solution. Typically, the initial guess for $\widehat{\mathbf{Q}}$ lies outside this region, and some sort of globalization technique is needed to ensure that the Newton method will converge for an arbitrary initial iterate. A uniform flow is often used as the initial iterate. Second, the linear system of equations (4.98) that must be solved is in general large and sparse. Direct solution of such systems based on a lower-upper (LU) factorization can require a large amount of memory relative to the original sparse system and a number of floating point operations that scales poorly as the system size increases. Hence direct solution is only effective for linear systems below a certain size, although the system size for which direct solution of the system is a feasible approach increases with each new generation of computer hardware. The high cost of direct solution of this linear system for problems of practical interest motivates *inexact Newton methods* in which the linear system (4.98) is instead solved iteratively to some tolerance at each iteration. Sequences of tolerances can be found that maintain the quadratic convergence property of the Newton method within the radius of convergence, provided the residual function meets certain conditions.

A natural way to address the problem that the initial iterate is likely outside the region of convergence of the Newton method is to consider a time-dependent path to steady state. Under certain conditions, the solution of the steady problem (4.97) is also the steady solution of the ODE system (4.95), which can be found by applying a time-marching method to (4.95) and advancing in time until a steady state is reached. Time accuracy is not required; we simply wish to integrate in time from some arbitrary initial state to the steady solution in a manner that will require the smallest amount of computational work. The entire transient portion of the solution can be considered parasitic, and hence the problem is stiff. This suggests the use of an implicit time-marching method, and, given that we are not interested in time resolution of the transient, there is no reason to seek better than first-order accuracy. Therefore the implicit Euler method is the logical choice for steady problems. Its relationship with the Newton method is discussed in Sect. 2.6.3.

For unsteady flow problems where time-accurate solutions are required, one would like at least second-order accuracy. Hence, the trapezoidal and second-order backward methods (see Sect. 2.6), which are both unconditionally stable, are reasonable choices. The second-order backward method has a larger region of stability than the trapezoidal method, making it the more robust of the two. Moreover, the trapezoidal method provides little damping of modes with eigenvalues with large negative real parts, which is undesirable for stiff problems. Implicit Runge-Kutta methods, which we will not discuss here, are another option for time-accurate solution of stiff ODEs.

This brings us to the challenge of solving a large sparse linear system, which is present whether one is solving steady or unsteady problems. Historically, due to computer hardware limitations, direct solution techniques were not practical even for relatively small problems. Even today they are not an efficient option for large-scale three-dimensional problems. Inexact Newton methods have gained in popularity since the introduction of efficient iterative techniques for nonsymmetric sparse linear systems, such as the generailized minimal residual method (GMRES) [15]. However, these were not available until the mid-1980s, so the first implicit computations of three-dimensional flows were performed using the now classical approximate factorization algorithm, which is the subject of Sect. 4.5.4.

4.5.1 Implicit Time-Marching

Based on the above discussion, whether we are solving an unsteady problem or a steady one, we seek to solve the coupled system of ODEs given by (4.95) using an implicit time-marching method. We will consider the following two-parameter family of time-marching methods [3]:

$$\widehat{\mathbf{Q}}^{n+1} = \frac{\theta\Delta t}{1+\varphi}\frac{\mathrm{d}}{\mathrm{d}t}\widehat{\mathbf{Q}}^{n+1} + \frac{(1-\theta)\Delta t}{1+\varphi}\frac{\mathrm{d}}{\mathrm{d}t}\widehat{\mathbf{Q}}^{n} + \frac{1+2\varphi}{1+\varphi}\widehat{\mathbf{Q}}^{n} - \frac{\varphi}{1+\varphi}\widehat{\mathbf{Q}}^{n-1} + O\left[(\theta - \frac{1}{2} - \varphi)\Delta t^2 + \Delta t^3\right], \tag{4.100}$$

where $\widehat{\mathbf{Q}}^n = \widehat{\mathbf{Q}}(n\Delta t)$. This family of methods is a subset of two-step linear multistep methods with the coefficient of

$$\frac{\mathrm{d}}{\mathrm{d}t}\widehat{\mathbf{Q}}^{n-1} \tag{4.101}$$

set to zero. One member of the family is third-order accurate, but that method is not of interest here, as it is not unconditionally stable. Our interest is in the first-order implicit Euler method obtained with $\theta = 1$ and $\varphi = 0$ for steady problems and the second-order backward method obtained with $\theta = 1$ and $\varphi = 1/2$ when time-accuracy is required.

For this exposition we will restrict ourselves to the implicit Euler method, but all of the subsequent development can easily be extended to any second-order scheme formed from (4.100). Applying the implicit Euler method to the thin-layer form of (4.95) results in the following expression at each node of the grid:

$$\widehat{Q}^{n+1} - \widehat{Q}^{n} = h\left(-\delta_\xi \widehat{E}^{n+1} + D_\xi^{n+1} - \delta_\eta \widehat{F}^{n+1} + D_\eta^{n+1} + Re^{-1}\delta_\eta \widehat{S}^{n+1}\right), \tag{4.102}$$

with $h = \Delta t$.

4.5.2 Local Time Linearization

We wish to solve (4.102) for $\widehat{Q}^{n+1}$ given $\widehat{Q}^n$. The flux vectors $\widehat{E}$, $\widehat{F}$, and $\widehat{S}$, and the artificial dissipation terms D_ξ, and D_η, are nonlinear functions of $\widehat{Q}$, and therefore the right-hand side of (4.102) is nonlinear in $\widehat{Q}^{n+1}$. Hence we proceed by locally linearizing with respect to t.

The flux vectors are linearized in time about $\widehat{Q}^n$ by Taylor series such that

$$\begin{aligned}
\widehat{E}^{n+1} &= \widehat{E}^n + \widehat{A}^n \Delta\widehat{Q}^n + O(h^2) \\
\widehat{F}^{n+1} &= \widehat{F}^n + \widehat{B}^n \Delta\widehat{Q}^n + O(h^2) \\
Re^{-1}\widehat{S}^{n+1} &= Re^{-1}\left[\widehat{S}^n + \widehat{M}^n \Delta\widehat{Q}^n\right] + O(h^2),
\end{aligned} \tag{4.103}$$

where $\widehat{A} = \partial\widehat{E}/\partial\widehat{Q}$, $\widehat{B} = \partial\widehat{F}/\partial\widehat{Q}$ and $\widehat{M} = \partial\widehat{S}/\partial\widehat{Q}$ are the flux Jacobians, and $\Delta\widehat{Q}^n$ is $O(h)$. As discussed in Sect. 2.6.3, such a local time linearization will not degrade the order of accuracy of time-marching methods of up to second order.

The inviscid flux Jacobian matrices $\widehat{A}$ and $\widehat{B}$ are given by

$$\begin{bmatrix}
\kappa_t & \kappa_x & \kappa_y & 0 \\
-u\theta + \kappa_x\phi^2 & \kappa_t + \theta - (\gamma-2)\kappa_x u & \kappa_y u - (\gamma-1)\kappa_x v & (\gamma-1)\kappa_x \\
-v\theta + \kappa_y\phi^2 & \kappa_x v - (\gamma-1)\kappa_y u & \kappa_t + \theta - (\gamma-2)\kappa_y v & (\gamma-1)\kappa_y \\
\theta[\phi^2 - a_1] & \kappa_x a_1 - (\gamma-1)u\theta & \kappa_y a_1 - (\gamma-1)v\theta & \gamma\theta + \kappa_t
\end{bmatrix}, \tag{4.104}$$

with $a_1 = \gamma(e/\rho) - \phi^2, \theta = \kappa_x u + \kappa_y v, \phi^2 = \frac{1}{2}(\gamma-1)(u^2+v^2)$, and $\kappa = \xi$ or η for $\widehat{A}$ or $\widehat{B}$, respectively. As an example, we will derive the first element in the second row of $\widehat{A}$, i.e.

$$\widehat{a}_{21} = \frac{\partial\widehat{e}_2}{\partial\widehat{q}_1}, \tag{4.105}$$

where

$$\widehat{Q} = \begin{bmatrix} \widehat{q}_1 \\ \widehat{q}_2 \\ \widehat{q}_3 \\ \widehat{q}_4 \end{bmatrix} = J^{-1}\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \widehat{E} = \begin{bmatrix} \widehat{e}_1 \\ \widehat{e}_2 \\ \widehat{e}_3 \\ \widehat{e}_4 \end{bmatrix} = J^{-1}\begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ U(e+p) - \xi_t p \end{bmatrix}. \tag{4.106}$$

In order to find $\widehat{a}_{21}$, the first step is to write $\widehat{e}_2$ in terms of the elements of $\widehat{Q}$. One obtains

$$\begin{aligned}\widehat{e_2} &= J^{-1}\rho u U + J^{-1}\xi_x p \\ &= J^{-1}\rho u\xi_t + J^{-1}\rho u^2\xi_x + J^{-1}\rho uv\xi_y \\ &\quad + J^{-1}\xi_x(\gamma-1)e - J^{-1}\xi_x(\gamma-1)\frac{1}{2}\rho u^2 - J^{-1}\xi_x(\gamma-1)\frac{1}{2}\rho v^2 \\ &= \xi_t\widehat{q}_2 + \xi_x\frac{\widehat{q}_2^2}{\widehat{q}_1} + \xi_y\frac{\widehat{q}_2\widehat{q}_3}{\widehat{q}_1} + \xi_x(\gamma-1)\widehat{q}_4 - \frac{\xi_x(\gamma-1)}{2}\frac{\widehat{q}_2^2}{\widehat{q}_1} - \frac{\xi_x(\gamma-1)}{2}\frac{\widehat{q}_3^2}{\widehat{q}_1}.\end{aligned} \tag{4.107}$$

From this we find

$$\begin{aligned}\widehat{a}_{21} = \frac{\partial\widehat{e}_2}{\partial\widehat{q}_1} &= -\xi_x\frac{\widehat{q}_2^2}{\widehat{q}_1^2} - \xi_y\frac{\widehat{q}_2\widehat{q}_3}{\widehat{q}_1^2} + \frac{\xi_x(\gamma-1)}{2}\frac{\widehat{q}_2^2}{\widehat{q}_1^2} + \frac{\xi_x(\gamma-1)}{2}\frac{\widehat{q}_3^2}{\widehat{q}_1^2} \\ &= -\xi_x u^2 - \xi_y uv + \frac{\xi_x(\gamma-1)}{2}u^2 + \frac{\xi_x(\gamma-1)}{2}v^2 \\ &= -u(\xi_x u + \xi_y v) + \frac{\xi_x(\gamma-1)}{2}(u^2+v^2),\end{aligned} \tag{4.108}$$

consistent with (4.104). The other terms in $\widehat{A}$ and $\widehat{B}$ are found in a similar manner.

The thin-layer viscous flux Jacobian is

$$\widehat{M} = J^{-1}\begin{bmatrix} 0 & 0 & 0 & 0 \\ m_{21} & \alpha_1\partial_\eta(\rho^{-1}) & \alpha_2\partial_\eta(\rho^{-1}) & 0 \\ m_{31} & \alpha_2\partial_\eta(\rho^{-1}) & \alpha_3\partial_\eta(\rho^{-1}) & 0 \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} J, \tag{4.109}$$

where

$$\begin{aligned} m_{21} &= -\alpha_1\partial_\eta(u/\rho) - \alpha_2\partial_\eta(v/\rho) \\ m_{31} &= -\alpha_2\partial_\eta(u/\rho) - \alpha_3\partial_\eta(v/\rho) \\ m_{41} &= \alpha_4\partial_\eta\left[-(e/\rho^2) + (u^2+v^2)/\rho\right] \\ &\quad -\alpha_1\partial_\eta(u^2/\rho) - 2\alpha_2\partial_\eta(uv/\rho) \\ &\quad -\alpha_3\partial_\eta(v^2/\rho) \\ m_{42} &= -\alpha_4\partial_\eta(u/\rho) - m_{21} \\ m_{43} &= -\alpha_4\partial_\eta(v/\rho) - m_{31} \\ m_{44} &= \alpha_4\partial_\eta(\rho^{-1}) \\ \alpha_1 &= \mu[(4/3)\eta_x{}^2 + \eta_y{}^2], \quad \alpha_2 = (\mu/3)\eta_x\eta_y \\ \alpha_3 &= \mu[\eta_x{}^2 + (4/3)\eta_y{}^2], \quad \alpha_4 = \gamma\mu Pr^{-1}(\eta_x{}^2 + \eta_y{}^2). \end{aligned}$$

Its derivation is made more complicated by virtue of the fact that $\widehat{S}$ includes within it derivatives of $\widehat{Q}$. Therefore the term $\widehat{M}^n \Delta \widehat{Q}^n$ in (4.103) also must contain derivatives of $\Delta \widehat{Q}^n$, so this term is not a simple matrix-vector product as is the case for the terms $\widehat{A}^n \Delta \widehat{Q}^n$ and $\widehat{B}^n \Delta \widehat{Q}^n$.

To clarify this, let us derive the second element in the second row of $\widehat{M}$. We begin by writing the second element of $\widehat{S}$ in terms of $\widehat{Q}$ as follows:

$$\begin{aligned} \widehat{s}_2 &= \frac{\alpha_1}{J} u_\eta + \frac{\alpha_2}{J} v_\eta \\ &= \frac{\alpha_1}{J} \frac{\partial}{\partial \eta} \left(\frac{\widehat{q}_2}{\widehat{q}_1} \right) + \frac{\alpha_2}{J} \frac{\partial}{\partial \eta} \left(\frac{\widehat{q}_3}{\widehat{q}_1} \right), \end{aligned} \tag{4.110}$$

where α_1 and α_2 are defined below (4.109). For this derivation we retain the analytical derivative from the original PDE rather than the finite-difference approximation, which can be applied later. It is clear that the second term on the right-hand side in (4.110), which does not involve $\widehat{q}_2$, will not enter into the term $\widehat{m}_{22}$ in $\widehat{M}$. Hence we define an operator $f(\widehat{q}_2)$ as follows:

$$f(\widehat{q}_2) = \frac{\alpha_1}{J} \frac{\partial}{\partial \eta} \left(\frac{\widehat{q}_2}{\widehat{q}_1} \right), \tag{4.111}$$

which is the first term in (4.110). We can then use a Fréchet derivative to find

$$\begin{aligned} \frac{\partial f}{\partial \widehat{q}_2} \Delta \widehat{q}_2 &= \lim_{\epsilon \to 0} \frac{f(\widehat{q}_2 + \epsilon \Delta \widehat{q}_2) - f(\widehat{q}_2)}{\epsilon} \\ &= \lim_{\epsilon \to 0} \left[\frac{\alpha_1}{J} \frac{\partial}{\partial \eta} \left(\frac{\widehat{q}_2 + \epsilon \Delta \widehat{q}_2}{\widehat{q}_1} \right) - \frac{\alpha_1}{J} \frac{\partial}{\partial \eta} \left(\frac{\widehat{q}_2}{\widehat{q}_1} \right) \right] / \epsilon \\ &= \lim_{\epsilon \to 0} \left[\frac{\alpha_1}{J} \frac{\partial}{\partial \eta} \left(\frac{\epsilon \Delta \widehat{q}_2}{\widehat{q}_1} \right) \right] / \epsilon \\ &= \frac{\alpha_1}{J} \frac{\partial}{\partial \eta} \left(\frac{\Delta \widehat{q}_2}{\widehat{q}_1} \right). \end{aligned} \tag{4.112}$$

Thus we see that the product $\widehat{m}_{22} \Delta \widehat{q}_2$ is

$$\widehat{m}_{22} \Delta \widehat{q}_2 = J^{-1} \alpha_1 \frac{\partial}{\partial \eta} \left(\frac{J}{\rho} \Delta \widehat{q}_2 \right). \tag{4.113}$$

This is identical to (4.109) and clarifies the precise meaning of that equation. The ∂_η derivatives in $\widehat{M}$ operate on the product of the term shown in $\widehat{M}$, e.g. ρ^{-1} in $\widehat{m}_{22}$, the J term shown to the right of the matrix in (4.109), and the appropriate component of $\Delta \widehat{Q}$.

The nonlinear artificial dissipation terms D_ξ and D_η appearing in (4.102) must also be locally linearized. As a result of the complexity of (4.79), for example, an inexact linearization of these terms is often used, especially in the context of

the approximate factorization algorithm. This is achieved by treating the coefficient terms in the artificial dissipation, such as $\epsilon^{(4)}|\widehat{A}|$ in (4.79), as frozen at time level n, making the linearization straightforward. This approximation is not made on the right-hand side.

Substituting the local time linearizations of the nonlinear flux vectors in (4.103) into (4.102) and grouping the $\Delta\widehat{Q}^n$ terms on the left-hand side produces the *delta form* of the algorithm:

$$\begin{aligned}\left[I + h\delta_\xi \widehat{A}^n - hL_\xi + h\delta_\eta \widehat{B}^n - hL_\eta - Re^{-1}h\,\delta_\eta \widehat{M}\right]\Delta\widehat{Q}^n \\ = -h\left(\delta_\xi \widehat{E}^n - D_\xi^n + \delta_\eta \widehat{F}^n - D_\eta^n - Re^{-1}\delta_\eta \widehat{S}^n\right),\end{aligned} \tag{4.114}$$

where L_ξ and L_η result from the linearization of the artificial dissipation terms. The right-hand side is simply h times the right-hand side of the thin-layer form of (4.94). This results in an important property of the delta form. If a fully converged steady solution of (4.114) is obtained, then it will be the correct steady solution of (4.94), *independent of the left-hand side of* (4.114). This means that approximations made to the left-hand side in order to reduce the computational work needed to converge to steady state, i.e. to drive the norm of $\mathbf{R}(\widehat{\mathbf{Q}})$ to machine zero, will have no effect on the converged solution.

The finite-difference operators on the left-hand side of (4.114) operate on the product of the terms immediately to their right within the square brackets and the $\Delta\widehat{Q}^n$ outside the square brackets. For example, the δ_ξ term results in

$$\frac{1}{2}h(\widehat{A}^n_{j+1,k}\Delta\widehat{Q}^n_{j+1,k} - \widehat{A}^n_{j-1,k}\Delta\widehat{Q}^n_{j-1,k}). \tag{4.115}$$

The viscous contribution on the left-hand side includes both the δ_η term shown in (4.114) and the finite-difference approximations of the partial derivatives with respect to η within the viscous flux Jacobian $\widehat{M}$. These must be consistent with the compact three-point operator used on the right-hand side given in (4.48). The $\Delta\widehat{Q}^n$ terms are of course unknown, and (4.114) represents a linear system of equations to be solved at each iteration of the implicit Euler method. Excluding the I term, the terms within the square brackets on the left-hand side of (4.114) are a linearization of the negative discrete residual operator, i.e. the negative of the right-hand side. Consequently, if the I term is omitted, we obtain the Newton method, consistent with the fact that the Newton method is obtained from the time linearized implicit Euler method in the limit as h goes to infinity (see Sect. 2.6.3).

4.5.3 Matrix Form of the Unfactored Algorithm

We refer to (4.114) as the unfactored algorithm. It produces a large banded system of algebraic equations. We now examine the associated matrix. Let the number of grid nodes in the ξ direction be J and in the η direction K. Temporarily ignoring the

viscous and artificial dissipation terms, the banded matrix is a $(J \cdot K \cdot 4) \times (J \cdot K \cdot 4)$ square matrix of the form

$$\left[I + h\delta_\xi \widehat{A}^n + h\delta_\eta \widehat{B}^n\right] \Rightarrow$$

$$\begin{bmatrix}
I & h\widehat{A}/2 & & & h\widehat{B}/2 & & & & & & & \\
-h\widehat{A}/2 & I & h\widehat{A}/2 & & & h\widehat{B}/2 & & & & & & \\
 & -h\widehat{A}/2 & I & h\widehat{A}/2 & & & h\widehat{B}/2 & & & & & \\
 & & -h\widehat{A}/2 & I & h\widehat{A}/2 & & & h\widehat{B}/2 & & & & \\
-h\widehat{B}/2 & & & -h\widehat{A}/2 & I & h\widehat{A}/2 & & & h\widehat{B}/2 & & & \\
 & \ddots & & & \ddots & \ddots & \ddots & & & \ddots & & \\
 & & -h\widehat{B}/2 & & & -h\widehat{A}/2 & I & h\widehat{A}/2 & & & h\widehat{B}/2 & \\
 & & & -h\widehat{B}/2 & & & -h\widehat{A}/2 & I & h\widehat{A}/2 & & & h\widehat{B}/2 \\
 & & & & -h\widehat{B}/2 & & & -h\widehat{A}/2 & I & h\widehat{A}/2 & & \\
 & & & & & \ddots & & & \ddots & \ddots & \ddots & \\
 & & & & & & -h\widehat{B}/2 & & & -h\widehat{A}/2 & I & h\widehat{A}/2 \\
 & & & & & & & -h\widehat{B}/2 & & & -h\widehat{A}/2 & I
\end{bmatrix}, \quad (4.116)$$

where the variables have been ordered with j running first and then k. Each entry is a 4×4 block. If we order the variables with k running first and then j, the roles of $\widehat{A}$ and $\widehat{B}$ are reversed in the above matrix, i.e. the $h\widehat{B}$ terms produce a tridiagonal form, while the $h\widehat{A}$ terms produce a much larger bandwidth. The thin-layer viscous terms involve a three-point operator in the η direction, so they add to the diagonal block and contribute to the $h\widehat{B}$ blocks shown in (4.116), but they do not alter the overall structure of the matrix. Finally, the artificial dissipation terms involve a five-point operator in each direction and thus further increase the matrix bandwidth. If the scalar artificial dissipation model is used, the corresponding entries are in the form σI, where σ is a scalar, and I is the 4×4 identity matrix.

Although this matrix is sparse, it would be very expensive computationally to solve the algebraic system directly through an LU factorization. For example, for an accurate computation of a three-dimensional transonic flow past a wing, one can easily require over ten million mesh nodes. The resulting linear system is a 50 million $\times$ 50 million matrix problem to be solved, and although one could take advantage of its banded sparse structure, it would still be very costly in terms of both computational work and memory. This motivates iterative and approximate solution strategies for sparse linear systems, such as the approximate factorization algorithm described next.

4.5.4 Approximate Factorization

One way to reduce the computational cost of the solution process is to introduce an approximate factorization of the two-dimensional operator into two one-dimensional operators. Ignoring the artificial dissipation for now, the left-hand side of (4.114) can be written as

$$\begin{aligned}&\left[I + h\delta_\xi \widehat{A}^n + h\delta_\eta \widehat{B}^n - hRe^{-1}\delta_\eta \widehat{M}^n\right] \Delta\widehat{Q}^n \\ &= \left[I + h\delta_\xi \widehat{A}^n\right] \left[I + h\delta_\eta \widehat{B}^n - hRe^{-1}\delta_\eta \widehat{M}^n\right] \Delta\widehat{Q}^n \\ &\quad -h^2\delta_\xi \widehat{A}^n \delta_\eta \widehat{B}^n \Delta\widehat{Q}^n + h^2 Re^{-1}\delta_\xi \widehat{A}^n \delta_\eta \widehat{M}^n \Delta\widehat{Q}^n. \end{aligned} \tag{4.117}$$

Noting that $\Delta\widehat{Q}^n$ is $O(h)$, the difference between the factored form and the unfactored form is $O(h^3)$. Therefore, this difference can be neglected without reducing the time accuracy below second order.

The resulting factored form of the algorithm is

$$\begin{aligned}&\left[I + h\delta_\xi \widehat{A}^n\right]\left[I + h\delta_\eta \widehat{B}^n - hRe^{-1}\delta_\eta \widehat{M}^n\right] \Delta\widehat{Q}^n \\ &= -h\left[\delta_\xi \widehat{E}^n + \delta_\eta \widehat{F}^n - Re^{-1}\delta_\eta \widehat{S}^n\right]. \end{aligned} \tag{4.118}$$

We now have two matrices each of which is block tridiagonal if the appropriate ordering of the variables is used. The structure of the block tridiagonal matrices is

$$\left[I + h\delta_\xi \widehat{A}^n\right] \Rightarrow \begin{bmatrix} I & h\widehat{A}/2 & & & & & & & \\ -h\widehat{A}/2 & I & h\widehat{A}/2 & & & & & & \\ & -h\widehat{A}/2 & I & h\widehat{A}/2 & & & & & \\ & & \ddots & \ddots & \ddots & & & & \\ & & & -h\widehat{A}/2 & I & h\widehat{A}/2 & & & \\ & & & & -h\widehat{A}/2 & I & h\widehat{A}/2 & & \\ & & & & & -h\widehat{A}/2 & I & h\widehat{A}/2 & \\ & & & & & & -h\widehat{A}/2 & I \end{bmatrix}.$$

The thin-layer viscous term $\widehat{M}$ is kept with the η factor. Since it is also based upon a three-point stencil, it will not affect the tridiagonal structure.

The mechanics of the approximate factorization algorithm are as follows. First solve the system

$$\left[I + h\delta_\xi \widehat{A}^n\right] \Delta\tilde{Q} = -h\left[\delta_\xi \widehat{E}^n + \delta_\eta \widehat{F}^n - Re^{-1}\delta_\eta \widehat{S}^n\right] \tag{4.119}$$

for $\Delta\tilde{Q}$, where $\Delta\tilde{Q}$ is an intermediate variable. This requires K solutions of a $(J\cdot 4)\times(J\cdot 4)$ system. With the variables ordered with j running first, followed by k, this is a block tridiagonal system, which can be efficiently solved by a block lower-upper (LU) decomposition. This step is equivalent to solving K one-dimensional problems, one for each ξ line in the mesh.

The next step is to permute, or reorder, $\Delta\tilde{Q}$ such that k is running first, followed by j. This reordering is only conceptual. In practice, this is handled through programming using array indices. Then solve

$$\left[I + h\delta_\eta \widehat{B}^n - hRe^{-1}\delta_\eta \widehat{M}^n\right] \Delta\widehat{Q}^n = \Delta\tilde{Q} \tag{4.120}$$

for $\Delta\widehat{Q}^n$. This requires J solutions of a $(K \cdot 4) \times (K \cdot 4)$ system. With the variables ordered with k running first, followed by j, this is also a block tridiagonal system. This step is equivalent to solving J one-dimensional problems, one for each η line in the mesh. The resulting vector $\Delta\widehat{Q}^n$ must be reordered back to the original database with j running first, again only conceptually, and added to $\widehat{Q}^n$ to form $\widehat{Q}^{n+1}$.

Since efficient specialized algorithms can be used to solve block tridiagonal systems, the factored form substantially reduces the computational work required for one implicit time step. Moreover, as a result of the use of the delta form, we are assured that the steady-state solution is unaffected by the factorization of the left-hand side operator. What remains to be seen is the effect of the factorization on the number of iterations needed to converge to the steady state. This we examine next.

For this purpose we will consider the following simple scalar model ODE:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \big[\lambda_x + \lambda_y\big]u + a, \tag{4.121}$$

where λ_x, λ_y, and a are complex constants, which has the exact solution

$$u(t) = c\mathrm{e}^{(\lambda_x+\lambda_y)t} - \frac{a}{\lambda_x + \lambda_y}. \tag{4.122}$$

We will assume that both λ_x and λ_y have negative real parts, so the ODE is inherently stable and has a steady solution given by

$$\lim_{t\to\infty} u(t) = -\frac{a}{\lambda_x + \lambda_y}. \tag{4.123}$$

Following the approach of Sect. 2.6.2, application of the unfactored form of the implicit Euler method leads to an OΔE that has the following solution:

$$u_n = c\sigma^n - \frac{a}{\lambda_x + \lambda_y}, \tag{4.124}$$

where

$$\sigma = \frac{1}{1 - h\,\lambda_x - h\,\lambda_y}.$$

This method is unconditionally stable and converges rapidly to the steady-state solution for large h because the magnitude of the amplification factor $|\sigma| \to 0$ as $h \to \infty$. As discussed, however, when applied to practical problems, the cost of this method can be prohibitive.

In contrast, the approximate factorization presented in this chapter produces the following OΔE when applied to (4.121):

$$(1 - h\,\lambda_x)\big(1 - h\,\lambda_y\big)(u_{n+1} - u_n) = h\big(\lambda_x u_n + \lambda_y u_n + a\big),$$

which reduces to

$$(1 - h\,\lambda_x)\big(1 - h\,\lambda_y\big)u_{n+1} = \Big(1 + h^2\lambda_x\lambda_y\Big)u_n + ha.$$

The solution of this OΔE is given by (4.124) with

$$\sigma = \frac{1 + h^2\lambda_x\lambda_y}{(1 - h\,\lambda_x)\big(1 - h\,\lambda_y\big)}. \tag{4.125}$$

Although this method remains unconditionally stable and produces the exact steady-state solution independent of h, it converges very slowly to the steady-state solution for large values of h, since the magnitude of the amplification factor $|\sigma| \to 1$ as $h \to \infty$. The factoring error has introduced an h^2 term in the numerator of the amplification factor that destroys the good convergence characteristics at large time steps. In comparison with the unfactored method, the factored form will take more iterations to converge, but each iteration will involve much less computational work.

Let us examine this in more detail. The amplification factor approaches unity as h goes to zero, and, for the factored form, its magnitude also tends to unity as h goes to infinity. The magnitude of the amplification factor thus has a minimum for some value of h, and this is the optimum choice of h for rapid convergence to steady-state. When solving a system of ODEs, there are many eigenvalues, and one cannot choose the optimum value of h for each one. Instead, one seeks an h that balances the magnitude of the amplification factor associated with the smallest eigenvalues with that associated with the largest eigenvalues. Choosing a smaller h will increase the amplification factor for the smallest eigenvalue, while a larger h will increase the amplification factor for the largest eigenvalue. Hence this choice of h is optimal in the sense that it minimizes the maximum amplification factor.

One can contrast this with the time step choice for an explicit time-marching scheme applied to a steady problem. Such schemes are conditionally stable, so there is firm upper bound on the time step. Optimal convergence to steady state is usually achieved with a time step just slightly below this stability limit. In other words, h must be chosen such that the largest eigenvalues lie in the stable region of the explicit method, which is generally a smaller time step than would be optimal for the factored implicit method. Therefore, the amplification factor for the smallest eigenvalues will be larger than for the factored method, and a larger number of iterations will be needed to reach a steady state. This must of course be weighed against the reduced cost per time step of the explicit method. As the spread in the eigenvalues increases, i.e. the problem becomes stiffer, the advantage tilts toward the implicit method. For example, implicit methods are typically preferred for problems involving chemical reactions or grid cells with very high aspect ratios as needed for the computation of turbulent flows at high Reynolds numbers.

Now we return to the contribution of the linearization of the artificial dissipation terms to the left-hand side of (4.114). The first operator, L_ξ, operates solely in the ξ direction, while the second, L_η, operates solely in the η direction. Hence these oper-

ators are amenable to approximate factorization with hL_ξ added to the $\left[I + h\delta_\xi \widehat{A}^n\right]$ factor and hL_η to the $\left[I + h\delta_\eta \widehat{B}^n - hRe^{-1}\delta_\eta \widehat{M}^n\right]$ factor. Since the artificial dissipation operators involve a five-point stencil, the matrices become block pentadiagonal rather than block tridiagonal.

4.5.5 Diagonal Form of the Implicit Algorithm

The approximate factorization algorithm based on solving block pentadiagonal factors is a viable and efficient algorithm. Nevertheless, the majority of the computational work resides in solving the block pentadiagonal systems, so it is worthwhile to examine strategies to reduce this. One way to reduce the computational work is to introduce a diagonalization of the blocks in the implicit operators, as developed by Pulliam and Chaussee [5]. The eigensystems of the flux Jacobians $\widehat{A}$ and $\widehat{B}$ are used in this construction. For now let us restrict ourselves to the Euler equations; application to the Navier-Stokes equations is discussed later.

The flux Jacobians $\widehat{A}$ and $\widehat{B}$ each have real eigenvalues and a complete set of eigenvectors. Therefore, the Jacobian matrices can be diagonalized as follows (see Warming et al. [16]):

$$\Lambda_\xi = T_\xi^{-1}\widehat{A}T_\xi \quad \text{and} \quad \Lambda_\eta = T_\eta^{-1}\widehat{B}T_\eta, \tag{4.126}$$

where Λ_ξ and Λ_η are diagonal matrices containing the eigenvalues of $\widehat{A}$ and $\widehat{B}$, T_ξ is a matrix whose columns are the eigenvectors of $\widehat{A}$, and T_η is the corresponding eigenvector matrix for $\widehat{B}$. These matrices are written out in the Appendix. We take the factored algorithm in delta form (4.118), neglect the viscous terms, and replace $\widehat{A}$ and $\widehat{B}$ with their respective eigensystem decompositions to obtain:

$$\begin{aligned}&\left[T_\xi T_\xi^{-1} + h\,\delta_\xi\left(T_\xi \Lambda_\xi T_\xi^{-1}\right)\right]\left[T_\eta T_\eta^{-1} + h\,\delta_\eta\left(T_\eta \Lambda_\eta T_\eta^{-1}\right)\right]\Delta\widehat{Q}^n \\ &\quad = -h\left[\delta_\xi \widehat{E}^n + \delta_\eta \widehat{F}^n\right] = \widehat{R}^n.\end{aligned} \tag{4.127}$$

Note that the identity matrix I has been replaced by $T_\xi T_\xi^{-1}$ and $T_\eta T_\eta^{-1}$ in each factor, respectively.

At this point, no approximations have been made, and with the exception of the viscous terms, (4.118) and (4.127) are equivalent. A modified form of (4.127) can be obtained by factoring the T_ξ and T_η eigenvector matrices outside the spatial derivative terms δ_ξ and δ_η. The eigenvector matrices are functions of ξ and η, and therefore this modification introduces an approximation on the left-hand side. The resulting equations are

$$T_\xi\left[I + h\,\delta_\xi\,\Lambda_\xi\right]\widehat{N}\left[I + h\,\delta_\eta\,\Lambda_\eta\right]T_\eta^{-1}\Delta\widehat{Q}^n = \widehat{R}^n, \tag{4.128}$$

where $\widehat{N} = T_\xi^{-1}T_\eta$ (see Appendix).

The approximation made to the left-hand side of (4.127) reduces the time accuracy to at best first order, and, moreover, gives time-accurate computations a nonconservative feature that leads to errors in shock speeds and jump conditions. However, the right-hand side is unmodified, so if the algorithm converges, it will converge to the correct steady-state solution. The advantage of the diagonal form is that the equations are decoupled as a result. Rather than a block tridiagonal system, we now have four scalar tridiagonal systems plus some additional 4×4 matrix-vector multiplies, leading to a substantial reduction in computational work. The computational work can be further decreased by exploiting the fact that the first two eigenvalues of the system are identical (see Appendix). This allows us to combine the coefficient calculations and part of the inversion work for the first two scalar operators.

The diagonal form reduces the computational work per time step and produces the correct steady solution. The next step is to examine its effect on the number of time steps needed to converge to steady state. Normally one would turn to linear stability analysis to assess the stability limits and convergence rate of an algorithm. However, linear analysis is of no use in analyzing the diagonal algorithm because the assumption of linear analysis is that the Jacobians are constant. With this assumption, the diagonalization introduces no approximation at all, so linear stability analysis predicts the diagonal algorithm to have the same unconditional stability as the original block algorithm. Therefore one must resort to computational experiments in order to investigate the impact of the diagonal form on the convergence properties of the diagonal algorithm. Pulliam and Chaussee [5] have shown that the convergence and stability limits of the diagonal algorithm are similar to those of the block form of the algorithm. The reader will have the opportunity to perform similar experiments as part of the exercises at the end of this chapter.

The steps involved in applying the diagonal form of the approximate factorization algorithm are as follows:

1. Beginning with (4.128), premultiply $\widehat{R}^n$ by T_ξ^{-1} to obtain the system

$$\left[I + h\,\delta_\xi\,\Lambda_\xi\right]\,\widehat{N}\,\left[I + h\,\delta_\eta\,\Lambda_\eta\right]\,T_\eta^{-1}\Delta\widehat{Q}^n = T_\xi^{-1}\widehat{R}^n. \tag{4.129}$$

2. With the variables ordered with j running first, solve the scalar tridiagonal system

$$\left[I + h\,\delta_\xi\,\Lambda_\xi\right]X_1 = T_\xi^{-1}\widehat{R}^n \tag{4.130}$$

for the temporary variable X_1. This produces the following

$$\widehat{N}\,\left[I + h\,\delta_\eta\,\Lambda_\eta\right]\,T_\eta^{-1}\Delta\widehat{Q}^n = X_1. \tag{4.131}$$

3. Premultiply by $\widehat{N}^{-1}$ to obtain

$$\left[I + h\,\delta_\eta\,\Lambda_\eta\right]\,T_\eta^{-1}\Delta\widehat{Q}^n = \widehat{N}^{-1}X_1. \tag{4.132}$$

4. With the variables ordered with k running first, solve the scalar tridiagonal system

$$\left[I + h\,\delta_\eta\,\Lambda_\eta\right] X_2 = \widehat{N}^{-1} X_1 \tag{4.133}$$

for X_2, giving

$$T_\eta^{-1}\,\Delta\widehat{Q}^n = X_2. \tag{4.134}$$

5. Premultiply X_2 by T_η to find $\Delta\widehat{Q}^n$.

The diagonal algorithm as presented above is only strictly valid for the Euler equations. This is because we have neglected the implicit linearization of the viscous flux $\widehat{S}^n$ in the implicit operator for the η direction. The viscous flux Jacobian $\widehat{M}^n$ is not simultaneously diagonalizable with the inviscid flux Jacobian $\widehat{B}^n$ and therefore to include it in the diagonal form is not straightforward. For viscous flows one can consider four options. One possibility is to use the diagonal form in the ξ direction only and the block algorithm in the η direction. This increases the computational work substantially. Another option is to introduce a third factor to the implicit side of Eq. 4.118 as follows:

$$\left[I - hRe^{-1}\delta_\eta \widehat{M}^n\right]. \tag{4.135}$$

This again increases the computational work since we now have an added block tridiagonal inversion. One could diagonalize this term, but it would nevertheless increase the cost substantially. The third option is to throw caution to the wind and actually neglect the viscous Jacobian, thereby gaining the increased efficiency of the diagonal algorithm. This can have an adverse effect on stability and convergence. The fourth option is to include a diagonal term on the implicit side that is a rough approximation to the viscous Jacobian spectral radius. Estimates that have been used successfully are

$$\begin{aligned}\lambda_v(\xi) &= \gamma Pr^{-1}\mu Re^{-1}\left(\xi_x^2 + \xi_y^2\right)\rho^{-1}\\ \lambda_v(\eta) &= \gamma Pr^{-1}\mu Re^{-1}\left(\eta_x^2 + \eta_y^2\right)\rho^{-1},\end{aligned} \tag{4.136}$$

which are added to the appropriate operators in Eq. 4.128 with a differencing stencil taken from Eq. 4.48. With these terms added, the diagonal algorithm is given as

$$T_\xi\left[I + h\,\delta_\xi\,\Lambda_\xi - h\,I\,\delta_{\xi\xi}\lambda_v(\xi)\right]\widehat{N}\left[I + h\,\delta_\eta\,\Lambda_\eta - h\,I\,\delta_{\eta\eta}\lambda_v(\eta)\right]T_\eta^{-1}\,\Delta\widehat{Q}^n = \widehat{R}^n. \tag{4.137}$$

The ξ term is not added if the thin layer approximation is used. Although this approach is not rigorous, given that the eigenvectors of the viscous Jacobians are distinct from those of the inviscid Jacobians, it has proven to be effective in terms of both efficiency

and reliability. It is thus the recommended approach for application of the diagonal form to viscous flows.

Next we consider the contribution of the linearization of the artificial dissipation terms, L_ξ and L_η in (4.114), in the context of the diagonal algorithm. Recall that the operator associated with the fourth-difference dissipation leads to a pentadiagonal matrix rather than a tridiagonal matrix, so the full block algorithm requires the solution of block pentadiagonal systems. If scalar dissipation is used, the contributions to the left-hand side are in the form σI, where σ is a scalar, so this is directly compatible with the diagonal form. With matrix dissipation, the diagonalization is also straightforward, since $\widehat{A}$ and $|\widehat{A}|$ share the same eigenvectors, and so do $\widehat{B}$ and $|\widehat{B}|$. With the linearization of the artificial dissipation included on the left-hand side, the diagonal form requires the solution of scalar pentadiagonal rather than block pentadiagonal systems, which results in a significant saving in computational work for the solution of steady flows.

The diagonal algorithm is an efficient and robust algorithm. However, there are some cases with specific properties for which it will not converge; in such cases, the block pentadiagonal algorithm is more reliable. An intermediate block form in which block tridiagonal systems are solved has also received considerable use. In this intermediate approach, the contribution of the fourth-difference dissipation on the left-hand side is approximated by a second-difference dissipation term with a coefficient equal to twice the coefficient of the fourth-difference dissipation on the right-hand side. It can be shown using linear theory that this approximation remains unconditionally stable. Such an algorithm will typically converge much more slowly than a full pentadiagonal linearization, but it has a lower cost per time step than the block pentadiagonal algorithm and can be more robust than the scalar pentadiagonal algorithm in some cases.

4.5.6 Convergence Acceleration for Steady Flow Computations

Local Time Stepping. As discussed in Sect. 4.5.4, the approximate factorization leads to an amplification factor σ that approaches unity as the time step h tends to infinity. Consequently, there is an optimum time step that minimizes the maximum magnitude of σ for the various eigenvalues associated with the Jacobian of the discrete spatial operator and hence produces the fastest possible convergence to steady state. For the inviscid flux terms, the eigenvalues of the Jacobian of the discrete residual vector are proportional to the characteristic speeds, e.g. u, $u + a$, $u - a$ in one dimension, divided by a characteristic mesh spacing, e.g. Δx in one dimension. The amplification factor σ is a function of the product of the eigenvalues and the time step h. Hence the convergence rate is dependent on the Courant (or CFL) number, given in one dimension by

$$C_{\mathrm{n}} = \frac{(|u| + a)h}{\Delta x}. \tag{4.138}$$

Here we have defined the Courant number based on the largest characteristic speed, $|u| + a$, but waves propagating at the other characteristic speeds will have a different effective Courant number.

Both the characteristic speeds and the mesh spacing can vary widely within a mesh. With a constant h, the local Courant number associated with each mesh node will thus also vary widely and will be suboptimal. When computing steady flows, we have the freedom to vary the time step locally in space. This destroys time accuracy but has no effect on the converged steady-state solution. Local time stepping can have a substantial influence on the convergence rate of a factored algorithm. It can be viewed as a way to condition the iteration matrix of the iterative methods defined via (4.118) or (4.128), or it can be interpreted as an attempt to use a more uniform (and hence closer to optimal) Courant number throughout the flow field. In any event, local time stepping can be effective for grid spacings that vary from very fine to very coarse—a situation usually encountered in simulations that contain a wide variety of length scales.

As a rule, one wishes to adjust the local time step at each grid node in proportion to the local grid spacing divided by the local characteristic speed of the flow, leading to a constant Courant number. In multiple dimensions, the situation is not quite so straightforward. For example, a cell with a high aspect ratio has two distinct grid spacings. In two dimensions, an approximation to a constant Courant number is achieved by the following formula for the local time step:

$$\Delta t = \frac{\Delta t_{\text{ref}}}{|U| + |V| + a\sqrt{\xi_x^2 + \xi_y^2 + \eta_x^2 + \eta_y^2}}, \tag{4.139}$$

where Δt_{ref} is defined by the user and must be chosen through experimentation to provide fast convergence.

For highly stretched grids, the grid spacing can vary by over six orders of magnitude. The variation in the characteristic speeds is generally more moderate. Therefore, the grid spacing is the more important parameter for maintaining a reasonably uniform Courant number, and a purely geometric variation of Δt can be effective. The following geometric formula for the local time step produces fast convergence when used with the approximately factored algorithm [2]:

$$\Delta t = \frac{\Delta t|_{\text{ref}}}{1 + \sqrt{J}}. \tag{4.140}$$

The term J^{-1} is closely related to the cell area. Therefore, this formula produces a Δt that is roughly proportional to the square root of the cell area. The addition of unity to the denominator prevents Δt from becoming too large at the largest grid cells.

To illustrate the advantage of using a variable time step, Fig. 4.5 shows the improvement in convergence rate when a variable time step based on (4.140) is substituted for a constant time step in a NACA 0012 airfoil test case where the Euler

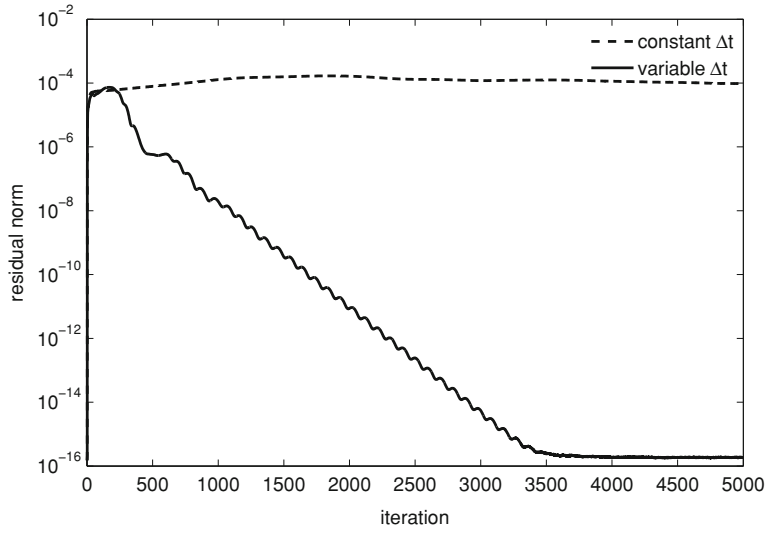

Fig. 4.5 Ċonvergence improvement due to local time stepping

equations are solved at a Mach number of 0.8 and an angle of attack of 1.25 degrees. The constant time step chosen is the largest stable constant time step. For this comparison all other parameters were held constant.

In the above discussion, we have considered only the local Courant number, which is related to the inviscid fluxes. For an implicit algorithm, determination of the local time step based on inviscid considerations is generally sufficient for high Reynolds number flows, as these are convection dominated. For flows at low Reynolds numbers, consideration also needs to be given to the local Von Neumann number (see Sect. 2.7.4). As we will see in Chap. 5, local time stepping is even more critical for explicit methods.

Mesh Sequencing. The mesh density is based on accuracy considerations. A sufficiently fine mesh must be used such that the numerical errors from the spatial discretization lie below a desired threshold. The iterative methods given by (4.118) or (4.128) require an initial solution to begin the process. Fewer iterations are needed to converge to steady state if the initial solution is not too far from the converged solution, which is of course unknown at the outset. It is common to initiate the iterations with a solution given by a uniform flow that satisfies some free-stream or inflow boundary conditions. This provides a relatively poor initial guess that is much different from the eventual steady solution. Therefore, one way to improve convergence is to begin the iterations using a much coarser mesh than that dictated by the accuracy requirements. On a coarse mesh, the iterations will converge with relatively little computational work to a solution that provides a much improved initial guess for the fine mesh iterations. The solution obtained after reducing the norm of the residual on the coarse mesh by a few orders of magnitude can be interpolated onto

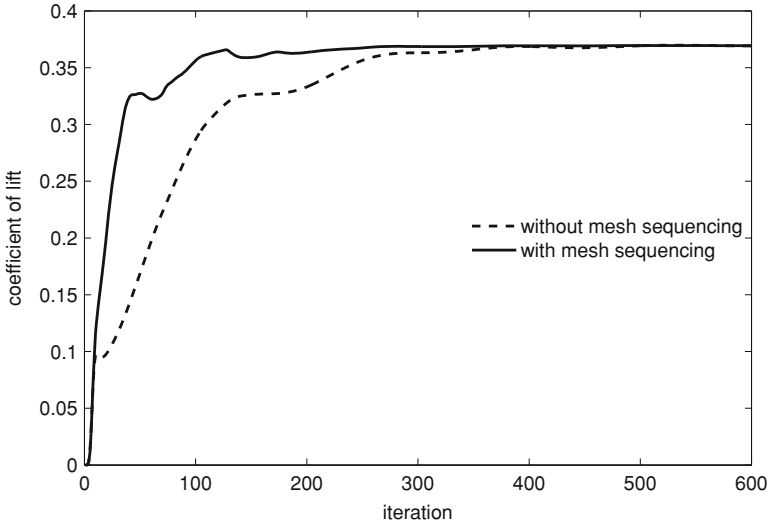

Fig. 4.6 İmprovement in convergence of lift coefficient due to mesh sequencing

the finer mesh to provide the initial iterate for the iterations on the fine mesh. This process can be repeated on a sequence of meshes, beginning with a very coarse mesh and ending on the fine mesh dictated by accuracy requirements. The use of mesh sequencing in this manner can also improve the robustness of a solver, as the coarse meshes are effective at damping initial transients, when nonlinear effects are large.

Figure 4.6 shows an example of the improvement in convergence resulting from mesh sequencing. For an inviscid flow over the NACA 0012 airfoil at a Mach number of 0.8 and an angle of attack of 1.25 degrees, a sequence of four C-meshes has been used. The first mesh is 32 by 17, the second 63 by 33, the third 125 by 69, and the final mesh has 249 by 98 nodes. Both cases were started with a free-stream initial condition.

4.5.7 Dual Time Stepping for Unsteady Flow Computations

The implicit algorithm described above is suitable for time-accurate computations of unsteady flows where the equations are integrated through time from some meaningful initial condition. A sufficiently fine mesh is needed to ensure that spatial discretization errors are small; in addition, the time step must be selected such that the temporal discretization errors are also below the desired threshold. Generally speaking, at least second-order temporal accuracy is desired. The local time linearization and approximate factorization preserve the order of accuracy of a second-order

implicit time-marching method, such as the second-order backward and trapezoidal methods discussed earlier. Neither the diagonal form nor local time stepping should be used for time-accurate computations of unsteady flows.

The second-order backwards time-marching method is given by

$$u_{n+1} = \frac{1}{3}[4u_n - u_{n-1} + 2hu'_{n+1}]. \tag{4.141}$$

Applying this method to the thin-layer form of (4.95) gives

$$\begin{aligned}\widehat{Q}^{n+1} = {} & \frac{4}{3}\widehat{Q}^n - \frac{1}{3}\widehat{Q}^{n-1} \\ & + \frac{2h}{3}\left(-\delta_\xi \widehat{E}^{n+1} + D_\xi^{n+1} - \delta_\eta \widehat{F}^{n+1} + D_\eta^{n+1} + Re^{-1}\delta_\eta \widehat{S}^{n+1}\right).\end{aligned} \tag{4.142}$$

After local time linearization and approximate factorization, a form analogous to (4.118) is obtained:

$$\begin{aligned}&\left[I + \frac{2h}{3}\delta_\xi \widehat{A}^n\right]\left[I + \frac{2h}{3}\delta_\eta \widehat{B}^n - \frac{2h}{3}Re^{-1}\delta_\eta \widehat{M}^n\right]\Delta\widehat{Q}^n \\ &\quad = \widehat{Q}^n - \widehat{Q}^{n-1} - \frac{2h}{3}\left[\delta_\xi \widehat{E}^n + \delta_\eta \widehat{F}^n - Re^{-1}\delta_\eta \widehat{S}^n\right].\end{aligned} \tag{4.143}$$

The method given by (4.143) is the approximately factored form of the second-order backward time-marching method. It is an efficient second-order implicit method for time-accurate computations of unsteady flows. However, despite the fact that the linearization and factorization errors do not diminish the order of accuracy of the method, they increase the error incurred per time step. This is the motivation for the dual time stepping approach, which eliminates linearization and factorization errors.

In order to demonstrate the dual time stepping approach, we begin by rearranging (4.142) as follows

$$\frac{3\widehat{Q}^{n+1} - 4\widehat{Q}^n + \widehat{Q}^{n-1}}{2h} + R(\widehat{Q}^{n+1}) = 0, \tag{4.144}$$

where

$$R(\widehat{Q}^{n+1}) = \left[\delta_\xi \widehat{E}^{n+1} - D_\xi^{n+1} + \delta_\eta \widehat{F}^{n+1} - D_\eta^{n+1} - Re^{-1}\delta_\eta \widehat{S}^{n+1}\right]. \tag{4.145}$$

This is a nonlinear algebraic equation that must be solved for $\widehat{Q}^{n+1}$ at each time step. To reflect this, we define $R_{\mathrm{u}}(\widehat{Q})$ as

$$R_{\mathrm{u}}(\widehat{Q}) = \frac{3\widehat{Q} - 4\widehat{Q}^n + \widehat{Q}^{n-1}}{2h} + R(\widehat{Q}), \tag{4.146}$$

so the nonlinear equation to be solved is simply

$$R_{\mathrm{u}}(\widehat{Q}) = 0. \tag{4.147}$$

One can readily observe the similarity between the nonlinear equation to be solved at every iteration of the second-order backward time-marching method, $R_{\mathrm{u}}(\widehat{Q}) = 0$, and the equation to be solved for a steady flow, $R(\widehat{Q}) = 0$. Therefore, any method developed for steady problems, such as inexact-Newton methods and implicit or explicit time-marching methods that follow a time-dependent path to steady state, can be used to solve (4.147).

In this chapter, our focus is on the approximate factorization algorithm, which follows a time-dependent, though not necessarily time-accurate, path to the steady solution. In order to enable application of this algorithm to the solution of (4.147), we introduce a pseudo-time variable τ (not to be confused with the variable τ in the generalized curvilinear coordinate transformation) to produce a system of ODEs as follows:

$$\frac{\mathrm{d}\widehat{Q}}{\mathrm{d}\tau} + R_{\mathrm{u}}(\widehat{Q}) = 0. \tag{4.148}$$

In order to solve for the steady-state solution of this ODE, which is the solution to (4.147), we can apply the approximately-factored implicit Euler method. We introduce a pseudo-time index p such that $\widehat{Q}^p = \widehat{Q}(p\Delta\tau)$, where $\Delta\tau = \tau_{p+1} - \tau_p$, to obtain

$$\begin{aligned}\left[I + \frac{\Delta\tau}{b}\delta_\xi \widehat{A}^p\right]\left[I + \frac{\Delta\tau}{b}\delta_\eta \widehat{B}^p - \frac{\Delta\tau}{b}Re^{-1}\delta_\eta \widehat{M}^p\right]\Delta\widehat{Q}^p \\ = -\frac{\Delta\tau}{b}R_{\mathrm{u}}(\widehat{Q}^p),\end{aligned} \tag{4.149}$$

where

$$b = 1 + \frac{3\Delta\tau}{2h},$$

and we have divided by b before factoring. The converged solution obtained from this iterative process provides $\widehat{Q}^{n+1}$. The accuracy of the time-marching method is dictated by the time step h, while the pseudo-time step $\Delta\tau$ can be chosen for fast convergence with no regard for time accuracy, since it has no effect on the converged solution of (4.147). Similarly, for the pseudo-time iterations, the diagonal form and local time stepping can be used to speed up convergence.

Dual time stepping is an example of an approach where an iterative method is used to solve the nonlinear equation that arises at each time step of an implicit method. This approach eliminates linearization and factorization errors and can also simplify the implementation of boundary conditions. It is natural to use a fast steady solver for the solution of this nonlinear equation along with any convergence acceleration

techniques developed for steady flows. One may question the efficiency of an approach where the unsteady problem is in effect solved as a sequence of steady problems. However, it is important to note that the initial iterate for the pseudo-time iterations is $\widehat{Q}^n$, which is a much better estimate of $\widehat{Q}^{n+1}$ then is usually available for steady computations. Hence one can expect the number pseudo-time steps needed to obtain a converged solution to (4.147) to be much less than the number of time steps needed to obtain a converged solution to a steady flow problem.

4.6 Boundary Conditions

There are a number of different ways to implement boundary conditions. Before describing one particular approach, we will introduce the important aspects of boundary condition development that must be considered in selecting an approach, which are as follows:

1. The physical definition of the flow problem must be properly represented. For example, a viscous flow ordinarily requires a no-slip condition at solid surfaces.
2. The physical conditions must be expressed in mathematical form and must be consistent with the mathematical description of the problem. For example, the no-slip condition referred to above must be expressed in terms of the variables selected. Moreover, this condition cannot be applied if inviscid governing equations are chosen.
3. The boundary conditions expressed in mathematical form must be approximated numerically.
4. Depending on the algorithm, the numerical scheme in the interior may require more boundary information than the physics provides. Hence a means must be developed for providing this additional boundary information.
5. The combination of the interior scheme with the boundary scheme must be checked for stability and accuracy. In general, the two should have consistent accuracy.
6. The boundary condition formulation must be assessed in terms of its impact on the efficiency and generality of the solver.

With these considerations in mind, one can approach the development of boundary conditions from several different directions. Moreover, there exist various different boundary types, such as inflow/outflow boundaries, solid walls, symmetry boundaries, and periodic boundaries, one or more of which can be present in a specific flow problem. In this chapter, we will cover an approach to the boundary conditions typically associated with computations of external flows. The basic principles covered are easily extended to other boundary types.

With an implicit solver, one might expect implicit boundary conditions to be a strict requirement. In order to obtain the benefits of an inexact-Newton method, they are certainly recommended. For an approximately-factored solver, however, the optimal time step is not so large that implicit boundary conditions are essential, and

the use of explicit boundary conditions does not typically degrade the convergence rate.

For external flows, one is faced with the problem that the boundary conditions are defined at an infinite distance from the body. Although coordinate transformations can be used to address this, it is much more common to introduce an artificial far-field boundary in order to limit the size of the computational domain. This boundary must be located a sufficient distance from the body that the error introduced does not exceed the desired error threshold. At a far-field boundary, viscous effects are typically negligible and the flow can be considered inviscid. Consequently, a characteristic approach is taken to inflow and outflow boundary conditions at the far-field boundary. Proper application of characteristic theory is essential in order to ensure well-posedness. At a far-field boundary through which a wake is advecting or viscous effects are not negligible, a different approach is used; this is discussed further below.

4.6.1 Characteristic Approach

The concept of characteristic theory is most easily demonstrated with the linearized one-dimensional Euler equations, where

$$\partial_t Q + \partial_x (AQ) = 0 \tag{4.150}$$

represents the model equation. Since A is a constant-coefficient matrix, we can diagonalize (4.150) using the relation $A = X\Lambda_A X^{-1}$, where X is the right eigenvector matrix, and

$$\Lambda_A = \begin{bmatrix} u & 0 & 0 \\ 0 & u+a & 0 \\ 0 & 0 & u-a \end{bmatrix}. \tag{4.151}$$

Premultiplying by X^{-1} and inserting the product XX^{-1} after A, we obtain

$$\partial_t \left(X^{-1} Q \right) + \Lambda_A \partial_x \left(X^{-1} Q \right) = 0. \tag{4.152}$$

Defining $X^{-1}Q = W$, we now have a diagonal system. The equations have been decoupled into three equations in the form of the linear convection equation with the characteristic speeds u, $u + a$, and $u - a$. The associated characteristic variables, or Riemann invariants, for this constant-coefficient linear system are defined by W. One can also obtain these same characteristic speeds and the associated Riemann invariants for the full nonlinear Euler equations without the assumption that A is a constant coefficient matrix.

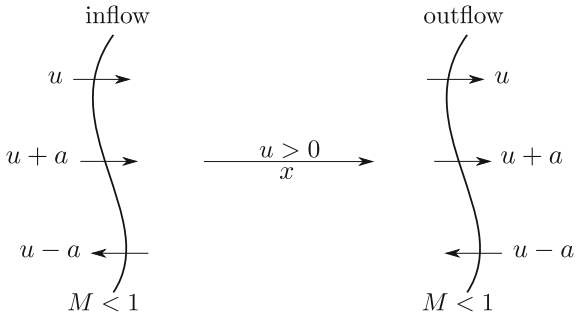

Fig. 4.7 Characteristics at subsonic inflow and outflow boundaries of a closed domain

With the diagonalized form of the equations, the boundary condition requirements are clear. Consider first a subsonic flow. At the left boundary of a closed physical domain, see Fig. 4.7, where $0 < u < a$ (for example, subsonic inflow for a channel flow), the two characteristic speeds $u, u + a$ are positive, while $u - a$ is negative. At inflow then, two pieces of information enter the domain along the two incoming characteristics, and one piece leaves along the outgoing characteristic. At the outflow boundary, one piece of information enters and two leave. Thus we can obtain a well-posed problem by specifying the first two components of W, which are the two incoming characteristic variables, at the inflow boundary and then handling the third characteristic variable such that its value is not constrained, i.e. it is determined by the interior flow. At the outflow boundary, we specify the third component of W and determine the first two from the interior flow. If the flow is supersonic, all characteristic speeds have the same sign. Hence one must specify all variables at inflow and none at outflow.

It is not necessary to specify the characteristic variables; other flow quantities can be used, as long as they lead to well-posed conditions. The major constraint is that the correct number of boundary values corresponding to incoming characteristics must be specified, regardless of the variables that are chosen. Some combinations of variables lead to a well-posed problems, others do not. In the next section, we describe a test to establish whether a given choice of variables is well posed.

4.6.2 Well-Posedness Test

A check on the well posedness of boundary conditions is given by Chakravarthy [17]. Let us consider one-dimensional flow with subsonic inflow and subsonic outflow. Then two variables can be specified at inflow, associated with the first two eigenvalues, and one variable can be specified at outflow, associated with the third eigenvalue. As an example, we test the following specified values: $\rho = \rho_{\text{in}}$, $\rho u = (\rho u)_{\text{in}}$ and $p = p_{\text{out}}$. These can be written as

$$B_{\text{in}}(Q) = \begin{bmatrix} q_1 \\ q_2 \\ 0 \end{bmatrix} = B_{\text{in}}(Q_{\text{in}}), \tag{4.153}$$

$$B_{\text{out}}(Q) = \begin{bmatrix} 0 \\ 0 \\ (\gamma-1)(q_3 - \frac{1}{2}q_2^2/q_1) \end{bmatrix} = B_{\text{out}}(Q_{\text{out}}), \tag{4.154}$$

with $q_1 = \rho,\ q_2 = \rho u,\ q_3 = e$.

Forming the Jacobians $C_{\text{in}} = \partial B_{\text{in}}/\partial Q$, and $C_{\text{out}} = \partial B_{\text{out}}/\partial Q$ we have

$$C_{\text{in}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad C_{\text{out}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ ((\gamma-1)/2)\ u^2 & -(\gamma-1)u & \gamma-1 \end{bmatrix}. \tag{4.155}$$

The left eigenvector matrix X^{-1} for the one-dimensional Euler equations is[3]

$$\begin{bmatrix} 1-\frac{u^2}{2}(\gamma-1)a^{-2} & (\gamma-1)ua^{-2} & -(\gamma-1)a^{-2} \\ \beta[(\gamma-1)\frac{u^2}{2} - ua] & \beta[a-(\gamma-1)u] & \beta(\gamma-1) \\ \beta[(\gamma-1)\frac{u^2}{2} + ua] & -\beta[a+(\gamma-1)u] & \beta(\gamma-1) \end{bmatrix}, \tag{4.156}$$

with $\beta = 1/(\sqrt{2}\rho a)$.

The condition for well-posedness of these example boundary conditions is that $\overline{C}_{\text{in}}^{-1}$ and $\overline{C}_{\text{out}}^{-1}$ exist, where

$$\overline{C}_{\text{in}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \beta[(\gamma-1)\frac{u^2}{2} + ua] & -\beta[a+(\gamma-1)u] & \beta(\gamma-1) \end{bmatrix}, \tag{4.157}$$

and

$$\overline{C}_{\text{out}} = \begin{bmatrix} 1-\frac{u^2}{2}(\gamma-1)a^{-2} & (\gamma-1)ua^{-2} & -(\gamma-1)a^{-2} \\ \beta[(\gamma-1)\frac{u^2}{2} - ua] & \beta[a-(\gamma-1)u] & \beta(\gamma-1) \\ (\gamma-1)\frac{u^2}{2} & -(\gamma-1)u & \gamma-1 \end{bmatrix}. \tag{4.158}$$

These matrices are formed by adjoining the eigenvectors associated with the outgoing characteristics at the boundary in question to the Jacobian matrices of the boundary conditions. The inverses of the above matrices will exist if their determinants are nonzero. For the two boundaries, we have $\det(\overline{C}_{\text{in}}) = \beta(\gamma-1) \neq 0$, and $\det(\overline{C}_{\text{out}}) = \beta(\gamma-1)a \neq 0$. Therefore, this particular choice of boundary conditions is well posed. Other choices for specified boundary values can be similarly checked.

[3] The rows of X^{-1} are the left eigenvectors of A.

4.6.3 Boundary Conditions for External Flows

We shall outline below some of the more commonly used boundary conditions. These will be presented in the context of a body-fitted C mesh, as depicted in Fig. 4.2, and are easily generalized to other mesh topologies. The approach taken is to solve the governing equations only at the interior nodes of the mesh. Therefore, all variables must be given at the boundary by the numerical boundary conditions. Since the physical boundary conditions provide boundary values for only some of the variables, the others must be determined by extrapolation from the interior flow solution. Moreover, the numerical boundary conditions can be implemented either explicitly or implicitly. In an explicit treatment, the boundary values are held fixed during one iteration of the approximate factorization algorithm. They are then updated based on the new $\widehat{Q}$, and the process is repeated. For an implicit implementation, the numerical boundary conditions must be linearized and the appropriate terms included in the left-hand-side operator of the implicit algorithm.

Body Surfaces. At a body surface, tangency must be satisfied for inviscid flow and the no-slip condition for viscous flow. In two-dimensions, body surfaces are usually mapped to η = constant coordinates, as in Fig. 4.2. In this case, as shown in Sect. 4.2.4, the normal component of velocity is given in terms of the metrics of the transformation by

$$V_n = \frac{\eta_x u + \eta_y v}{\sqrt{\eta_x^2 + \eta_y^2}}, \tag{4.159}$$

and the tangential component by

$$V_t = \frac{\eta_y u - \eta_x v}{\sqrt{\eta_x^2 + \eta_y^2}}. \tag{4.160}$$

For inviscid flows, flow tangency is satisfied by setting $V_n = 0$. The tangential velocity V_t is obtained at the body surface through linear extrapolation along the coordinate line approaching the surface, using the interior values of Q at the nodes above the surface. It is preferable to extrapolate Cartesian velocity components and then form the tangential velocity component based on the extrapolated values. The Cartesian velocity components at the surface are found from the following relation obtained by solving (4.159) and (4.160) for u and v:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{\sqrt{\eta_x^2 + \eta_y^2}} \begin{bmatrix} \eta_y & \eta_x \\ -\eta_x & \eta_y \end{bmatrix} \begin{pmatrix} V_t \\ V_n \end{pmatrix}, \tag{4.161}$$

with V_n set to zero, and V_t determined from the extrapolation. For a viscous flow, the no-slip condition gives $u = v = 0$.

For an inviscid flow, flow tangency is the only physical boundary condition. Therefore only one variable can be specified, which is the normal velocity component, and three more variables must be determined from the interior flow solution. The tangential velocity component is extrapolated, as described above. In addition, pressure and density, for example, can be extrapolated. For steady inviscid flows with uniform upstream conditions, the total or stagnation enthalpy ($H = (e + p)/\rho$) is constant, at least in the exact solution. This requirement can be exploited to determine one variable. For example, after u, v, and p are obtained at the surface, the density can be found by requiring that the total enthalpy at the boundary be equal to the free-stream total enthalpy. Once boundary values for u, v, p, and ρ are determined, the corresponding conservative variables are easily found using their definitions along with the equation of state.

For viscous flows, there is an additional boundary condition related to heat transfer that determines the temperature or its gradient normal to the surface. If the wall remains at constant temperature, then this temperature must be specified. More commonly, an adiabatic condition is appropriate. In this case, there is no heat transfer to or from the wall, giving

$$\frac{\partial T}{\partial n} = 0, \tag{4.162}$$

where n is the direction normal to the wall, and the derivative must be approximated numerically using a one-sided difference formula. This condition provides the temperature at the wall. The wall pressure can be determined by extrapolation from the interior; the conservative variables can then be found from the values of u, v, T, and p.

Far-Field Boundaries. The far-field boundary must be located a sufficient distance away from the body that its effect on the computed solution is negligible. This can be determined by experimentation. The basic goal of the boundary conditions at the far-field boundary is to permit disturbances to exit the domain with little or no reflection, as such artificial reflections can pollute the solution in the interior of the domain. For problems where accurate propagation of waves to and through the outer boundary is critical, specialized non-reflecting boundary conditions have been developed (see for example the discussion by Colonius and Lele [18]). For many flow problems, non-reflecting boundary conditions based on the method of characteristics are sufficient; these are described here.

Following the discussion in Sect. 4.6.1, the idea is to specify incoming Riemann invariants and determine outgoing Riemann invariants from the interior solution by extrapolation. For subsonic flows, we describe an extension to two dimensions based on locally one-dimensional Riemann invariants. The relevant velocity component is that normal to the outer boundary V_n. With n pointing outward from the flow domain, a positive V_n defines an outflow boundary, while a negative V_n defines an inflow boundary. As shown in the Appendix, the two-dimensional inviscid flux Jacobians have three distinct eigenvalues, with the eigenvalue corresponding to the convective speed repeated. From the one-dimensional theory, we have three Riemann

invariants, so one more variable is needed in two dimensions that will be associated with the repeated eigenvalue. The velocity component tangential to the boundary can be used for this purpose. Therefore, we have the following characteristic speeds and associated variables:

$$\begin{aligned}
\lambda_1 &= V_n - a, \quad & R_1 &= V_n - 2a/(\gamma - 1) \\
\lambda_2 &= V_n + a, \quad & R_2 &= V_n + 2a/(\gamma - 1) \\
\lambda_3 &= V_n, & R_3 &= S = \ln \frac{p}{\rho^\gamma} \quad \text{(entropy)} \\
\lambda_4 &= V_n, & R_4 &= V_t.
\end{aligned} \tag{4.163}$$

For a subsonic inflow boundary, where $V_n < 0$, the characteristic speeds satisfy the following:

$$\lambda_1 < 0,\ \lambda_2 > 0,\ \lambda_3 < 0,\ \lambda_4 < 0.$$

A negative characteristic speed corresponds to an incoming characteristic; hence the associated variables must be specified based on free-stream values. The variables associated with positive characteristic speeds must be determined from the interior flow. In this case, R_1, R_3, and R_4 must be specified, and R_2 must be extrapolated from the interior. Once these four variables are determined at the boundary, the four conservative variables can be obtained.

For a subsonic outflow boundary, where $V_n > 0$, the eigenvalues satisfy the following:

$$\lambda_1 < 0,\ \lambda_2 > 0,\ \lambda_3 > 0,\ \lambda_4 > 0.$$

Therefore, R_1 must be set to its free-stream value, and R_2, R_3, and R_4 must be extrapolated from the interior.

For supersonic inflow boundaries, all flow variables are specified; for supersonic outflow boundaries, all variables are extrapolated. For a subsonic boundary through which a viscous wake is flowing, all variables are extrapolated (see [19] for a detailed discussion). Special treatments may be needed at interfaces between blocks in multi-block meshes or at wake cuts. See, for example, Osusky and Zingg [20].

Far-Field Circulation Correction. For computations of two-dimensional flows over lifting bodies, the far-field circulation correction reduces the effect of the far-field boundary location. This enables the distance to the far-field boundary to be reduced without compromising accuracy.Far from a lifting airfoil in a subsonic free-stream, the perturbation caused by the airfoil approaches that induced by a point vortex. This can be exploited by adding the perturbation associated with a point vortex to the free-stream values when applying the far-field boundary conditions.

Following Salas et al. [21], a compressible potential vortex solution is added as a perturbation to the free-stream quantities at the far-field boundary. With the present nondimensionalization, the free-stream velocity components are $u_\infty = M_\infty \cos\alpha$

and $v_\infty = M_\infty \sin\alpha$, where M_∞ is the free-stream Mach number, and α is the angle of incidence of the flow relative to the x axis. The perturbed far-field boundary velocities are defined as

$$u_\mathrm{f} = u_\infty + \frac{\beta\Gamma \sin(\theta)}{2\pi r\left(1 - M_\infty^2 \sin^2(\theta - \alpha)\right)} \tag{4.164}$$

and

$$v_\mathrm{f} = v_\infty - \frac{\beta\Gamma \cos(\theta)}{2\pi r\left(1 - M_\infty^2 \sin^2(\theta - \alpha)\right)}, \tag{4.165}$$

where the circulation $\Gamma = \frac{1}{2} M_\infty l C_l$, l is the chord length, C_l is the coefficient of lift, α is the angle of attack, $\beta = \sqrt{1 - M_\infty^2}$, and r, θ are polar coordinates to the point of application on the outer boundary relative to an origin at the quarter-chord point on the airfoil center line. A corrected speed of sound is used that enforces constant free-stream enthalpy at the boundary:

$$a_\mathrm{f}^2 = (\gamma - 1)\left(H_\infty - \frac{1}{2}(u_\mathrm{f}^2 + v_\mathrm{f}^2)\right). \tag{4.166}$$

Equations (4.164), (4.165) and (4.166) are used instead of free-stream values in defining the specified quantities for the far-field characteristic boundary conditions. The circulation Γ is determined by the solution and is not known at the outset; hence it must be calculated and updated as the iterations progress. At convergence, the value of Γ used in the far-field circulation correction is consistent with the lift coefficient computed for the airfoil.

Figure 4.8 shows the coefficient of lift C_l plotted against the inverse of the distance to the outer boundary for an inviscid flow over the NACA 0012 airfoil at $M_\infty = 0.63$, $\alpha = 2.0$ degrees. The distance to the outer boundary varies from 5 to 200 chord lengths, where outer mesh rings were eliminated from the largest mesh to produce the smaller meshes.

4.7 Three-Dimensional Algorithm

The three-dimensional form of the implicit algorithm follows the same development as the two-dimensional algorithm. The curvilinear coordinate transformation is carried out in the same fashion. The block and diagonal algorithms take the same format. Boundary conditions are analogous. In this section, we briefly outline the equations in three dimensions.

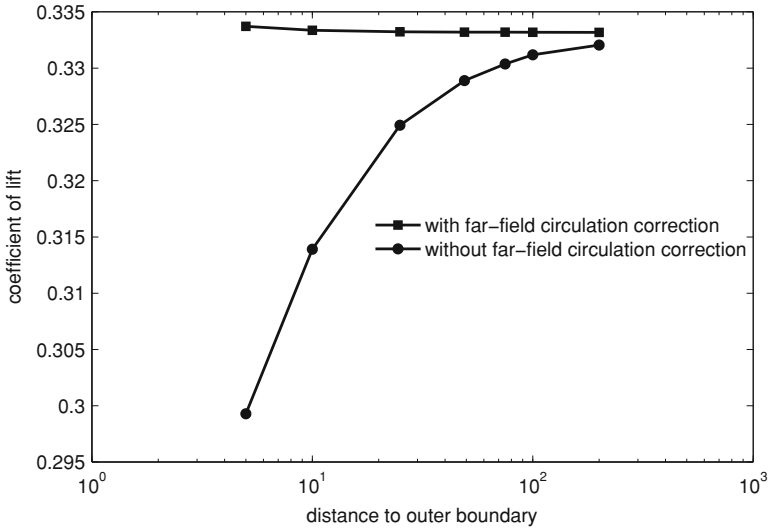

Fig. 4.8 Ėffect on lift coefficient of varying outer boundary distance (in chord lengths) with and without far-field circulation correction

4.7.1 Flow Equations

The full three-dimensional Navier-Stokes equations in strong conservation law form are reduced to the thin-layer form under the same restrictions and assumptions as in two dimensions. The equations in generalized curvilinear coordinates are

$$\partial_\tau \widehat{Q} + \partial_\xi \widehat{E} + \partial_\eta \widehat{F} + \partial_\zeta \widehat{G} = Re^{-1} \partial_\zeta \widehat{S}, \tag{4.167}$$

where

$$\widehat{Q} = J^{-1} \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix}, \quad \widehat{E} = J^{-1} \begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ \rho w U + \xi_z p \\ U(e+p) - \xi_t p \end{bmatrix},$$

$$\widehat{F} = J^{-1} \begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ \rho w V + \eta_z p \\ V(e+p) - \eta_t p \end{bmatrix}, \quad \widehat{G} = J^{-1} \begin{bmatrix} \rho W \\ \rho u W + \zeta_x p \\ \rho v W + \zeta_y p \\ \rho w W + \zeta_z p \\ W(e+p) - \zeta_t p \end{bmatrix}, \tag{4.168}$$

with

$$\begin{aligned} U &= \xi_t + \xi_x u + \xi_y v + \xi_z w, \\ V &= \eta_t + \eta_x u + \eta_y v + \eta_z w \\ W &= \zeta_t + \zeta_x u + \zeta_y v + \zeta_z w, \end{aligned} \tag{4.169}$$

and

$$\widehat{S} = J^{-1} \begin{bmatrix} 0 \\ \mu m_1 u_\zeta + (\mu/3) m_2 \zeta_x \\ \mu m_1 v_\zeta + (\mu/3) m_2 \zeta_y \\ \mu m_1 w_\zeta + (\mu/3) m_2 \zeta_z \\ \mu m_1 m_3 + (\mu/3) m_2 (\zeta_x u + \zeta_y v + \zeta_z w) \end{bmatrix}. \tag{4.170}$$

Here $m_1 = \zeta_x^2 + \zeta_y^2 + \zeta_z^2$, $m_2 = \zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta$, and $m_3 = (u^2 + v^2 + w^2)_\zeta/2 + Pr^{-1}(\gamma - 1)^{-1}(a^2)_\zeta$. Pressure is again related to the conservative flow variables, Q, by the equation of state:

$$p = (\gamma - 1)\left(e - \frac{1}{2}\rho(u^2 + v^2 + w^2)\right). \tag{4.171}$$

The metric terms are defined as

$$\begin{aligned} \xi_x &= J(y_\eta z_\zeta - y_\zeta z_\eta), \quad & \eta_x &= J(z_\xi y_\zeta - y_\xi z_\zeta) \\ \xi_y &= J(z_\eta x_\zeta - z_\zeta x_\eta), \quad & \eta_y &= J(x_\xi z_\zeta - z_\xi x_\zeta) \\ \xi_z &= J(x_\eta y_\zeta - y_\eta x_\zeta), \quad & \eta_z &= J(y_\xi x_\zeta - x_\xi y_\zeta) \\ \zeta_x &= J(y_\xi z_\eta - z_\xi y_\eta), \quad & \xi_t &= -x_\tau \xi_x - y_\tau \xi_y - z_\tau \xi_z \\ \zeta_y &= J(z_\xi x_\eta - x_\xi z_\eta), \quad & \eta_t &= -x_\tau \eta_x - y_\tau \eta_y - z_\tau \eta_z \\ \zeta_z &= J(x_\xi y_\eta - y_\xi x_\eta), \quad & \zeta_t &= -x_\tau \zeta_x - y_\tau \zeta_y - z_\tau \zeta_z \end{aligned} \tag{4.172}$$

with

$$J^{-1} = x_\xi y_\eta z_\zeta + x_\zeta y_\xi z_\eta + x_\eta y_\zeta z_\xi - x_\xi y_\zeta z_\eta - x_\eta y_\xi z_\zeta - x_\zeta y_\eta z_\xi. \tag{4.173}$$

4.7.2 Numerical Methods

The implicit approximate factorization algorithm applied to the three-dimensional equations is

$$\begin{aligned} &\left[I + h\delta_\xi \widehat{A}^n\right]\left[I + h\delta_\eta \widehat{B}^n\right]\left[I + h\delta_\zeta \widehat{C}^n - hRe^{-1}\delta_\zeta \widehat{M}^n\right] \Delta \widehat{Q}^n \\ &\quad = -h\left(\delta_\xi \widehat{E}^n + \delta_\eta \widehat{F}^n + \delta_\zeta \widehat{G}^n - Re^{-1}\delta_\zeta \widehat{S}^n\right). \end{aligned} \tag{4.174}$$

The three-dimensional inviscid flux Jacobians $\widehat{A}$, $\widehat{B}$, $\widehat{C}$ are defined in the Appendix along with the viscous flux Jacobian $\widehat{M}$. The spatial discretization, including the artificial dissipation, extends directly to three dimensions. Calculation of the grid metrics in three dimensions is discussed in Sect. 4.4.1. The diagonal algorithm in three dimensions has the form

$$T_\xi \left[I + h\,\delta_\xi\,\Lambda_\xi\right]\,\widehat{N}\,\left[I + h\,\delta_\eta\,\Lambda_\eta\right]\widehat{P}\left[I + h\,\delta_\zeta\,\Lambda_\zeta\right]T_\zeta^{-1}\Delta\widehat{Q}^n = \widehat{R}^n \quad (4.175)$$

with $\widehat{N} = T_\xi^{-1}T_\eta$ and $\widehat{P} = T_\eta^{-1}T_\zeta$.

A linear constant-coefficient Fourier analysis for the three-dimensional model wave equation shows unconditional instability for the three-dimensional factored algorithm in the absence of numerical dissipation. This is due to the cross term errors. In contrast to the case of two dimensions where the cross term errors just affect the rapid convergence capability of the algorithm at large time steps, in three dimensions they result in a weak instability. The method becomes stable when a small amount of artificial dissipation is added to the spatial discretization.

4.8 One-Dimensional Examples

In order to demonstrate the performance of the algorithm presented in this chapter, we present numerical results obtained for steady flows governed by the quasi-one-dimensional Euler equations and an unsteady flow in a shock tube. The flow conditions coincide with those associated with the exercises of Chap. 3 and the present chapter. Hence the results presented in this section provide a useful reference for the reader when developing the code associated with this chapter's exercises. These one-dimensional problems should not be used to assess the efficiency of the algorithm, as their properties are simply too different from multi-dimensional problems. In particular, the implicit operator is tightly banded, which is not the case in multidimensions.

Three problems are considered, a subsonic channel flow, a transonic channel flow, and a shock tube. Flow conditions are as described in Sect. 3.3. The implicit algorithm is implemented as described in this chapter, although the coordinate transformation, the approximate factorization, and the viscous terms are not needed in this context. Boundary conditions are handled explicitly based on prescribing or extrapolating Riemann invariants. Zeroth-order extrapolation is used for outgoing Riemann invariants, i.e. the boundary value is set to the value at the first interior node. This is not desirable but leads to fast convergence for the two steady problems and has no impact on the shock-tube problem. Linear extrapolation is preferred and is needed to obtain second-order accuracy. It can be implemented through some minor changes to how the boundary values are handled (for example by choosing an updated boundary value that is the average of the value calculated using linear extrapolation and the previous value) or through an implicit treatment of the boundary conditions.

Alternatively, convergence can be obtained with linear extrapolation through the use of a low Courant number (e.g. $C_n = 2$). In multidimensional external flows, good convergence can typically be obtained with linear extrapolation. Finally, in the implementation of the diagonal form, the contribution of the source term to the left-hand side operator is neglected.

The artificial dissipation coefficient values are $\kappa_2 = 0$, $\kappa_4 = 0.02$ for the subsonic channel flow problem, $\kappa_2 = 0.5$, $\kappa_4 = 0.02$ for both the transonic channel flow problem and the shock-tube problem. A nonzero value of κ_2 can be used for the subsonic problem but is not needed. The state at the inflow boundary is used as the initial condition for the channel flow problems. For these problems, which are steady, a local time step is calculated from (4.138) based on an input value of the Courant number. For the shock-tube problem, a constant time step is used based on an input Courant number and representative values of u and a. The values used are $u = 300\,\mathrm{m/s}$ and $a = 315\,\mathrm{m/s}$.

For the subsonic channel flow, Fig. 4.9 shows that the solution computed on a mesh with 49 interior nodes lies very close to the exact solution. Some oscillations are visible near the boundaries; these are associated with the zeroth-order extrapolation of the outgoing Riemann invariants. With linear extrapolation these are not seen. Results with 199 interior nodes are shown in Fig. 4.10; the oscillations are reduced.

One can compute the numerical error in density, for example, as

$$e_\rho = \sqrt{\sum_{j=1}^{M} \frac{(\rho_j - \rho_j^{\text{exact}})^2}{M}}, \tag{4.176}$$

where M is the number of grid nodes, and ρ^{exact} is the exact solution. The error in density is plotted versus the grid spacing in Fig. 4.11. The numerical solution was obtained with linear extrapolation of the outgoing Riemann invariants at the boundaries and $\kappa_2 = 0$. The slope of the log-log plot is very close to two, consistent with second-order accuracy. This is a good test to verify a code.

Figures 4.12 and 4.13 display some convergence histories for the block form of the implicit algorithm applied to the subsonic channel problem. The L_2 norm of the residual is plotted versus the number of iterations for various grid sizes and Courant numbers. Figure 4.12 shows the dependence on the Courant number for a grid with 99 interior nodes, while Fig. 4.13 shows the dependence on the number of nodes in the grid with $C_n = 40$.

The convergence of the diagonal form of the implicit algorithm is displayed in Fig. 4.14. The convergence behaviour of the diagonal form is comparable to that of the block form shown in Fig. 4.13. As a result, the savings associated with solving scalar pentadiagonal systems rather than block pentadiagonal systems translate into savings in computing time.

Results for the transonic channel flow problem are displayed in Figs. 4.15 through 4.17. The solutions again show good agreement with the exact solution, as shown in Fig. 4.15. Note in particular the manner in which the shock is captured with the

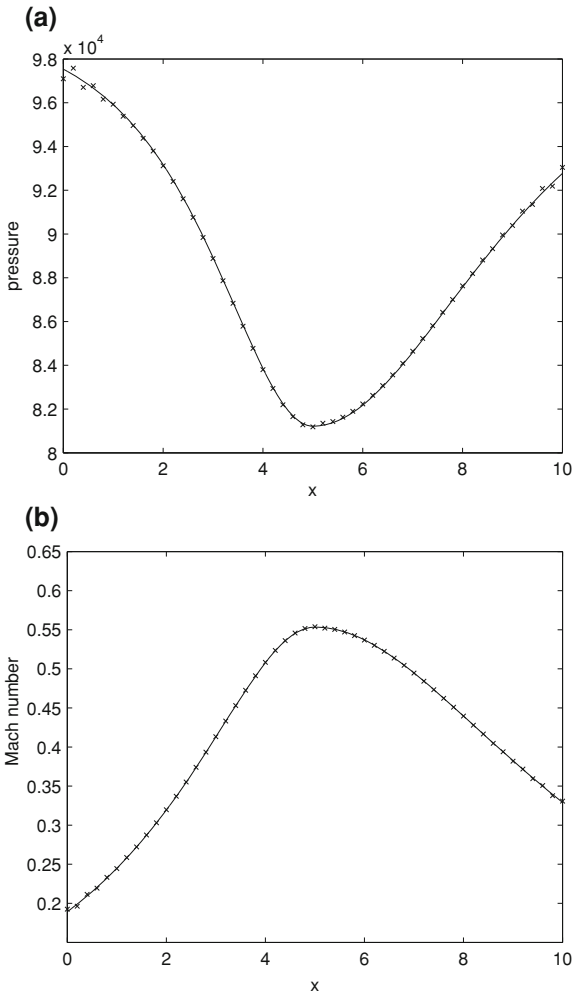

Fig. 4.9 Comparison of exact (-) solution for the subsonic channel flow problem with the numerical (x) solution computed on a grid with with 49 interior nodes

solution at one grid node lying midway between the values upstream and downstream of the shock. Figure 4.16 shows the residual convergence achieved with the block form of the algorithm at a Courant number of 120. The diagonal form proves to be unstable at a Courant number of 120 with a grid consisting of 99 interior nodes. However, at a Courant number of 70 it converges in slightly fewer iterations than the block form, as shown in Fig. 4.17.

Finally, Fig. 4.18 compares the numerical and exact solutions for the shock-tube problem on a grid with 400 cells with a maximum Courant number of unity. With the present numerical dissipation model, the shock wave and contact surface are spread out over several cells. This is the motivation for the methods described in Chap. 6.

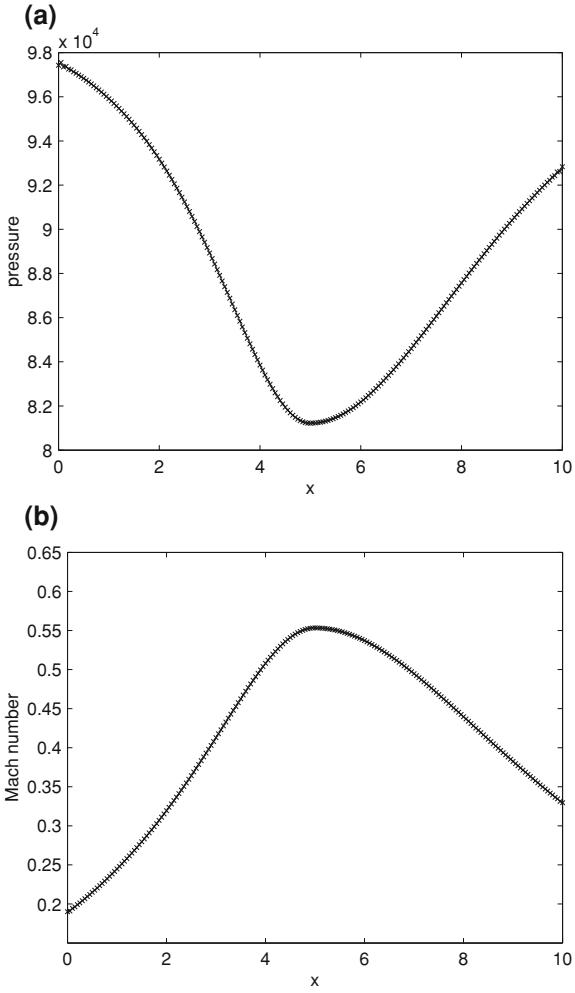

Fig. 4.10 Comparison of exact (-) solution for the subsonic channel flow problem with the numerical (x) solution computed on a grid with 199 interior nodes

4.9 Summary

The algorithm described in this chapter has the following key features:

- The discretization of the spatial derivatives is accomplished through second-order centered difference operators applied in a uniform computational space. This is facilitated by a curvilinear coordinate transformation that is defined implicitly through a structured grid. This approach is restricted to structured or block-structured grids. Numerical dissipation is added through a nonlinear artificial dissipation scheme that combines a third-order dissipative term in smooth regions

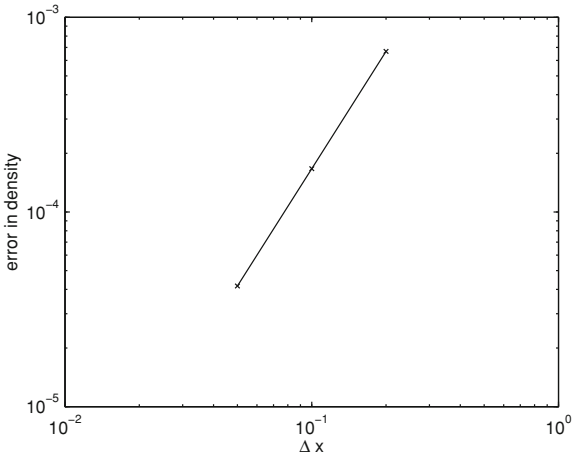

Fig. 4.11 Numerical error in density plotted versus grid spacing for the subsonic channel flow problem computed with linear extrapolation of outgoing Riemann invariants and $\kappa_2 = 0$

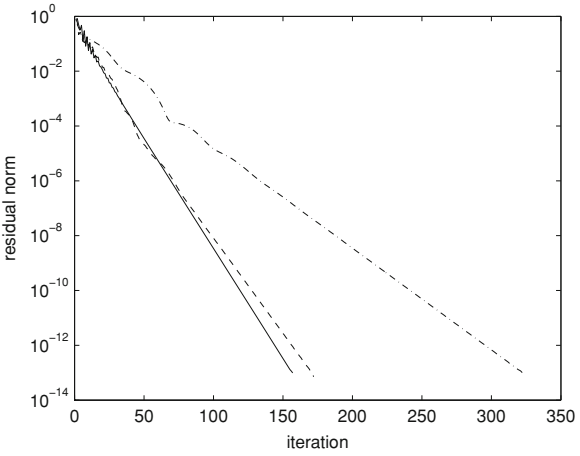

Fig. 4.12 Residual convergence histories for the subsonic channel flow problem using the block form of the implicit algorithm on a grid with 99 interior nodes with $C_n = 40$ (-), $C_n = 20$ (- -), and $C_n = 10$ (-·)

of the flow with a first-order term near shock waves. A pressure-based term is used as a shock sensor.

- After discretization in space, the original PDEs are converted to a large system of ODEs. For computations of steady flows, the implicit Euler method is used to follow a time dependent, though not time accurate, path to steady state. A local time linearization is applied, and the implicit operator is approximately factored in order to reduce the computational work required at each time step. With the

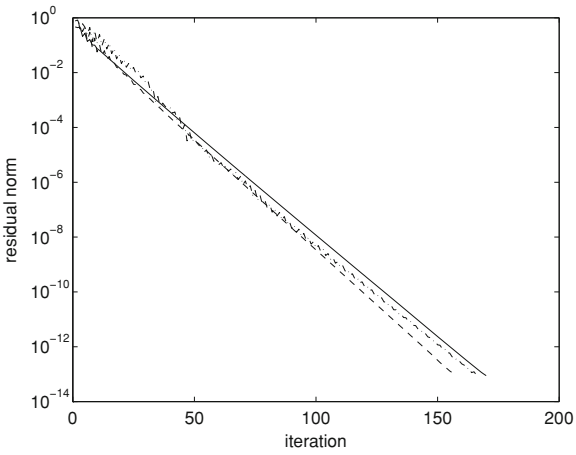

Fig. 4.13 Residual convergence histories for the subsonic channel flow problem using the block form of the implicit algorithm with $C_{\mathrm{n}} = 40$ on a grid with 49 interior nodes (-), 99 interior nodes (- -), and 199 interior nodes (-·)

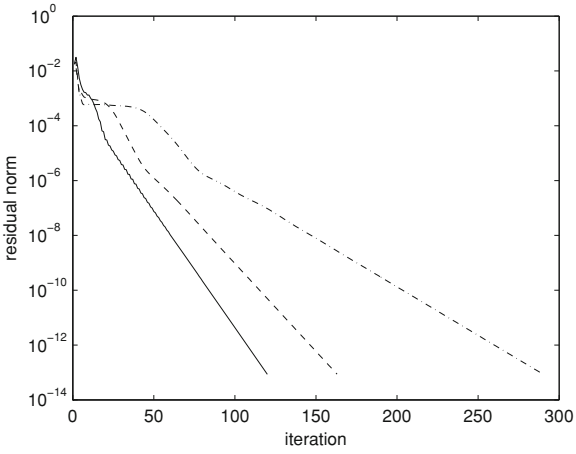

Fig. 4.14 Residual convergence histories for the subsonic channel flow problem using the diagonal form of the implicit algorithm on a grid with 99 interior nodes with $C_{\mathrm{n}} = 40$ (-), $C_{\mathrm{n}} = 20$ (- -), and $C_{\mathrm{n}} = 10$ (-·)

approximately factored form, block pentadiagonal linear systems must be solved. The approximate factorization has a detrimental effect on the convergence rate at large time steps but greatly reduces the computational cost per time step in comparison with a direct solution technique. The cost per time step can be further reduced through the use of the diagonal form, which reduces the necessary inversions to scalar pentadiagonal matrices. Convergence can be further accelerated through local time stepping and mesh sequencing. For time-accurate computations of unsteady flows, the block form of the approximate factorization algorithm can be applied to the second-order backward or the trapezoidal implicit time-marching methods. Alternatively, the dual time stepping approach can be used where the steady form of the algorithm is used to solve the nonlinear problem arising at each implicit time step.

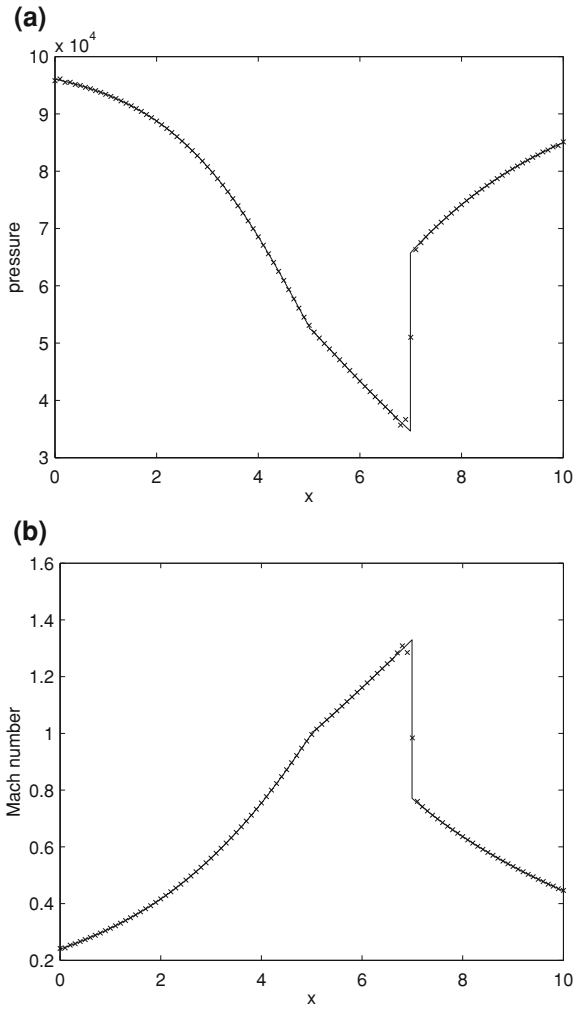

Fig. 4.15 Comparison of exact (-) solution for the transonic channel flow problem with the numerical (x) solution computed on a grid with with 99 interior nodes

4.10 Exercises

For related discussion, see Sect. 4.8.

4.1 Write a computer program to apply the implicit finite-difference algorithm presented in this chapter to the quasi-one-dimensional Euler equations for the following subsonic problem. $S(x)$ is given by

$$S(x) = \begin{cases} 1 + 1.5\left(1 - \frac{x}{5}\right)^2 & 0 \le x \le 5 \\ 1 + 0.5\left(1 - \frac{x}{5}\right)^2 & 5 \le x \le 10 \end{cases} \tag{4.177}$$

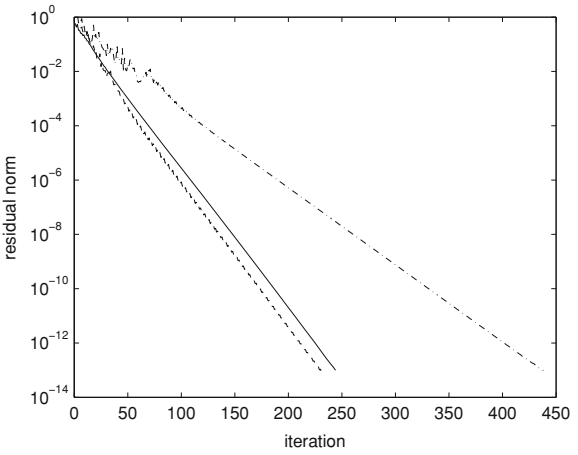

Fig. 4.16 Residual convergence histories for the transonic channel flow problem using the block form of the implicit algorithm with $C_{\mathrm{n}} = 120$ on a grid with 49 interior nodes (-), 99 interior nodes (- -), and 199 interior nodes (-·)

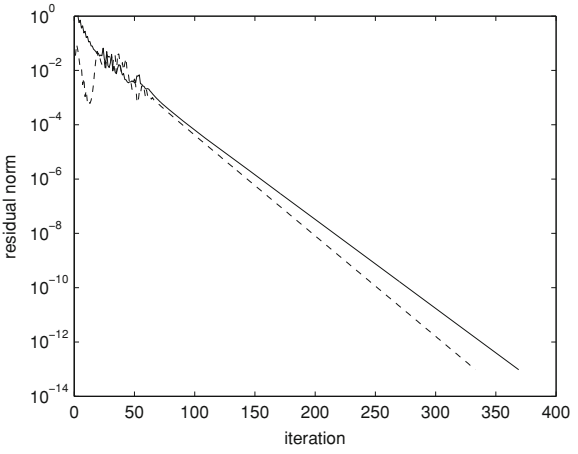

Fig. 4.17 Residual convergence histories for the transonic channel flow problem using the block form (-) and the diagonal form (- -) of the implicit algorithm with $C_{\mathrm{n}} = 70$ on a grid with 99 interior nodes

where $S(x)$ and x are in meters. The fluid is air, which is considered to be a perfect gas with $R = 287\ \mathrm{N\,m\,kg^{-1}\,K^{-1}}$, and $\gamma = 1.4$, the total temperature is $T_0 = 300$ K, and the total pressure at the inlet is $p_{01} = 100$ kPa. The flow is subsonic throughout the channel, with $S^* = 0.8$. Use implicit Euler time marching with and without the diagonal form. Use the nonlinear scalar artificial dissipation model. Compare your solution with the exact solution computed in Exercise 3.1. Show the convergence history for each case. Experiment with parameters, such as the Courant number and the artificial dissipation coefficients, to examine their effect on convergence and accuracy.

4.2 Repeat Exercise 4.1 for a transonic flow in the same channel. The flow is subsonic at the inlet, there is a shock at $x = 7$, and $S^* = 1$. Compare your solution with that calculated in Exercise 3.2.

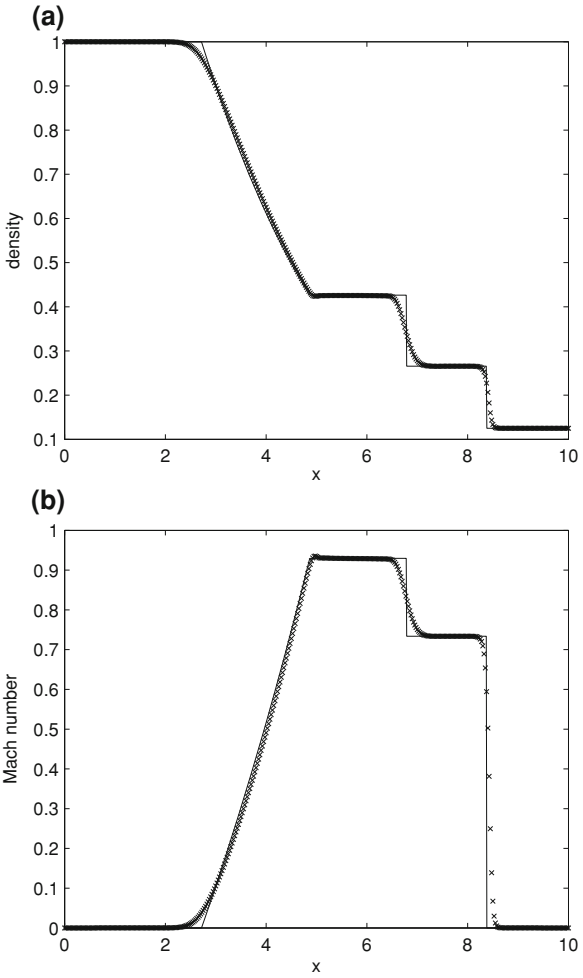

Fig. 4.18 Comparison of the exact solution (-) for the shock-tube problem at $t = 6.1$ ms with the numerical solution (x) computed on a grid with 400 cells with a maximum Courant number of unity

4.3 Write a computer program to apply the implicit finite-difference algorithm presented in this chapter to the following shock-tube problem: $p_L = 10^5$, $\rho_L = 1$, $p_R = 10^4$, and $\rho_R = 0.125$, where the pressures are in Pa and the densities in kg/m^3. The fluid is a perfect gas with $\gamma = 1.4$. Use both implicit Euler and second-order backwards time marching with and without the diagonal form. Compare your solution at $t = 6.1$ ms with that found in Exercise 3.3. Examine the effect of the time step and the artificial dissipation parameters on the accuracy of the solution.

Appendix: Flux Jacobian Eigensystems in Two and Three Dimensions

The flux Jacobian matrices of Eq. 4.104 have real eigenvalues and a complete set of eigenvectors. The similarity transforms are

$$\widehat{A} = T_\xi \Lambda_\xi T_\xi^{-1} \quad \text{and} \quad \widehat{B} = T_\eta \Lambda_\eta T_\eta^{-1}. \tag{4.178}$$

where

$$\Lambda_\xi = \begin{bmatrix} U & & & \\ & U & & \\ & & U + a\sqrt{\xi_x^2 + \xi_y^2} & \\ & & & U - a\sqrt{\xi_x^2 + \xi_y^2} \end{bmatrix} \tag{4.179}$$

$$\Lambda_\eta = \begin{bmatrix} V & & & \\ & V & & \\ & & V + a\sqrt{\eta_x^2 + \eta_y^2} & \\ & & & V - a\sqrt{\eta_x^2 + \eta_y^2} \end{bmatrix}, \tag{4.180}$$

with

$$T_\kappa = \begin{bmatrix} 1 & 0 & \alpha & \alpha \\ u & \widetilde{\kappa}_y \rho & \alpha(u + \widetilde{\kappa}_x a) & \alpha(u - \widetilde{\kappa}_x a) \\ v & -\widetilde{\kappa}_x \rho & \alpha(v + \widetilde{\kappa}_y a) & \alpha(v - \widetilde{\kappa}_y a) \\ \frac{\phi^2}{(\gamma-1)} & \rho(\widetilde{\kappa}_y u - \widetilde{\kappa}_x v) & \alpha\left[\frac{\phi^2+a^2}{(\gamma-1)} + a\widetilde{\theta}\right] & \alpha\left[\frac{\phi^2+a^2}{(\gamma-1)} - a\widetilde{\theta}\right] \end{bmatrix} \tag{4.181}$$

$$T_\kappa^{-1} = \begin{bmatrix} (1 - \phi^2/a^2) & (\gamma - 1)u/a^2 & (\gamma - 1)v/a^2 & -(\gamma - 1)/a^2 \\ -(\widetilde{\kappa}_y u - \widetilde{\kappa}_x v)/\rho & \widetilde{\kappa}_y/\rho & -\widetilde{\kappa}_x/\rho & 0 \\ \beta(\phi^2 - a\widetilde{\theta}) & \beta[\widetilde{\kappa}_x a - (\gamma - 1)u] & \beta[\widetilde{\kappa}_y a - (\gamma - 1)v] & \beta(\gamma - 1) \\ \beta(\phi^2 + a\widetilde{\theta}) & -\beta[\widetilde{\kappa}_x a + (\gamma - 1)u] & -\beta[\widetilde{\kappa}_y a + (\gamma - 1)v] & \beta(\gamma - 1) \end{bmatrix}, \tag{4.182}$$

and $\alpha = \rho/(\sqrt{2}a)$, $\beta = 1/(\sqrt{2}\rho a)$, $\widetilde{\theta} = \widetilde{\kappa}_x u + \widetilde{\kappa}_y v$, $\phi = \frac{1}{2}(\gamma - 1)(u^2 + v^2)$ and, for example, $\widetilde{\kappa}_x = \kappa_x/\sqrt{\kappa_x^2 + \kappa_y^2}$.

Relations exist between T_ξ and T_η of the form

$$\widehat{N} = T_{\xi}^{-1} T_{\eta}, \quad \widehat{N}^{-1} = T_{\eta}^{-1} T_{\xi}, \tag{4.183}$$

where

$$\widehat{N} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & m_1 & -\mu m_2 & \mu m_2 \\ 0 & \mu m_2 & \mu^2(1+m_1) & \mu^2(1-m_1) \\ 0 & -\mu m_2 & \mu^2(1-m_1) & \mu^2(1+m_1) \end{bmatrix}, \tag{4.184}$$

and

$$\widehat{N}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & m_1 & \mu m_2 & -\mu m_2 \\ 0 & -\mu m_2 & \mu^2(1+m_1) & \mu^2(1-m_1) \\ 0 & \mu m_2 & \mu^2(1-m_1) & \mu^2(1+m_1) \end{bmatrix}, \tag{4.185}$$

with $m_1 = \left(\widetilde{\xi}_x\ \widetilde{\eta}_x + \widetilde{\xi}_y \widetilde{\eta}_y\right),\ m_2 = \left(\widetilde{\xi}_x\ \widetilde{\eta}_y - \widetilde{\xi}_y \widetilde{\eta}_x\right)$ and $\mu = 1/\sqrt{2}$. It is interesting to note that the matrix $\widehat{N}$ is only a function of the metrics and not the flow variables.

In three dimensions the Jacobian matrices $\widehat{A},\ \widehat{B},\ \text{or}\ \widehat{C} =$

$$\begin{bmatrix} \kappa_t & \kappa_x & \kappa_y & \kappa_z & 0 \\ \kappa_x\phi^2 - u\theta & \kappa_t + \theta - \kappa_x(\gamma-2)u & \kappa_y u - \kappa_x(\gamma-1)v & \kappa_z u - \kappa_x(\gamma-1)w & \kappa_x(\gamma-1) \\ \kappa_y\phi^2 - v\theta & \kappa_x v - \kappa_y(\gamma-1)u & \kappa_t + \theta - \kappa_y(\gamma-2)v & \kappa_z v - \kappa_y(\gamma-1)w & \kappa_y(\gamma-1) \\ \kappa_z\phi^2 - w\theta & \kappa_x w - \kappa_z(\gamma-1)u & \kappa_y w - \kappa_z(\gamma-1)v & \kappa_t + \theta - \kappa_z(\gamma-2)w & \kappa_z(\gamma-1) \\ -\theta\left(\gamma e/\rho - 2\phi^2\right) & \kappa_x\left(\gamma e/\rho - \phi^2\right) - (\gamma-1)u\theta & \kappa_y\left(\gamma e\rho^{-1} - \phi^2\right) - (\gamma-1)v\theta & \kappa_z\left(\gamma e\rho^{-1} - \phi^2\right) - (\gamma-1)w\theta & \kappa_t + \gamma\theta \end{bmatrix}, \tag{4.186}$$

where

$$\begin{aligned} \theta &= \kappa_x u + \kappa_y v + \kappa_z w \\ \phi^2 &= (\gamma-1)\left(\frac{u^2+v^2+w^2}{2}\right), \end{aligned} \tag{4.187}$$

with $\kappa = \xi\ ,\ \eta,\ \text{or}\ \zeta$ for $\widehat{A},\ \widehat{B},\ \text{or}\ \widehat{C}$, respectively.

The viscous flux Jacobian is

$$\widehat{M} = J^{-1} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ m_{21} & \alpha_1\partial_\zeta(\rho^{-1}) & \alpha_2\partial_\zeta(\rho^{-1}) & \alpha_3\partial_\zeta(\rho^{-1}) & 0 \\ m_{31} & \alpha_2\partial_\zeta(\rho^{-1}) & \alpha_4\partial_\zeta(\rho^{-1}) & \alpha_5\partial_\zeta(\rho^{-1}) & 0 \\ m_{41} & \alpha_3\partial_\zeta(\rho^{-1}) & \alpha_5\partial_\zeta(\rho^{-1}) & \alpha_6\partial_\zeta(\rho^{-1}) & 0 \\ m_{51} & m_{52} & m_{53} & m_{54} & \alpha_0\partial_\zeta(\rho^{-1}) \end{bmatrix} J, \quad (4.188)$$

where

$$\begin{aligned} m_{21} &= -\alpha_1\partial_\zeta(u/\rho) - \alpha_2\partial_\zeta(v/\rho) - \alpha_3\partial_\zeta(w/\rho) \\ m_{31} &= -\alpha_2\partial_\zeta(u/\rho) - \alpha_4\partial_\zeta(v/\rho) - \alpha_5\partial_\zeta(w/\rho) \\ m_{41} &= -\alpha_3\partial_\zeta(u/\rho) - \alpha_5\partial_\zeta(v/\rho) - \alpha_6\partial_\zeta(w/\rho) \\ m_{51} &= \alpha_0\partial_\zeta\left[-(e/\rho^2) + (u^2+v^2+w^2)/\rho\right] \\ &\quad -\alpha_1\partial_\zeta(u^2/\rho) - \alpha_4\partial_\zeta(v^2/\rho) - \alpha_6\partial_\zeta(w^2/\rho) \\ &\quad -2\alpha_2\partial_\zeta(uv/\rho) - 2\alpha_3\partial_\zeta(uw/\rho) - 2\alpha_5\partial_\zeta(vw/\rho) \\ m_{52} &= -\alpha_0\partial_\zeta(u/\rho) - m_{21}, \qquad m_{53} = -\alpha_0\partial_\zeta(v/\rho) - m_{31} \\ m_{54} &= -\alpha_0\partial_\zeta(w/\rho) - m_{41}, \qquad m_{44} = \alpha_4\partial_\zeta(\rho^{-1}) \\ \alpha_0 &= \gamma\mu Pr^{-1}({\zeta_x}^2 + {\zeta_y}^2 + {\zeta_z}^2), \quad \alpha_1 = \mu[(4/3){\zeta_x}^2 + {\zeta_y}^2 + {\zeta_z}^2] \\ \alpha_2 &= (\mu/3)\zeta_x\zeta_y, \quad \alpha_3 = (\mu/3)\zeta_x\zeta_z, \quad \alpha_4 = \mu[{\zeta_x}^2 + (4/3){\zeta_y}^2 + {\zeta_z}^2] \\ \alpha_5 &= (\mu/3)\zeta_y\zeta_z, \qquad \alpha_6 = \mu[{\zeta_x}^2 + {\zeta_y}^2 + (4/3){\zeta_z}^2]. \end{aligned} \quad (4.189)$$

The eigensystem decompositions of the three-dimensional Jacobians have the form $\widehat{A} = T_\xi \Lambda_\xi T_\xi^{-1}$, $\widehat{B} = T_\eta \Lambda_\eta T_\eta^{-1}$, and $\widehat{C} = T_\zeta \Lambda_\zeta T_\zeta^{-1}$. The eigenvalues are

$$\begin{aligned} \lambda_1 &= \lambda_2 = \lambda_3 = \kappa_t + \kappa_x u + \kappa_y v + \kappa_z w \\ \lambda_4 &= \lambda_1 + \kappa a, \quad \lambda_5 = \lambda_1 - \kappa a \\ \kappa &= \sqrt{\kappa_x^2 + \kappa_y^2 + \kappa_z^2}. \end{aligned} \quad (4.190)$$

The matrix T_κ, representing the left eigenvectors, is

$$T_\kappa = \begin{bmatrix} \tilde{\kappa}_x & \tilde{\kappa}_y & \tilde{\kappa}_z & \alpha & \alpha \\ \tilde{\kappa}_x u & \tilde{\kappa}_y u - \tilde{\kappa}_z \rho & \tilde{\kappa}_z u + \tilde{\kappa}_y \rho & \alpha(u + \tilde{\kappa}_x a) & \alpha(u - \tilde{\kappa}_x a) \\ \tilde{\kappa}_x v + \tilde{\kappa}_z \rho & \tilde{\kappa}_y u & \tilde{\kappa}_z v - \tilde{\kappa}_x \rho & \alpha(v + \tilde{\kappa}_y a) & \alpha(v - \tilde{\kappa}_y a) \\ \tilde{\kappa}_x w + \tilde{\kappa}_y \rho & \tilde{\kappa}_y w + \tilde{\kappa}_x \rho & \tilde{\kappa}_z w & \alpha(w + \tilde{\kappa}_z a) & \alpha(w - \tilde{\kappa}_z a) \\ \left[\tilde{\kappa}_x \phi^2/(\gamma-1) + \rho(\tilde{\kappa}_z v - \tilde{\kappa}_y w)\right] & \left[\tilde{\kappa}_y \phi^2/(\gamma-1) + \rho(\tilde{\kappa}_x w - \tilde{\kappa}_z u)\right] & \left[\tilde{\kappa}_z \phi^2/(\gamma-1) + \rho(\tilde{\kappa}_y u - \tilde{\kappa}_x v)\right] & \alpha\left[(\phi^2 + a^2)/(\gamma-1) + \tilde{\theta} a\right] & \alpha\left[(\phi^2 + a^2)/(\gamma-1) + \tilde{\theta} a\right] \end{bmatrix}, \tag{4.191}$$

where

$$\alpha = \frac{\rho}{\sqrt{2}a}, \quad \widetilde{\kappa}_x = \frac{\kappa_x}{\kappa}, \quad \widetilde{\kappa}_y = \frac{\kappa_y}{\kappa}, \quad \widetilde{\kappa}_z = \frac{\kappa_z}{\kappa}, \quad \widetilde{\theta} = \frac{\theta}{\kappa}. \tag{4.192}$$

The corresponding T_κ^{-1} is

$$T_\kappa^{-1} = \begin{bmatrix} \widetilde{\kappa}_x(1 - \phi^2/a^2) - (\widetilde{\kappa}_z v - \widetilde{\kappa}_y w)/\rho & \widetilde{\kappa}_x(\gamma-1)u/a^2 & \widetilde{\kappa}_x(\gamma-1)v/a^2 + \widetilde{\kappa}_z/\rho & \widetilde{\kappa}_x(\gamma-1)w/a^2 - \widetilde{\kappa}_y/\rho & -\widetilde{\kappa}_x(\gamma-1)/a^2 \\ \widetilde{\kappa}_y(1 - \phi^2/a^2) - (\widetilde{\kappa}_x w - \widetilde{\kappa}_z u)/\rho & \widetilde{\kappa}_y(\gamma-1)u/a^2 - \widetilde{\kappa}_z/\rho & \widetilde{\kappa}_y(\gamma-1)v/a^2 & \widetilde{\kappa}_y(\gamma-1)w/a^2 + \widetilde{\kappa}_x/\rho & -\widetilde{\kappa}_y(\gamma-1)/a^2 \\ \widetilde{\kappa}_z(1 - \phi^2/a^2) - (\widetilde{\kappa}_y u - \widetilde{\kappa}_x v)/\rho & \widetilde{\kappa}_z(\gamma-1)u/a^2 + \widetilde{\kappa}_y/\rho & \widetilde{\kappa}_z(\gamma-1)v/a^2 - \widetilde{\kappa}_x/\rho & \widetilde{\kappa}_z(\gamma-1)w/a^2 & -\widetilde{\kappa}_z(\gamma-1)/a^2 \\ \beta(\phi^2 - \widetilde{\theta}a) & -\beta[(\gamma-1)u - \widetilde{\kappa}_x a] & -\beta[(\gamma-1)v - \widetilde{\kappa}_y a] & -\beta[(\gamma-1)w - \widetilde{\kappa}_z a] & \beta(\gamma-1) \\ \beta(\phi^2 + \widetilde{\theta}a) & -\beta[(\gamma-1)u + \widetilde{\kappa}_x a] & -\beta[(\gamma-1)v + \widetilde{\kappa}_y a] & -\beta[(\gamma-1)w + \widetilde{\kappa}_z a] & \beta(\gamma-1) \end{bmatrix}, \tag{4.193}$$

where

$$\beta = \frac{1}{\sqrt{2}\rho a}. \tag{4.194}$$

References

1. Beam, R.M., Warming, R.F.: An implicit finite-difference algorithm for hyperbolic systems in conservation law form. J. Comput. Phys. **22**, 87–110 (1976)
2. Steger, J.L.: Implicit finite difference simulation of flow about arbitrary geometries with application to airfoils. AIAA Paper 77-665 (1977)
3. Warming, R.F., Beam, R.M.: On the construction and application of implicit factored schemes for conservation laws. In: SIAM-AMS Proceedings, vol. 11 (1978)
4. Pulliam, T.H., Steger, J.L.: Implicit finite-difference simulations of three dimensional compressible flow. AIAA J. **18**, 159–167 (1980)
5. Pulliam, T.H., Chaussee, D.S.: A diagonal form of an implicit approximate factorization algorithm. J. Comput. Phys. **39**, 347–363 (1981)
6. Pulliam, T.H.: Efficient solution methods for the Navier-Stokes equations. In: Von Karman Institute for Fluid Dynamics Numerical Techniques for Viscous Flow Calculations in Turbomachinery Bladings (1986)
7. Viviand, H.: Formes conservatives des équations de la dynamique des gaz. Recherche Aérospatiale **1**, 65–66 (1974)
8. Vinokur, M.: Conservation equations of gasdynamics in curvilinear coordinate systems. J. Comput. Phys. **14**, 105–125 (1974)
9. Baldwin, B.S., Lomax, H.: Thin-layer approximation and algebraic model for separated turbulent flows. AIAA Paper 78-257 (1978)
10. Gustafsson, B.: The convergence rate for difference approximations to mixed initial boundary value problems. Math. Comput. **29**, 396–406 (1975)
11. Thomas, P.D., Lombard, C.K.: Geometric conservation law and its application to flow computations on moving grids. AIAA J. **17**, 1030–1037 (1979)
12. Jameson, A., Schmidt, W., Turkel, E.: Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes. AIAA Paper 81-1259 (1981)
13. Lomax, H., Pulliam, T.H., Zingg, D.W.: Fundamentals of Computational Fluid Dynamics. Springer, Berlin (2001)
14. Pulliam, T.H.: Artificial dissipation models for the Euler equations. AIAA J. **24**, 1931–1940 (1986)
15. Saad, Y., Schultz, M.H.: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **7**, 856–869 (1986)
16. Warming, R.F., Beam, R.M., Hyett, B.J.: Diagonalization and simultaneous symmetrization of the gas-dynamic matrices. Math. Comput. **29**, 1037–1045 (1975)
17. Chakravarty, S.: Euler equations-implicit schemes and implicit boundary conditions. AIAA Paper 82-0228 (1982)
18. Colonius, T., Lele, S.K.: Computational aeroacoustics: progress on nonlinear problems of sound generation. Prog. Aerosp. Sci. **40**, 345–416 (2004)
19. Svard, M., Carpenter, M.H., Nordström, J.: A stable high-order finite difference scheme for the compressible Navier-Stokes equations, far-field boundary conditions. J. Comput. Phys. **225**, 1020–1038 (2007)
20. Osusky, M., Zingg, D.W.: Parallel Newton-Krylov-Schur Flow Solver for the Navier-Stokes Equations. AIAA J. **51**, 2833–2851 (2013)
21. Salas, M., Jameson, A., Melnik, R.A.: Comparative study of the nonuniqueness problem of the potential equation. AIAA Paper 83-1888 (1983)