Chapter 3

FINITE-DIFFERENCE APPROXIMATIONS

In common with the equations governing unsteady fluid flow, our model equations contain partial derivatives with respect to both space and time. One can approximate these simultaneously and then solve the resulting difference equations. Alternatively, one can approximate the spatial derivatives first, thereby producing a system of ordinary differential equations. The time derivatives are approximated next, leading to a *time-marching method* which produces a set of difference equations. This is the approach emphasized here. In this chapter, the concept of finite-difference approximations to partial derivatives is presented. While these can be applied either to spatial derivatives or time derivatives, our emphasis in this chapter is on spatial derivatives; time derivatives are treated in Chapter 6. Strategies for applying these finite-difference approximations will be discussed in Chapter 4.

All of the material below is presented in a Cartesian system. We emphasize the fact that quite general classes of meshes expressed in general curvilinear coordinates in *physical space* can be transformed to a uniform Cartesian mesh with equispaced intervals in a so-called *computational space*, as shown in Figure 3.1. The *computational space* is uniform; all the geometric variation is absorbed into variable coefficients of the transformed equations. For this reason, in much of the following accuracy analysis, we use an equispaced Cartesian system without being unduly restrictive or losing practical application.

3.1 Meshes and Finite-Difference Notation

The simplest mesh involving both time and space is shown in Figure 3.2. Inspection of this figure permits us to define the terms and notation needed to describe finite-

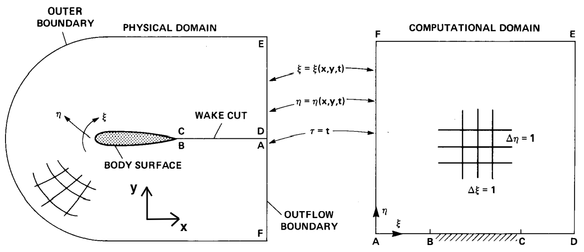

Figure 3.1: Physical and computational spaces

difference approximations. In general, the dependent variables, u, for example, are functions of the independent variables t, and x, y, z. For the first several chapters we consider primarily the 1-D case $u = u(x,t)$. When only one variable is denoted, dependence on the other is assumed. The mesh index for x is *always* j, and that for t is *always* n. Then on an equispaced grid

$$x = x_j = j\Delta x \tag{3.1}$$
$$t = t_n = n\Delta t = nh \tag{3.2}$$

where Δx is the spacing in x and Δt the spacing in t, as shown in Figure 3.2. Note that $h = \Delta t$ throughout. Later k and l are used for y and z in a similar way. When n, j, k, l are used for other purposes (which is sometimes necessary), local context should make the meaning obvious.

The convention for subscript and superscript indexing is as follows:

$$\begin{aligned} u(t + kh) &= u([n + k]h) = u_{n+k} \\ u(x + m\Delta x) &= u([j + m]\Delta x) = u_{j+m} \\ u(x + m\Delta x, t + kh) &= u_{j+m}^{(n+k)} \end{aligned} \tag{3.3}$$

Notice that when used alone, both the time and space indices appear as a subscript, but when used together, time is always a superscript and is usually enclosed with parentheses to distinguish it from an exponent.

Derivatives are expressed according to the usual conventions. Thus for partial derivatives in space or time we use interchangeably

$$\partial_x u = \frac{\partial u}{\partial x}, \quad \partial_t u = \frac{\partial u}{\partial t}, \quad \partial_{xx} u = \frac{\partial^2 u}{\partial x^2}, \quad \text{etc.} \tag{3.4}$$

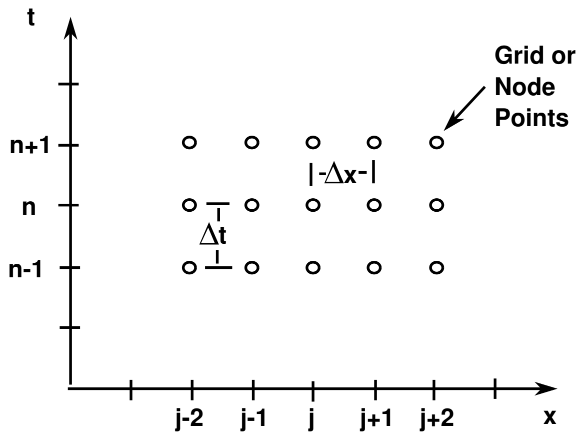

Figure 3.2: Space-time grid arrangement.

For the ordinary time derivative in the study of ODE's we use

$$u' = \frac{du}{dt} \tag{3.5}$$

In this text, subscripts on dependent variables are never used to express derivatives. Thus u_x will not be used to represent the first derivative of u with respect to x.

The notation for difference approximations follows the same philosophy, but (with one exception) it is not unique. By this we mean that the symbol δ is used to represent a difference approximation to a derivative such that, for example,

$$\delta_x \approx \partial_x, \qquad \delta_{xx} \approx \partial_{xx} \tag{3.6}$$

but the precise nature (and order) of the approximation is not carried in the symbol δ. Other ways are used to determine its precise meaning. The one exception is the symbol Δ, which is defined such that

$$\Delta t_n = t_{n+1} - t_n, \quad \Delta x_j = x_{j+1} - x_j, \quad \Delta u_n = u_{n+1} - u_n, \quad \text{etc.} \tag{3.7}$$

When there is no subscript on Δt or Δx, the spacing is uniform.

3.2 Space Derivative Approximations

A difference approximation can be generated or evaluated by means of a simple Taylor series expansion. For example, consider $u(x, t)$ with t fixed. Then, following the notation convention given in Eqs. 3.1 to 3.3, $x = j\Delta x$ and $u(x + k\Delta x) = u(j\Delta x +$

$k\Delta x) = u_{j+k}$. Expanding the latter term about x gives[1]

$$u_{j+k} = u_j + (k\Delta x)\left(\frac{\partial u}{\partial x}\right)_j + \frac{1}{2}(k\Delta x)^2\left(\frac{\partial^2 u}{\partial x^2}\right)_j + \cdots + \frac{1}{n!}(k\Delta x)^n\left(\frac{\partial^n u}{\partial x^n}\right)_j + \cdots \quad (3.8)$$

Local difference approximations to a given partial derivative can be formed from linear combinations of u_j and u_{j+k} for $k = \pm 1, \pm 2, \cdots$.

For example, consider the Taylor series expansion for u_{j+1}:

$$u_{j+1} = u_j + (\Delta x)\left(\frac{\partial u}{\partial x}\right)_j + \frac{1}{2}(\Delta x)^2\left(\frac{\partial^2 u}{\partial x^2}\right)_j + \cdots + \frac{1}{n!}(\Delta x)^n\left(\frac{\partial^n u}{\partial x^n}\right)_j + \cdots \quad (3.9)$$

Now subtract u_j and divide by Δx to obtain

$$\frac{u_{j+1} - u_j}{\Delta x} = \left(\frac{\partial u}{\partial x}\right)_j + \frac{1}{2}(\Delta x)\left(\frac{\partial^2 u}{\partial x^2}\right)_j + \cdots \quad (3.10)$$

Thus the expression $(u_{j+1} - u_j)/\Delta x$ is a reasonable approximation for $\left(\frac{\partial u}{\partial x}\right)_j$ as long as Δx is small relative to some pertinent length scale. Next consider the space difference approximation $(u_{j+1} - u_{j-1})/(2\Delta x)$. Expand the terms in the numerator about j and regroup the result to form the following equation

$$\frac{u_{j+1} - u_{j-1}}{2\Delta x} - \left(\frac{\partial u}{\partial x}\right)_j = \frac{1}{6}\Delta x^2\left(\frac{\partial^3 u}{\partial x^3}\right)_j + \frac{1}{120}\Delta x^4\left(\frac{\partial^5 u}{\partial x^5}\right)_j \ldots \quad (3.11)$$

When expressed in this manner, it is clear that the discrete terms on the left side of the equation represent a first derivative with a certain amount of error which appears on the right side of the equal sign. It is also clear that the error depends on the grid spacing to a certain order. The error term containing the grid spacing to the lowest power gives the order of the method. From Eq. 3.10, we see that the expression $(u_{j+1} - u_j)/\Delta x$ is a first-order approximation to $\left(\frac{\partial u}{\partial x}\right)_j$. Similarly, Eq. 3.11 shows that $(u_{j+1} - u_{j-1})/(2\Delta x)$ is a second-order approximation to a first derivative. The latter is referred to as the three-point centered difference approximation, and one often sees the summary result presented in the form

$$\left(\frac{\partial u}{\partial x}\right)_j = \frac{u_{j+1} - u_{j-1}}{2\Delta x} + O(\Delta x^2) \quad (3.12)$$

[1]We assume that $u(x,t)$ is continuously differentiable.

3.3 Finite-Difference Operators

3.3.1 Point Difference Operators

Perhaps the most common examples of finite-difference formulas are the three-point centered-difference approximations for the first and second derivatives:[2]

$$\left(\frac{\partial u}{\partial x}\right)_j = \frac{1}{2\Delta x}(u_{j+1} - u_{j-1}) + O(\Delta x^2) \tag{3.13}$$

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_j = \frac{1}{\Delta x^2}(u_{j+1} - 2u_j + u_{j-1}) + O(\Delta x^2) \tag{3.14}$$

These are the basis for *point difference operators* since they give an approximation to a derivative at one discrete point in a mesh in terms of surrounding points. However, neither of these expressions tells us how other points in the mesh are differenced or how boundary conditions are enforced. Such additional information requires a more sophisticated formulation.

3.3.2 Matrix Difference Operators

Consider the relation

$$(\delta_{xx} u)_j = \frac{1}{\Delta x^2}(u_{j+1} - 2u_j + u_{j-1}) \tag{3.15}$$

which is a point difference approximation to a second derivative. Now let us derive a *matrix* operator representation for the same approximation. Consider the four point mesh with boundary points at a and b shown below. Notice that when we speak of "the number of points in a mesh", we mean the number of *interior* points excluding the boundaries.

	a	1	2	3	4	b
$x =$	0	$-$	$-$	$-$	$-$	π
$j =$		1	$\cdot$	$\cdot$	M	

Four point mesh. $\Delta x = \pi/(M+1)$

Now impose Dirichlet boundary conditions, $u(0) = u_a$, $u(\pi) = u_b$ and use the centered difference approximation given by Eq. 3.15 at every point in the mesh. We

[2] We will derive the second derivative operator shortly.

arrive at the four equations:

$$
\begin{aligned}
(\delta_{xx}u)_1 &= \frac{1}{\Delta x^2}(u_a - 2u_1 + u_2) \\
(\delta_{xx}u)_2 &= \frac{1}{\Delta x^2}(u_1 - 2u_2 + u_3) \\
(\delta_{xx}u)_3 &= \frac{1}{\Delta x^2}(u_2 - 2u_3 + u_4) \\
(\delta_{xx}u)_4 &= \frac{1}{\Delta x^2}(u_3 - 2u_4 + u_b)
\end{aligned}
\tag{3.16}
$$

Writing these equations in the more suggestive form

$$
\begin{array}{lcllllllll}
(\delta_{xx}u)_1 & = & (& u_a & -2u_1 & +u_2 & & & &)/\Delta x^2 \\
(\delta_{xx}u)_2 & = & (& & u_1 & -2u_2 & +u_3 & & &)/\Delta x^2 \\
(\delta_{xx}u)_3 & = & (& & & u_2 & -2u_3 & +u_4 & &)/\Delta x^2 \\
(\delta_{xx}u)_4 & = & (& & & & u_3 & -2u_4 & +u_b &)/\Delta x^2
\end{array}
\tag{3.17}
$$

it is clear that we can express them in a vector-matrix form, and further, that the resulting matrix has a very special form. Introducing

$$
\vec{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}, \qquad \left(\vec{bc}\right) = \frac{1}{\Delta x^2}\begin{bmatrix} u_a \\ 0 \\ 0 \\ u_b \end{bmatrix}
\tag{3.18}
$$

and

$$
A = \frac{1}{\Delta x^2}\begin{bmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & 1 & -2 & 1 \\ & & 1 & -2 \end{bmatrix}
\tag{3.19}
$$

we can rewrite Eq. 3.17 as

$$
\delta_{xx}\vec{u} = A\vec{u} + \left(\vec{bc}\right)
\tag{3.20}
$$

This example illustrates a matrix difference operator. Each line of a matrix difference operator is based on a point difference operator, but the point operators used from line to line are not necessarily the same. For example, boundary conditions may dictate that the lines at or near the bottom or top of the matrix be modified. In the extreme case of the matrix difference operator representing a spectral method, none of the lines is the same. The matrix operators representing the three-point central-difference approximations for a first and second derivative with Dirichlet boundary conditions on a four-point mesh are

$$\delta_x = \frac{1}{2\Delta x}\begin{bmatrix} 0 & 1 & & \\ -1 & 0 & 1 & \\ & -1 & 0 & 1 \\ & & -1 & 0 \end{bmatrix}, \quad \delta_{xx} = \frac{1}{\Delta x^2}\begin{bmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & 1 & -2 & 1 \\ & & 1 & -2 \end{bmatrix} \tag{3.21}$$

As a further example, replace the fourth line in Eq. 3.16 by the following point operator for a Neumann boundary condition (See Section 3.6.):

$$(\delta_{xx}u)_4 = \frac{2}{3}\frac{1}{\Delta x}\left(\frac{\partial u}{\partial x}\right)_b - \frac{2}{3}\frac{1}{\Delta x^2}(u_4 - u_3) \tag{3.22}$$

where the boundary condition is

$$\left(\frac{\partial u}{\partial x}\right)_{x=\pi} = \left(\frac{\partial u}{\partial x}\right)_b \tag{3.23}$$

Then the matrix operator for a three-point central-differencing scheme at interior points and *a second-order approximation* for a Neumann condition on the right is given by

$$\delta_{xx} = \frac{1}{\Delta x^2}\begin{bmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & 1 & -2 & 1 \\ & & 2/3 & -2/3 \end{bmatrix} \tag{3.24}$$

Each of these matrix difference operators is a square matrix with elements that are all zeros except for those along bands which are clustered around the central diagonal. We call such a matrix a *banded matrix* and introduce the notation

$$B(M : a, b, c) = \begin{array}{c} \begin{bmatrix} b & c & & & \\ a & b & c & & \\ & & \ddots & & \\ & & a & b & c \\ & & & a & b \end{bmatrix} \end{array} \begin{array}{c} 1 \\ \\ \\ \vdots \\ M \end{array} \tag{3.25}$$

where the matrix dimensions are $M \times M$. Use of M in the argument is optional, and the illustration is given for a simple *tridiagonal* matrix although any number of bands is a possibility. A tridiagonal matrix without constants along the bands can be expressed as $B(\vec{a}, \vec{b}, \vec{c})$. The arguments for a banded matrix are always odd in number and the central one *always* refers to the central diagonal.

We can now generalize our previous examples. Defining $\vec{u}$ as[3]

$$\vec{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_M \end{bmatrix} \tag{3.26}$$

we can approximate the second derivative of $\vec{u}$ by

$$\delta_{xx}\vec{u} = \frac{1}{\Delta x^2} B(1, -2, 1)\vec{u} + \left(\vec{bc}\right) \tag{3.27}$$

where $\left(\vec{bc}\right)$ stands for the vector holding the Dirichlet boundary conditions on the left and right sides:

$$\left(\vec{bc}\right) = \frac{1}{\Delta x^2}[u_a, 0, \cdots, 0, u_b]^T \tag{3.28}$$

If we prescribe Neumann boundary conditions on the right side, as in Eqs. 3.24 and 3.22, we find

$$\delta_{xx}\vec{u} \;=\; \frac{1}{\Delta x^2} B(\vec{a}, \vec{b}, 1)\vec{u} + \left(\vec{bc}\right) \tag{3.29}$$

where

$$\begin{aligned} \vec{a} &= [1, 1, \cdots, 2/3]^T \\ \vec{b} &= [-2, -2, -2, \cdots, -2/3]^T \\ \left(\vec{bc}\right) &= \frac{1}{\Delta x^2}\left[u_a, 0, 0, \cdots, \frac{2\Delta x}{3}\left(\frac{\partial u}{\partial x}\right)_b\right]^T \end{aligned}$$

Notice that the matrix operators given by Eqs. 3.27 and 3.29 carry more information than the point operator given by Eq. 3.15. In Eqs. 3.27 and 3.29, the boundary conditions have been uniquely specified and it is clear that the same point operator has been applied at every point in the field except at the boundaries. The ability to specify in the matrix derivative operator the exact nature of the approximation at the various points in the field including the boundaries permits the use of quite general constructions which will be useful later in considerations of stability.

[3]Note that $\vec{u}$ is a function of time only since each element corresponds to one specific spatial location.

Since we make considerable use of both matrix and point operators, it is important to establish a relation between them. A point operator is generally written for some derivative at the reference point j in terms of neighboring values of the function. For example

$$(\delta_x u)_j = a_2 u_{j-2} + a_1 u_{j-1} + b u_j + c_1 u_{j+1} \tag{3.30}$$

might be the point operator for a first derivative. The corresponding matrix operator has for its arguments the coefficients giving the weights to the values of the function at the various locations. A j-shift in the point operator corresponds to a diagonal shift in the matrix operator. Thus the matrix equivalent of Eq. 3.30 is

$$\delta_x \vec{u} = B(a_2, a_1, b, c_1, 0)\vec{u} \tag{3.31}$$

Note the addition of a zero in the fifth element which makes it clear that b is the coefficient of u_j.

3.3.3 Periodic Matrices

The above illustrated cases in which the boundary conditions are fixed. If the boundary conditions are *periodic*, the form of the matrix operator changes. Consider the eight-point periodic mesh shown below. This can either be presented on a linear mesh with repeated entries, or more suggestively on a circular mesh as in Figure 3.3. When the mesh is laid out on the perimeter of a circle, it doesn't really matter where the numbering starts as long as it "ends" at the point just preceding its starting location.

$\cdots$	7	8	1	2	3	4	5	6	7	8	1	2	$\cdots$
$x =$	–	–	0	–	–	–	–	–	–	–	2π	–	
$j =$			0	1	·	·	·	·	·	·	M		

Eight points on a linear periodic mesh. $\Delta x = 2\pi/M$

The matrix that represents differencing schemes for scalar equations on a periodic mesh is referred to as a *periodic* matrix. A typical periodic tridiagonal matrix operator with nonuniform entries is given for a 6-point mesh by

$$B_p(6 : \vec{a}, \vec{b}, \vec{c}) = \begin{bmatrix} b_1 & c_2 & & & & a_6 \\ a_1 & b_2 & c_3 & & & \\ & a_2 & b_3 & c_4 & & \\ & & a_3 & b_4 & c_5 & \\ & & & a_4 & b_5 & c_6 \\ c_1 & & & & a_5 & b_6 \end{bmatrix} \tag{3.32}$$

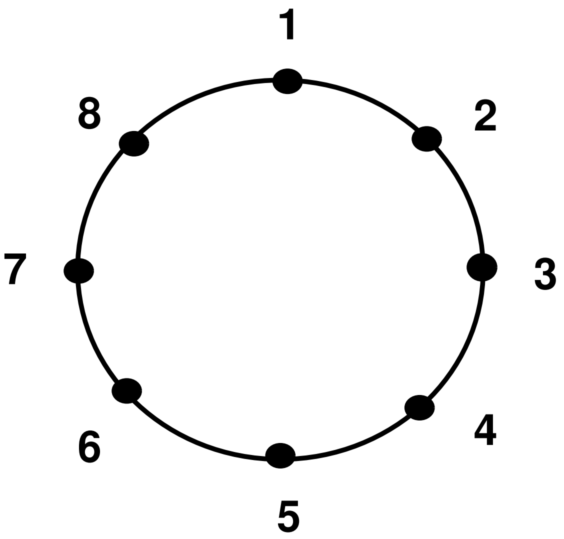

Figure 3.3: Eight points on a circular mesh.

3.3.4 Circulant Matrices

In general, as shown in the example, the elements along the diagonals of a periodic matrix are not constant. However, a special subset of a periodic matrix is the circulant matrix, formed when the elements along the various bands are constant. Circulant matrices play a vital role in our analysis. We will have much more to say about them later. The most general circulant matrix of order 4 is

$$\begin{bmatrix} b_0 & b_1 & b_2 & b_3 \\ b_3 & b_0 & b_1 & b_2 \\ b_2 & b_3 & b_0 & b_1 \\ b_1 & b_2 & b_3 & b_0 \end{bmatrix} \tag{3.33}$$

Notice that each row of a circulant matrix is shifted (see Figure 3.3) one element to the right of the one above it. The special case of a tridiagonal circulant matrix is given by

$$B_p(M : a, b, c) = \begin{array}{c} \begin{bmatrix} b & c & & & a \\ a & b & c & & \\ & & \ddots & & \\ & & a & b & c \\ c & & & a & b \end{bmatrix} \end{array} \begin{array}{c} 1 \\ \\ \\ \vdots \\ M \end{array} \tag{3.34}$$

When the standard three-point central-differencing approximations for a first and second derivative, see Eq. 3.21, are used with periodic boundary conditions, they take the form

$$(\delta_x)_p = \frac{1}{2\Delta x}\begin{bmatrix} 0 & 1 & & -1 \\ -1 & 0 & 1 & \\ & -1 & 0 & 1 \\ 1 & & -1 & 0 \end{bmatrix} = \frac{1}{2\Delta x}B_p(-1,0,1)$$

and

$$(\delta_{xx})_p = \frac{1}{\Delta x^2}\begin{bmatrix} -2 & 1 & & 1 \\ 1 & -2 & 1 & \\ & 1 & -2 & 1 \\ 1 & & 1 & -2 \end{bmatrix} = \frac{1}{\Delta x^2}B_p(1,-2,1) \tag{3.35}$$

Clearly, these special cases of periodic operators are also circulant operators. Later on we take advantage of this special property. Notice that there are no boundary condition vectors since this information is all interior to the matrices themselves.

3.4 Constructing Differencing Schemes of Any Order

3.4.1 Taylor Tables

The Taylor series expansion of functions about a fixed point provides a means for constructing finite-difference point-operators of any order. A simple and straightforward way to carry this out is to construct a "Taylor table," which makes extensive use of the expansion given by Eq. 3.8. As an example, consider Table 3.1, which represents a Taylor table for an approximation of a second derivative using three values of the function centered about the point at which the derivative is to be evaluated.

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_j - \frac{1}{\Delta x^2}(a\,u_{j-1} + b\,u_j + c\,u_{j+1}) = \,?$$

	u_j	$\Delta x \cdot \left(\frac{\partial u}{\partial x}\right)_j$	$\Delta x^2 \cdot \left(\frac{\partial^2 u}{\partial x^2}\right)_j$	$\Delta x^3 \cdot \left(\frac{\partial^3 u}{\partial x^3}\right)_j$	$\Delta x^4 \cdot \left(\frac{\partial^4 u}{\partial x^4}\right)_j$
$\Delta x^2 \cdot \left(\frac{\partial^2 u}{\partial x^2}\right)_j$			1		
$-a \cdot u_{j-1}$	$-a$	$-a \cdot (\text{-}1) \cdot \frac{1}{1}$	$-a \cdot (\text{-}1)^2 \cdot \frac{1}{2}$	$-a \cdot (\text{-}1)^3 \cdot \frac{1}{6}$	$-a \cdot (\text{-}1)^4 \cdot \frac{1}{24}$
$-b \cdot u_j$	$-b$				
$-c \cdot u_{j+1}$	$-c$	$-c \cdot (1) \cdot \frac{1}{1}$	$-c \cdot (1)^2 \cdot \frac{1}{2}$	$-c \cdot (1)^3 \cdot \frac{1}{6}$	$-c \cdot (1)^4 \cdot \frac{1}{24}$

Table 3.1. Taylor table for centered 3-point Lagrangian approximation to a second derivative.

The table is constructed so that some of the algebra is simplified. At the top of the table we see an expression with a question mark. This represents one of the questions that a study of this table can answer; namely, what is the local error caused by the use of this approximation? Notice that all of the terms in the equation appear in a column at the left of the table (although, in this case, Δx^2 has been multiplied into each term in order to simplify the terms to be put into the table). Then notice that at the head of each column there appears the common factor that occurs in the expansion of each term about the point j, that is,

$$\Delta x^k \cdot \left(\frac{\partial^k u}{\partial x^k}\right)_j \qquad k = 0, 1, 2, \cdots$$

The columns to the right of the leftmost one, under the headings, make up the Taylor table. Each entry is the coefficient of the term at the top of the corresponding column in the Taylor series expansion of the term to the left of the corresponding row. For example, the last row in the table corresponds to the Taylor series expansion of $-c\,u_{j+1}$:

$$\begin{aligned} -c\,u_{j+1} \;=\; & -c\,u_j - c \cdot (1) \cdot \frac{1}{1}\Delta x \cdot \left(\frac{\partial u}{\partial x}\right)_j - c \cdot (1)^2 \cdot \frac{1}{2}\Delta x^2 \cdot \left(\frac{\partial^2 u}{\partial x^2}\right)_j \\ & -c \cdot (1)^3 \cdot \frac{1}{6}\Delta x^3 \cdot \left(\frac{\partial^3 u}{\partial x^3}\right)_j - c \cdot (1)^4 \cdot \frac{1}{24}\Delta x^4 \cdot \left(\frac{\partial^4 u}{\partial x^4}\right)_j - \cdots \end{aligned} \qquad (3.36)$$

A Taylor table is simply a convenient way of forming linear combinations of Taylor series on a term by term basis.

Consider the sum of each of these columns. To maximize the order of accuracy of the method, we proceed from left to right and force, by the proper choice of a, b, and c, these sums to be zero. One can easily show that the sums of the first three columns are zero if we satisfy the equation

$$\begin{bmatrix} -1 & -1 & -1 \\ 1 & 0 & -1 \\ -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -2 \end{bmatrix}$$

The solution is given by $[a, b, c] = [1, -2, 1]$.

The columns that do not sum to zero constitute the error.

> We designate the first non-vanishing sum to be er_t, and refer to it as the Taylor series error.

In this case er_t occurs at the fifth column in the table (for this example all even columns will vanish by symmetry) and one finds

$$er_t = \frac{1}{\Delta x^2}\left[\frac{-a}{24} + \frac{-c}{24}\right]\Delta x^4 \left(\frac{\partial^4 u}{\partial x^4}\right)_j = \frac{-\Delta x^2}{12}\left(\frac{\partial^4 u}{\partial x^4}\right)_j \tag{3.37}$$

Note that Δx^2 has been divided through to make the error term consistent. We have just derived the familiar 3-point central-differencing point operator for a second derivative

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_j - \frac{1}{\Delta x^2}(u_{j-1} - 2u_j + u_{j+1}) = O(\Delta x^2) \tag{3.38}$$

The Taylor table for a 3-point backward-differencing operator representing a first derivative is shown in Table 3.2.

$$\left(\frac{\partial u}{\partial x}\right)_j - \frac{1}{\Delta x}(a_2 u_{j-2} + a_1 u_{j-1} + b\, u_j) = \,?$$

	u_j	$\Delta x \cdot \left(\frac{\partial u}{\partial x}\right)_j$	$\Delta x^2 \cdot \left(\frac{\partial^2 u}{\partial x^2}\right)_j$	$\Delta x^3 \cdot \left(\frac{\partial^3 u}{\partial x^3}\right)_j$	$\Delta x^4 \cdot \left(\frac{\partial^4 u}{\partial x^4}\right)_j$
$\Delta x \cdot \left(\frac{\partial u}{\partial x}\right)_j$		1			
$-a_2 \cdot u_{j-2}$	$-a_2$	$-a_2 \cdot (\text{-}2) \cdot \frac{1}{1}$	$-a_2 \cdot (\text{-}2)^2 \cdot \frac{1}{2}$	$-a_2 \cdot (\text{-}2)^3 \cdot \frac{1}{6}$	$-a_2 \cdot (\text{-}2)^4 \cdot \frac{1}{24}$
$-a_1 \cdot u_{j-1}$	$-a_1$	$-a_1 \cdot (\text{-}1) \cdot \frac{1}{1}$	$-a_1 \cdot (\text{-}1)^2 \cdot \frac{1}{2}$	$-a_1 \cdot (\text{-}1)^3 \cdot \frac{1}{6}$	$-a_1 \cdot (\text{-}1)^4 \cdot \frac{1}{24}$
$-b \cdot u_j$	$-b$				

Table 3.2. Taylor table for backward 3-point Lagrangian approximation to a first derivative.

This time the first three columns sum to zero if

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 1 & 0 \\ -4 & -1 & 0 \end{bmatrix} \begin{bmatrix} a_2 \\ a_1 \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}$$

which gives $[a_2, a_1, b] = \frac{1}{2}[1, -4, 3]$. In this case the fourth column provides the leading truncation error term:

$$er_t = \frac{1}{\Delta x}\left[\frac{8a_2}{6} + \frac{a_1}{6}\right]\Delta x^3 \left(\frac{\partial^3 u}{\partial x^3}\right)_j = \frac{\Delta x^2}{3}\left(\frac{\partial^3 u}{\partial x^3}\right)_j \tag{3.39}$$

Thus we have derived a second-order backward-difference approximation of a first derivative:

$$\left(\frac{\partial u}{\partial x}\right)_j - \frac{1}{2\Delta x}(u_{j-2} - 4u_{j-1} + 3u_j) = O(\Delta x^2) \tag{3.40}$$

3.4.2 Generalization of Difference Formulas

In general, a difference approximation to the mth derivative at grid point j can be cast in terms of $q + p + 1$ neighboring points as

$$\left(\frac{\partial^m u}{\partial x^m}\right)_j - \sum_{i=-p}^{q} a_i u_{j+i} = er_t \tag{3.41}$$

where the a_i are coefficients to be determined through the use of Taylor tables to produce approximations of a given order. Clearly this process can be used to find forward, backward, skewed, or central point operators of any order for any derivative. It could be computer automated and extended to higher dimensions. More important, however, is the fact that it can be further generalized. In order to do this, let us approach the subject in a slightly different way, that is from the point of view of interpolation formulas. These formulas are discussed in many texts on numerical analysis.

3.4.3 Lagrange and Hermite Interpolation Polynomials

The Lagrangian interpolation polynomial is given by

$$u(x) = \sum_{k=0}^{K} a_k(x) u_k \tag{3.42}$$

where $a_k(x)$ are polynomials in x of degree K. The construction of the $a_k(x)$ can be taken from the simple Lagrangian formula for quadratic interpolation (or extrapolation) with non-equispaced points

$$\begin{aligned} u(x) = u_0\frac{(x_1-x)(x_2-x)}{(x_1-x_0)(x_2-x_0)} &+ u_1\frac{(x_0-x)(x_2-x)}{(x_0-x_1)(x_2-x_1)} \\ &+ u_2\frac{(x_0-x)(x_1-x)}{(x_0-x_2)(x_1-x_2)} \end{aligned} \tag{3.43}$$

Notice that the coefficient of each u_k is one when $x = x_k$, and zero when x takes any other discrete value in the set. If we take the first or second derivative of $u(x)$, impose an equispaced mesh, and evaluate these derivatives at the appropriate discrete point, we rederive the finite-difference approximations just presented. Finite-difference schemes that can be derived from Eq. 3.42 are referred to as Lagrangian approximations.

A generalization of the Lagrangian approach is brought about by using Hermitian interpolation. To construct a polynomial for $u(x)$, Hermite formulas use values of the function *and its derivative(s)* at given points in space. Our illustration is for the case in which discrete values of the function and its first derivative are used, producing the expression

$$u(x) = \sum a_k(x)u_k + \sum b_k(x)\left(\frac{\partial u}{\partial x}\right)_k \tag{3.44}$$

Obviously higher-order derivatives could be included as the problems dictate. A complete discussion of these polynomials can be found in many references on numerical methods, but here we need only the concept.

The previous examples of a Taylor table constructed *explicit* point difference operators from Lagrangian interpolation formulas. Consider next the Taylor table for an *implicit* space differencing scheme for a first derivative arising from the use of a Hermite interpolation formula. A generalization of Eq. 3.41 can include derivatives at neighboring points, i.e.,

$$\sum_{i=-r}^{s} b_i\left(\frac{\partial^m u}{\partial x^m}\right)_{j+i} - \sum_{i=-p}^{q} a_i u_{j+i} = er_t \tag{3.45}$$

analogous to Eq. 3.44. An example formula is illustrated at the top of Table 3.3. Here not only is the derivative at point j represented, but also included are derivatives at points $j-1$ and $j+1$, which also must be expanded using Taylor series about point

j. This requires the following generalization of the Taylor series expansion given in Eq. 3.8:

$$\left(\frac{\partial^m u}{\partial x^m}\right)_{j+k} = \left\{\left[\sum_{n=0}^{\infty} \frac{1}{n!}(k\Delta x)^n \frac{\partial^n}{\partial x^n}\right]\left(\frac{\partial^m u}{\partial x^m}\right)\right\}_j \tag{3.46}$$

The derivative terms now have coefficients (the coefficient on the j point is taken as one to simplify the algebra) which must be determined using the Taylor table approach as outlined below.

$$d\left(\frac{\partial u}{\partial x}\right)_{j-1} + \left(\frac{\partial u}{\partial x}\right)_j + e\left(\frac{\partial u}{\partial x}\right)_{j+1} - \frac{1}{\Delta x}(au_{j-1} + bu_j + cu_{j+1}) = ?$$

	u_j	$\Delta x \cdot \left(\frac{\partial u}{\partial x}\right)_j$	$\Delta x^2 \cdot \left(\frac{\partial^2 u}{\partial x^2}\right)_j$	$\Delta x^3 \cdot \left(\frac{\partial^3 u}{\partial x^3}\right)_j$	$\Delta x^4 \cdot \left(\frac{\partial^4 u}{\partial x^4}\right)_j$	$\Delta x^5 \cdot \left(\frac{\partial^5 u}{\partial x^5}\right)_j$
$\Delta x \cdot d\left(\frac{\partial u}{\partial x}\right)_{j-1}$		d	$d \cdot (\text{-}1) \cdot \frac{1}{1}$	$d \cdot (\text{-}1)^2 \cdot \frac{1}{2}$	$d \cdot (\text{-}1)^3 \cdot \frac{1}{6}$	$d \cdot (\text{-}1)^4 \cdot \frac{1}{24}$
$\Delta x \cdot \left(\frac{\partial u}{\partial x}\right)_j$		1				
$\Delta x \cdot e\left(\frac{\partial u}{\partial x}\right)_{j+1}$		e	$e \cdot (1) \cdot \frac{1}{1}$	$e \cdot (1)^2 \cdot \frac{1}{2}$	$e \cdot (1)^3 \cdot \frac{1}{6}$	$e \cdot (1)^4 \cdot \frac{1}{24}$
$-a \cdot u_{j-1}$	$-a$	$-a \cdot (\text{-}1) \cdot \frac{1}{1}$	$-a \cdot (\text{-}1)^2 \cdot \frac{1}{2}$	$-a \cdot (\text{-}1)^3 \cdot \frac{1}{6}$	$-a \cdot (\text{-}1)^4 \cdot \frac{1}{24}$	$-a \cdot (\text{-}1)^5 \cdot \frac{1}{120}$
$-b \cdot u_j$	$-b$					
$-c \cdot u_{j+1}$	$-c$	$-c \cdot (1) \cdot \frac{1}{1}$	$-c \cdot (1)^2 \cdot \frac{1}{2}$	$-c \cdot (1)^3 \cdot \frac{1}{6}$	$-c \cdot (1)^4 \cdot \frac{1}{24}$	$-c \cdot (1)^5 \cdot \frac{1}{120}$

Table 3.3. Taylor table for central 3-point Hermitian approximation to a first derivative.

To maximize the order of accuracy, we must satisfy the relation

$$\begin{bmatrix} -1 & -1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 1 & 1 \\ -1 & 0 & -1 & -2 & 2 \\ 1 & 0 & -1 & 3 & 3 \\ -1 & 0 & -1 & -4 & 4 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

having the solution $[a, b, c, d, e] = \frac{1}{4}[-3, 0, 3, 1, 1]$. Under these conditions the sixth column sums to

$$er_t = \frac{\Delta x^4}{120}\left(\frac{\partial^5 u}{\partial x^5}\right)_j \tag{3.47}$$

and the method can be expressed as

$$\left(\frac{\partial u}{\partial x}\right)_{j-1} + 4\left(\frac{\partial u}{\partial x}\right)_j + \left(\frac{\partial u}{\partial x}\right)_{j+1} - \frac{3}{\Delta x}(-u_{j-1} + u_{j+1}) = O(\Delta x^4) \tag{3.48}$$

This is also referred to as a *Padé* formula.

3.4.4 Practical Application of Padé Formulas

It is one thing to construct methods using the Hermitian concept and quite another to implement them in a computer code. In the form of a point operator it is probably not evident at first just how Eq. 3.48 can be applied. However, the situation is quite easy to comprehend if we express the same method in the form of a matrix operator. A banded matrix notation for Eq. 3.48 is

$$\frac{1}{6}B(1,4,1)\delta_x\vec{u} = \frac{1}{2\Delta x}B(-1,0,1)\vec{u} + \left(\vec{bc}\right) \tag{3.49}$$

in which Dirichlet boundary conditions have been imposed.[4] Mathematically this is equivalent to

$$\delta_x\vec{u} = 6[B(1,4,1)]^{-1}\left[\frac{1}{2\Delta x}B(-1,0,1)\vec{u} + \left(\vec{bc}\right)\right] \tag{3.50}$$

which can be reexpressed by the "predictor-corrector" sequence

$$\begin{aligned} \tilde{\vec{u}} &= \frac{1}{2\Delta x}B(-1,0,1)\vec{u} + \left(\vec{bc}\right) \\ \delta_x\vec{u} &= 6\,[B(1,4,1)]^{-1}\tilde{\vec{u}} \end{aligned} \tag{3.51}$$

With respect to practical implementation, the meaning of the predictor in this sequence should be clear. It simply says – take the vector array $\vec{u}$, difference it, add the boundary conditions, and store the result in the intermediate array $\tilde{\vec{u}}$. The meaning of the second row is more subtle, since it is demanding the evaluation of an inverse operator, but it still can be given a simple interpretation. An inverse matrix operator implies the solution of a coupled set of linear equations. These operators are very common in finite difference applications. They appear in the form of banded matrices having a small bandwidth, in this case a tridiagonal. The evaluation of $[B(1,4,1)]^{-1}$ is found by means of a tridiagonal "solver", which is simple to code, efficient to run, and widely used. In general, Hermitian or Padé approximations can be practical when they can be implemented by the use of efficient banded solvers.

[4] In this case the vector containing the boundary conditions would include values of both u and $\partial u/\partial x$ at both boundaries.

3.4.5 Other Higher-Order Schemes

Hermitian forms of the second derivative can also be easily derived by means of a Taylor table. For example

$$\delta_{xx}\vec{u} = 12\left[B(1,10,1)\right]^{-1}\left[\frac{1}{\Delta x^2}B(1,-2,1)\vec{u} + \left(\vec{bc}\right)\right] \tag{3.52}$$

is $O(\Delta x^4)$ and makes use of only tridiagonal operations. It should be mentioned that the spline approximation is one form of a Padé matrix difference operator. It is given by

$$\delta_{xx}\vec{u} = 6\left[B(1,4,1)\right]^{-1}\left[\frac{1}{\Delta x^2}B(1,-2,1)\vec{u} + \left(\vec{bc}\right)\right] \tag{3.53}$$

but its order of accuracy is only $O(\Delta x^2)$. How much this reduction in accuracy is offset by the increased global continuity built into a spline fit is not known. We note that the spline fit of a first derivative is identical to any of the expressions in Eqs. 3.48 to 3.51.

A final word on Hermitian approximations. Clearly they have an advantage over 3-point Lagrangian schemes because of their increased accuracy. However, a more subtle point is that they get this increase in accuracy using information that is still local to the point where the derivatives are being evaluated. In application, this can be advantageous at boundaries and in the vicinity of steep gradients. It is obvious, of course, that five point schemes using Lagrangian approximations can be derived that have the same order of accuracy as the methods given in Eqs. 3.48 and 3.52, but they will have a wider spread of space indices. In particular, two Lagrangian schemes with the same order of accuracy are (here we ignore the problem created by the boundary conditions, although this is one of the principal issues in applying these schemes):

$$\frac{\partial\vec{u}}{\partial x} - \frac{1}{12\Delta x}B_p(1,-8,0,8,-1)\vec{u} = O\left(\Delta x^4\right) \tag{3.54}$$

$$\frac{\partial^2\vec{u}}{\partial x^2} - \frac{1}{12\Delta x^2}B_p(-1,16,-30,16,-1)\vec{u} = O\left(\Delta x^4\right) \tag{3.55}$$

3.5 Fourier Error Analysis

In order to select a finite-difference scheme for a given application one must be able to assess the accuracy of the candidate schemes. The accuracy of an operator is often expressed in terms of the order of the leading error term determined from a Taylor table. While this is a useful measure, it provides a fairly limited description. Further information about the error behavior of a finite-difference scheme can be obtained using Fourier error analysis.

3.5.1 Application to a Spatial Operator

An arbitrary periodic function can be decomposed into its Fourier components, which are in the form $e^{i\kappa x}$, where κ is the wavenumber. It is therefore of interest to examine how well a given finite-difference operator approximates derivatives of $e^{i\kappa x}$. We will concentrate here on first derivative approximations, although the analysis is equally applicable to higher derivatives.

The exact first derivative of $e^{i\kappa x}$ is

$$\frac{\partial e^{i\kappa x}}{\partial x} = i\kappa e^{i\kappa x} \tag{3.56}$$

If we apply, for example, a second-order centered difference operator to $u_j = e^{i\kappa x_j}$, where $x_j = j\Delta x$, we get

$$\begin{aligned}
(\delta_x u)_j &= \frac{u_{j+1} - u_{j-1}}{2\Delta x} \\
&= \frac{e^{i\kappa\Delta x(j+1)} - e^{i\kappa\Delta x(j-1)}}{2\Delta x} \\
&= \frac{(e^{i\kappa\Delta x} - e^{-i\kappa\Delta x})e^{i\kappa x_j}}{2\Delta x} \\
&= \frac{1}{2\Delta x}[(\cos\kappa\Delta x + i\sin\kappa\Delta x) - (\cos\kappa\Delta x - i\sin\kappa\Delta x)]e^{i\kappa x_j} \\
&= i\frac{\sin\kappa\Delta x}{\Delta x}e^{i\kappa x_j} \\
&= i\kappa^* e^{i\kappa x_j}
\end{aligned} \tag{3.57}$$

where κ^* is the modified wavenumber. The modified wavenumber is so named because it appears where the wavenumber, κ, appears in the exact expression. Thus the degree to which the modified wavenumber approximates the actual wavenumber is a measure of the accuracy of the approximation.

For the second-order centered difference operator the modified wavenumber is given by

$$\kappa^* = \frac{\sin\kappa\Delta x}{\Delta x} \tag{3.58}$$

Note that κ^* approximates κ to second-order accuracy, as is to be expected, since

$$\frac{\sin\kappa\Delta x}{\Delta x} = \kappa - \frac{\kappa^3\Delta x^2}{6} + \ldots$$

Equation 3.58 is plotted in Figure 3.4, along with similar relations for the standard fourth-order centered difference scheme and the fourth-order Padé scheme. The

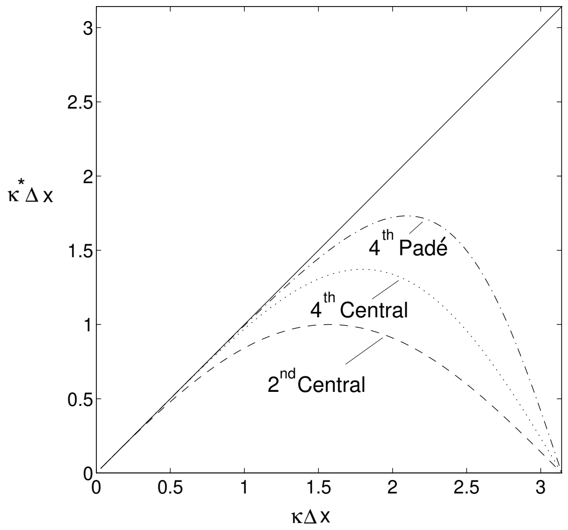

Figure 3.4: Modified wavenumber for various schemes.

expression for the modified wavenumber provides the accuracy with which a given wavenumber component of the solution is resolved for the entire wavenumber range available in a mesh of a given size, $0 \leq \kappa\Delta x \leq \pi$.

In general, finite-difference operators can be written in the form

$$(\delta_x)_j = (\delta_x^a)_j + (\delta_x^s)_j$$

where $(\delta_x^a)_j$ is an antisymmetric operator and $(\delta_x^s)_j$ is a symmetric operator.[5] If we restrict our interest to schemes extending from $j-3$ to $j+3$, then

$$(\delta_x^a u)_j = \frac{1}{\Delta x}[a_1(u_{j+1} - u_{j-1}) + a_2(u_{j+2} - u_{j-2}) + a_3(u_{j+3} - u_{j-3})]$$

and

$$(\delta_x^s u)_j = \frac{1}{\Delta x}[d_0 u_j + d_1(u_{j+1} + u_{j-1}) + d_2(u_{j+2} + u_{j-2}) + d_3(u_{j+3} + u_{j-3})]$$

The corresponding modified wavenumber is

$$\begin{aligned} i\kappa^* &= \frac{1}{\Delta x}[d_0 + 2(d_1 \cos\kappa\Delta x + d_2 \cos 2\kappa\Delta x + d_3 \cos 3\kappa\Delta x) \\ &+ 2i(a_1 \sin\kappa\Delta x + a_2 \sin 2\kappa\Delta x + a_3 \sin 3\kappa\Delta x)] \end{aligned} \tag{3.59}$$

[5] In terms of a circulant matrix operator A, the antisymmetric part is obtained from $(A - A^T)/2$ and the symmetric part from $(A + A^T)/2$.

When the finite-difference operator is antisymmetric (centered), the modified wavenumber is purely real. When the operator includes a symmetric component, the modified wavenumber is complex, with the imaginary component being entirely error. The fourth-order Padé scheme is given by

$$(\delta_x u)_{j-1} + 4(\delta_x u)_j + (\delta_x u)_{j+1} = \frac{3}{\Delta x}(u_{j+1} - u_{j-1})$$

The modified wavenumber for this scheme satisfies[6]

$$i\kappa^* e^{-i\kappa\Delta x} + 4i\kappa^* + i\kappa^* e^{i\kappa\Delta x} = \frac{3}{\Delta x}(e^{i\kappa\Delta x} - e^{-i\kappa\Delta x})$$

which gives

$$i\kappa^* = \frac{3i \sin \kappa\Delta x}{(2 + \cos \kappa\Delta x)\Delta x}$$

The modified wavenumber provides a useful tool for assessing difference approximations. In the context of the linear convection equation, the errors can be given a physical interpretation. Consider once again the linear convection equation in the form

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

on a domain extending from $-\infty$ to ∞. Recall from Section 2.3.2 that a solution initiated by a harmonic function with wavenumber κ is

$$u(x,t) = f(t)e^{i\kappa x} \tag{3.60}$$

where $f(t)$ satisfies the ODE

$$\frac{df}{dt} = -ia\kappa f$$

Solving for $f(t)$ and substituting into Eq. 3.60 gives the exact solution as

$$u(x,t) = f(0)e^{i\kappa(x-at)}$$

If second-order centered differences are applied to the spatial term, the following ODE is obtained for $f(t)$:

$$\frac{df}{dt} = -ia\left[\frac{\sin \kappa\Delta x}{\Delta x}\right] f = -ia\kappa^* f \tag{3.61}$$

[6]Note that terms such as $(\delta_x u)_{j-1}$ are handled by letting $(\delta_x u)_j = ik^* e^{i\kappa j\Delta x}$ and evaluating the shift in j.

Solving this ODE exactly (since we are considering the error from the spatial approximation only) and substituting into Eq. 3.60, we obtain

$$u_{\text{numerical}}(x,t) = f(0)e^{i\kappa(x-a^*t)} \tag{3.62}$$

where a^* is the numerical (or modified) phase speed, which is related to the modified wavenumber by

$$\frac{a^*}{a} = \frac{\kappa^*}{\kappa}$$

For the above example,

$$\frac{a^*}{a} = \frac{\sin \kappa\Delta x}{\kappa\Delta x}$$

The numerical phase speed is the speed at which a harmonic function is propagated numerically. Since $a^*/a \le 1$ for this example, the numerical solution propagates too slowly. Since a^* is a function of the wavenumber, the numerical approximation introduces dispersion, although the original PDE is nondispersive. As a result, a waveform consisting of many different wavenumber components eventually loses its original form.

Figure 3.5 shows the numerical phase speed for the schemes considered previously. The number of *points per wavelength* (PPW) by which a given wave is resolved is given by $2\pi/\kappa\Delta x$. The resolving efficiency of a scheme can be expressed in terms of the PPW required to produce errors below a specified level. For example, the second-order centered difference scheme requires 80 PPW to produce an error in phase speed of less than 0.1 percent. The 5-point fourth-order centered scheme and the fourth-order Padé scheme require 15 and 10 PPW respectively to achieve the same error level.

For our example using second-order centered differences, the modified wavenumber is purely real, but in the general case it can include an imaginary component as well, as shown in Eq. 3.59. In that case, the error in the phase speed is determined from the real part of the modified wavenumber, while the imaginary part leads to an error in the amplitude of the solution, as can be seen by inspecting Eq. 3.61. Thus the antisymmetric portion of the spatial difference operator determines the error in speed and the symmetric portion the error in amplitude. This will be further discussed in Section 11.2.

3.6 Difference Operators at Boundaries

As discussed in Section 3.3.2, a matrix difference operator incorporates both the difference approximation in the interior of the domain and that at the boundaries. In this section, we consider the boundary operators needed for our model equations

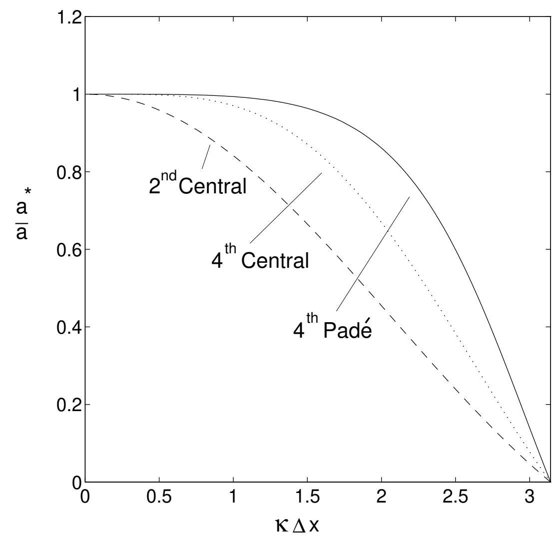

Figure 3.5: Numerical phase speed for various schemes.

for convection and diffusion. In the case of periodic boundary conditions, no special boundary operators are required.

3.6.1 The Linear Convection Equation

Referring to Section 2.3, the boundary conditions for the linear convection equation can be either periodic or of inflow-outflow type. In the latter case, a Dirichlet boundary condition is given at the inflow boundary, while no condition is specified at the outflow boundary. Here we assume that the wave speed a is positive; thus the left-hand boundary is the inflow boundary, and the right-hand boundary is the outflow boundary. Thus the vector of unknowns is

$$\vec{u} = [u_1,\ u_2,\ \ldots,\ u_M]^T \tag{3.63}$$

and u_0 is specified.

Consider first the inflow boundary. It is clear that as long as the interior difference approximation does not extend beyond u_{j-1}, then no special treatment is required for this boundary. For example, with second-order centered differences we obtain

$$\delta_x \vec{u} = A\vec{u} + \left(\vec{bc}\right) \tag{3.64}$$

with

$$A = \frac{1}{2\Delta x}\begin{bmatrix} 0 & 1 & & & \\ -1 & 0 & 1 & & \\ & -1 & 0 & 1 & \\ & & & \ddots & \end{bmatrix}, \qquad \left(\vec{bc}\right) = \frac{1}{2\Delta x}\begin{bmatrix} -u_0 \\ 0 \\ 0 \\ \vdots \end{bmatrix} \tag{3.65}$$

However, if we use the fourth-order interior operator given in Eq. 3.54, then the approximation at $j = 1$ requires a value of u_{j-2}, which is outside the domain. Hence, a different operator is required at $j = 1$ which extends only to $j - 1$, while having the appropriate order of accuracy. Such an operator, known as a *numerical boundary scheme*, can have an order of accuracy which is *one order lower* than that of the interior scheme, and the global accuracy will equal that of the interior scheme.[7] For example, with fourth-order centered differences, we can use the following third-order operator at $j = 1$:

$$(\delta_x u)_1 = \frac{1}{6\Delta x}(-2u_0 - 3u_1 + 6u_2 - u_3) \tag{3.66}$$

which is easily derived using a Taylor table. The resulting difference operator has the form of Eq. 3.64 with

$$A = \frac{1}{12\Delta x}\begin{bmatrix} -6 & 12 & -2 & & & \\ -8 & 0 & 8 & -1 & & \\ 1 & -8 & 0 & 8 & -1 & \\ & & & \ddots & & \end{bmatrix}, \qquad \left(\vec{bc}\right) = \frac{1}{12\Delta x}\begin{bmatrix} -4u_0 \\ u_0 \\ 0 \\ \vdots \end{bmatrix} \tag{3.67}$$

This approximation is globally fourth-order accurate.

At the outflow boundary, no boundary condition is specified. We must approximate $\partial u/\partial x$ at node M with no information about u_{M+1}. Thus the second-order centered-difference operator, which requires u_{j+1}, cannot be used at $j = M$. A backward-difference formula must be used. With a second-order interior operator, the following first-order backward formula can be used:

$$(\delta_x u)_M = \frac{1}{\Delta x}(u_M - u_{M-1}) \tag{3.68}$$

[7]Proof of this theorem is beyond the scope of this book; the interested reader should consult the literature for further details.

This produces a difference operator with

$$A = \frac{1}{2\Delta x}\begin{bmatrix} 0 & 1 & & & & \\ -1 & 0 & 1 & & & \\ & -1 & 0 & 1 & & \\ & & & \ddots & & \\ & & & -1 & 0 & 1 \\ & & & & -2 & 2 \end{bmatrix}, \quad (\vec{bc}) = \frac{1}{2\Delta x}\begin{bmatrix} -u_0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.69}$$

In the case of a fourth-order centered interior operator the last two rows of A require modification.

Another approach to the development of boundary schemes is in terms of space extrapolation. The following formula allows u_{M+1} to be extrapolated from the interior data to arbitrary order on an equispaced grid:

$$(1 - E^{-1})^p u_{M+1} = 0 \tag{3.70}$$

where E is the shift operator defined by $Eu_j = u_{j+1}$ and the order of the approximation is $p - 1$. For example, with $p = 2$ we obtain

$$(1 - 2E^{-1} + E^{-2})u_{M+1} = u_{M+1} - 2u_M + u_{M-1} = 0 \tag{3.71}$$

which gives the following first-order approximation to u_{M+1}:

$$u_{M+1} = 2u_M - u_{M-1} \tag{3.72}$$

Substituting this into the second-order centered-difference operator applied at node M gives

$$\begin{aligned} (\delta_x u)_M = \frac{1}{2\Delta x}(u_{M+1} - u_{M-1}) &= \frac{1}{2\Delta x}(2u_M - u_{M-1} - u_{M-1}) \\ &= \frac{1}{\Delta x}(u_M - u_{M-1}) \end{aligned} \tag{3.73}$$

which is identical to Eq. 3.68.

3.6.2 The Diffusion Equation

In solving the diffusion equation, we must consider Dirichlet and Neumann boundary conditions. The treatment of a Dirichlet boundary condition proceeds along the same lines as the inflow boundary for the convection equation discussed above. With the second-order centered interior operator

$$(\delta_{xx} u)_j = \frac{1}{\Delta x^2}(u_{j+1} - 2u_j + u_{j-1}) \tag{3.74}$$

no modifications are required near boundaries, leading to the matrix difference operator given in Eq. 3.27.

For a Neumann boundary condition, we assume that $\partial u/\partial x$ is specified at $j = M + 1$, that is

$$\left(\frac{\partial u}{\partial x}\right)_{M+1} = \left(\frac{\partial u}{\partial x}\right)_b \tag{3.75}$$

Thus we design an operator at node M which is in the following form:

$$(\delta_{xx}u)_M = \frac{1}{\Delta x^2}(au_{M-1} + bu_M) + \frac{c}{\Delta x}\left(\frac{\partial u}{\partial x}\right)_{M+1} \tag{3.76}$$

where a, b, and c are constants which can easily be determined using a Taylor table, as shown in Table 3.4.

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_j - \left[\frac{1}{\Delta x^2}(au_{j-1} + bu_j) + \frac{c}{\Delta x}\left(\frac{\partial u}{\partial x}\right)_{j+1}\right] = ?$$

	u_j	$\Delta x \cdot \left(\frac{\partial u}{\partial x}\right)_j$	$\Delta x^2 \cdot \left(\frac{\partial^2 u}{\partial x^2}\right)_j$	$\Delta x^3 \cdot \left(\frac{\partial^3 u}{\partial x^3}\right)_j$	$\Delta x^4 \cdot \left(\frac{\partial^4 u}{\partial x^4}\right)_j$
$\Delta x^2 \cdot \left(\frac{\partial^2 u}{\partial x^2}\right)_j$			1		
$-a \cdot u_{j-1}$	$-a$	$-a \cdot (\text{-}1) \cdot \frac{1}{1}$	$-a \cdot (\text{-}1)^2 \cdot \frac{1}{2}$	$-a \cdot (\text{-}1)^3 \cdot \frac{1}{6}$	$-a \cdot (\text{-}1)^4 \cdot \frac{1}{24}$
$-b \cdot u_j$	$-b$				
$-\Delta x \cdot c \cdot \left(\frac{\partial u}{\partial x}\right)_{j+1}$		$-c$	$-c \cdot (1) \cdot \frac{1}{1}$	$-c \cdot (1)^2 \cdot \frac{1}{2}$	$-c \cdot (1)^3 \cdot \frac{1}{6}$

Table 3.4. Taylor table for Neumann boundary condition.

Solving for a, b, and c, we obtain the following first-order operator:

$$(\delta_{xx}u)_M = \frac{1}{3\Delta x^2}(2u_{M-1} - 2u_M) + \frac{2}{3\Delta x}\left(\frac{\partial u}{\partial x}\right)_{M+1} \tag{3.77}$$

which produces the matrix difference operator given in Eq. 3.29.

We can also obtain the operator in Eq. 3.77 using the space extrapolation idea. Consider a second-order backward-difference approximation applied at node $M + 1$:

$$\left(\frac{\partial u}{\partial x}\right)_{M+1} = \frac{1}{2\Delta x}(u_{M-1} - 4u_M + 3u_{M+1}) + O(\Delta x^2) \tag{3.78}$$

Solving for u_{M+1} gives

$$u_{M+1} = \frac{1}{3}\left[4u_M - u_{M-1} + 2\Delta x\left(\frac{\partial u}{\partial x}\right)_{M+1}\right] + O(\Delta x^3) \tag{3.79}$$

Substituting this into the second-order centered difference operator for a second derivative applied at node M gives

$$\begin{aligned}
(\delta_{xx}u)_M &= \frac{1}{\Delta x^2}(u_{M+1} - 2u_M + u_{M-1}) && (3.80)\\
&= \frac{1}{3\Delta x^2}\left[3u_{M-1} - 6u_M + 4u_M - u_{M-1} + 2\Delta x\left(\frac{\partial u}{\partial x}\right)_{M+1}\right] &&\\
&= \frac{1}{3\Delta x^2}(2u_{M-1} - 2u_M) + \frac{2}{3\Delta x}\left(\frac{\partial u}{\partial x}\right)_{M+1} && (3.81)
\end{aligned}$$

which is identical to Eq. 3.77.

3.7 Problems

1. Derive a third-order finite-difference approximation to a first derivative in the form

$$(\delta_x u)_j = \frac{1}{\Delta x}(au_{j-2} + bu_{j-1} + cu_j + du_{j+1})$$

   Find the leading error term.

2. Derive a finite-difference approximation to a first derivative in the form

$$a(\delta_x u)_{j-1} + (\delta_x u)_j = \frac{1}{\Delta x}(bu_{j-1} + cu_j + du_{j+1})$$

   Find the leading error term.

3. Using a 4 (interior) point mesh, write out the 4×4 matrices and the boundary-condition vector formed by using the scheme derived in question 2 when both u and $\partial u/\partial x$ are given at $j = 0$ and u is given at $j = 5$.

4. Repeat question 2 with $d = 0$.

5. Derive a finite-difference approximation to a third derivative in the form

$$(\delta_{xxx}u)_j = \frac{1}{\Delta x^3}(au_{j-2} + bu_{j-1} + cu_j + du_{j+1} + eu_{j+2})$$

   Find the leading error term.

6. Derive a compact (or Padé) finite-difference approximation to a second derivative in the form

$$d(\delta_{xx}u)_{j-1} + (\delta_{xx}u)_j + e(\delta_{xx}u)_{j+1} = \frac{1}{\Delta x^2}(au_{j-1} + bu_j + cu_{j+1})$$

   Find the leading error term.

7. Find the modified wavenumber for the operator derived in question 1. Plot the real and imaginary parts of $\kappa^* \Delta x$ vs. $\kappa \Delta x$ for $0 \le \kappa \Delta x \le \pi$. Compare the real part with that obtained from the fourth-order centered operator (Eq. 3.54).

8. Application of the second-derivative operator to the function $e^{i\kappa x}$ gives

$$\frac{\partial^2 e^{i\kappa x}}{\partial x^2} = -\kappa^2 e^{i\kappa x}$$

   Application of a difference operator for the second derivative gives

$$(\delta_{xx} e^{i\kappa j \Delta x})_j = -\kappa^{*2} e^{i\kappa x}$$

   thus defining the modified wavenumber κ^* for a second derivative approximation. Find the modified wavenumber for the second-order centered difference operator for a second derivative, the noncompact fourth-order operator (Eq. 3.55), and the compact fourth-order operator derived in question 6. Plot $(\kappa^* \Delta x)^2$ vs. $\kappa \Delta x$ for $0 \le \kappa \Delta x \le \pi$.

9. Find the grid-points-per-wavelength (PPW) requirement to achieve a phase speed error less than 0.1 percent for sixth-order noncompact and compact centered approximations to a first derivative.

10. Consider the following one-sided differencing schemes, which are first-, second-, and third-order, respectively:

$$(\delta_x u)_j = (u_j - u_{j-1}) / \Delta x$$

$$(\delta_x u)_j = (3u_j - 4u_{j-1} + u_{j-2}) / (2\Delta x)$$

$$(\delta_x u)_j = (11u_j - 18u_{j-1} + 9u_{j-2} - 2u_{j-3}) / (6\Delta x)$$

   Find the modified wavenumber for each of these schemes. Plot the real and imaginary parts of $\kappa^* \Delta x$ vs. $\kappa \Delta x$ for $0 \le \kappa \Delta x \le \pi$. Derive the two leading terms in the truncation error for each scheme.

Chapter 4

THE SEMI-DISCRETE APPROACH

One strategy for obtaining finite-difference approximations to a PDE is to start by differencing the space derivatives only, without approximating the time derivative. In the following chapters, we proceed with an analysis making considerable use of this concept, which we refer to as the *semi-discrete* approach. Differencing the space derivatives converts the basic PDE into a set of coupled ODE's. In the most general notation, these ODE's would be expressed in the form

$$\frac{d\vec{u}}{dt} = \vec{F}(\vec{u}, t) \tag{4.1}$$

which includes all manner of nonlinear and time-dependent possibilities. On occasion, we use this form, but the rest of this chapter is devoted to a more specialized matrix notation described below.

Another strategy for constructing a finite-difference approximation to a PDE is to approximate all the partial derivatives at once. This generally leads to a point difference operator (see Section 3.3.1) which, in turn, can be used for the time advance of the solution at any given point in the mesh. As an example let us consider the model equation for diffusion

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2}$$

Using three-point central-differencing schemes for both the time and space derivatives, we find

$$\frac{u_j^{(n+1)} - u_j^{(n-1)}}{2h} = \nu \left[\frac{u_{j+1}^{(n)} - 2u_j^{(n)} + u_{j-1}^{(n)}}{\Delta x^2} \right]$$

or

$$u_j^{(n+1)} = u_j^{(n-1)} + \frac{2h\nu}{\Delta x^2} \left[u_{j+1}^{(n)} - 2u_j^{(n)} + u_{j-1}^{(n)} \right] \tag{4.2}$$

Clearly Eq. 4.2 is a difference equation which can be used at the space point j to advance the value of u from the previous time levels n and $n-1$ to the level $n+1$. It is a full discretization of the PDE. Note, however, that the spatial and temporal discretizations are separable. Thus, this method has an intermediate semi-discrete form and can be analyzed by the methods discussed in the next few chapters.

Another possibility is to replace the value of $u_j^{(n)}$ in the right hand side of Eq. 4.2 with the time average of u at that point, namely $(u_j^{(n+1)} + u_j^{(n-1)})/2$. This results in the formula

$$u_j^{(n+1)} = u_j^{(n-1)} + \frac{2h\nu}{\Delta x^2}\left[u_{j+1}^{(n)} - 2\left(\frac{u_j^{(n+1)} + u_j^{(n-1)}}{2}\right) + u_{j-1}^{(n)}\right] \tag{4.3}$$

which can be solved for $u^{(n+1)}$ and time advanced at the point j. In this case, the spatial and temporal discretizations are not separable, and no semi-discrete form exists.

Equation 4.2 is sometimes called Richardson's method of overlapping steps and Eq. 4.3 is referred to as the DuFort-Frankel method. As we shall see later on, there are subtle points to be made about using these methods to find a numerical solution to the diffusion equation. There are a number of issues concerning the accuracy, stability, and convergence of Eqs. 4.2 and 4.3 which we cannot comment on until we develop a framework for such investigations. We introduce these methods here only to distinguish between methods in which the temporal and spatial terms are discretized separately and those for which no such separation is possible. For the time being, we shall separate the space difference approximations from the time differencing. In this approach, we reduce the governing PDE's to ODE's by discretizing the spatial terms and use the well-developed theory of ODE solutions to aid us in the development of an analysis of accuracy and stability.

4.1 Reduction of PDE's to ODE's

4.1.1 The Model ODE's

First let us consider the model PDE's for diffusion and biconvection described in Chapter 2. In these simple cases, we can approximate the space derivatives with difference operators and express the resulting ODE's with a matrix formulation. This is a simple and natural formulation when the ODE's are linear.

Model ODE for Diffusion

For example, using the 3-point central-differencing scheme to represent the second derivative in the scalar PDE governing diffusion leads to the following ODE diffusion model

$$\frac{d\vec{u}}{dt} = \frac{\nu}{\Delta x^2} B(1, -2, 1)\vec{u} + (\vec{bc}) \tag{4.4}$$

with Dirichlet boundary conditions folded into the $(\vec{bc})$ vector.

Model ODE for Biconvection

The term biconvection was introduced in Section 2.3. It is used for the scalar convection model when the boundary conditions are periodic. In this case, the 3-point central-differencing approximation produces the ODE model given by

$$\frac{d\vec{u}}{dt} = -\frac{a}{2\Delta x} B_p(-1, 0, 1)\vec{u} \tag{4.5}$$

where the boundary condition vector is absent because the flow is periodic.

Eqs. 4.4 and 4.5 are the model ODE's for diffusion and biconvection of a scalar in one dimension. They are linear with coefficient matrices which are independent of x and t.

4.1.2 The Generic Matrix Form

The generic matrix form of a semi-discrete approximation is expressed by the equation

$$\boxed{\frac{d\vec{u}}{dt} = A\vec{u} - \vec{f}(t)} \tag{4.6}$$

Note that the elements in the matrix A depend upon both the PDE and the type of differencing scheme chosen for the space terms. The vector $\vec{f}(t)$ is usually determined by the boundary conditions and possibly source terms. In general, even the Euler and Navier-Stokes equations can be expressed in the form of Eq. 4.6. In such cases the equations are nonlinear, that is, the elements of A depend on the solution $\vec{u}$ and are usually derived by finding the Jacobian of a flux vector. Although the equations are nonlinear, the linear analysis presented in this book leads to diagnostics that are surprisingly accurate when used to evaluate many aspects of numerical methods as they apply to the Euler and Navier-Stokes equations.

4.2 Exact Solutions of Linear ODE's

In order to advance Eq. 4.1 in time, the system of ODE's must be integrated using a time-marching method. In order to analyze time-marching methods, we will make use of exact solutions of coupled systems of ODE's, which exist under certain conditions. The ODE's represented by Eq. 4.1 are said to be *linear* if F is linearly dependent on u (i.e., if $\partial F/\partial u = A$ where A is independent of u). As we have already pointed out, when the ODE's are linear they can be expressed in a matrix notation as Eq. 4.6 in which the coefficient matrix, A, is independent of u. If A *does* depend explicitly on t, the general solution *cannot* be written; whereas, if A *does not* depend explicitly on t, the general solution to Eq. 4.6 *can* be written. This holds regardless of whether or not the forcing function, $\vec{f}$, depends explicitly on t.

As we shall soon see, the exact solution of Eq. 4.6 can be written in terms of the eigenvalues and eigenvectors of A. This will lead us to a representative scalar equation for use in analyzing time-marching methods. These ideas are developed in the following sections.

4.2.1 Eigensystems of Semi-Discrete Linear Forms

Complete Systems

An $M \times M$ matrix is represented by a *complete eigensystem* if it has a complete set of linearly independent eigenvectors (see Appendix A) . An eigenvector, $\vec{x}_m$, and its corresponding eigenvalue, λ_m, have the property that

$$\begin{aligned} A\vec{x}_m &= \lambda_m \vec{x}_m \qquad \text{or} \\ [\mathrm{A} - \lambda_m \mathrm{I}]\vec{x}_m &= 0 \end{aligned} \tag{4.7}$$

The eigenvalues are the roots of the equation

$$\det[\mathrm{A} - \lambda \mathrm{I}] = 0$$

We form the right-hand eigenvector matrix of a complete system by filling its columns with the eigenvectors $\vec{x}_m$:

$$X = \left[\vec{x}_1\ ,\ \vec{x}_2\ \ldots\ ,\vec{x}_M\right]$$

The inverse is the left-hand eigenvector matrix, and together they have the property that

$$X^{-1}AX = \Lambda \tag{4.8}$$

where Λ is a diagonal matrix whose elements are the eigenvalues of A.

Defective Systems

If an $M \times M$ matrix does *not* have a complete set of linearly independent eigenvectors, it cannot be transformed to a diagonal matrix of scalars, and it is said to be defective. It can, however, be transformed to a diagonal set of *blocks*, some of which may be scalars (see Appendix A). In general, there exists some S which transforms any matrix A such that

$$S^{-1}AS = J$$

where

$$J = \begin{bmatrix} J_1 & & & & \\ & J_2 & & & \\ & & \ddots & & \\ & & & J_m & \\ & & & & \ddots \end{bmatrix}$$

and

$$J_m^{(n)} = \begin{array}{cc} \begin{bmatrix} \lambda_m & 1 & & \\ & \lambda_m & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_m \end{bmatrix} & \begin{matrix} 1 \\ \vdots \\ \vdots \\ n \end{matrix} \end{array}$$

The matrix J is said to be in *Jordan canonical form*, and an eigenvalue with multiplicity n within a Jordan block is said to be a defective eigenvalue. Defective systems play a role in numerical stability analysis.

4.2.2 Single ODE's of First- and Second-Order

First-Order Equations

The simplest nonhomogeneous ODE of interest is given by the single, first-order equation

$$\frac{du}{dt} = \lambda u + ae^{\mu t} \tag{4.9}$$

where λ, a, and μ are scalars, all of which can be complex numbers. The equation is linear because λ does not depend on u, and has a general solution because λ does not depend on t. It has a steady-state solution if the right-hand side is independent of t, i.e., if $\mu = 0$, and is homogeneous if the forcing function is zero, i.e., if $a = 0$. Although it is quite simple, the numerical analysis of Eq. 4.9 displays many of the fundamental properties and issues involved in the construction and study of most popular time-marching methods. This theme will be developed as we proceed.

The exact solution of Eq. 4.9 is, for $\mu \neq \lambda$,

$$u(t) = c_1 e^{\lambda t} + \frac{ae^{\mu t}}{\mu - \lambda}$$

where c_1 is a constant determined by the initial conditions. In terms of the initial value of u, it can be written

$$u(t) = u(0)e^{\lambda t} + a\frac{e^{\mu t} - e^{\lambda t}}{\mu - \lambda}$$

The interesting question can arise: What happens to the solution of Eq. 4.9 when $\mu = \lambda$? This is easily found by setting $\mu = \lambda + \epsilon$, solving, and then taking the limit as $\epsilon \to 0$. Using this limiting device, we find that the solution to

$$\frac{du}{dt} = \lambda u + ae^{\lambda t} \tag{4.10}$$

is given by

$$u(t) = [u(0) + at]e^{\lambda t}$$

As we shall soon see, this solution is required for the analysis of defective systems.

Second-Order Equations

The homogeneous form of a second-order equation is given by

$$\frac{d^2u}{dt^2} + a_1\frac{du}{dt} + a_0 u = 0 \tag{4.11}$$

where a_1 and a_0 are complex constants. Now we introduce the differential operator D such that

$$\boxed{D \equiv \frac{d}{dt}}$$

and factor $u(t)$ out of Eq. 4.11, giving

$$(D^2 + a_1 D + a_0)\, u(t) = 0$$

The polynomial in D is referred to as a *characteristic polynomial* and designated $P(D)$. Characteristic polynomials are fundamental to the analysis of both ODE's and OΔE's, since the roots of these polynomials determine the solutions of the equations. For ODE's, we often label these roots in order of increasing modulus as λ_1, λ_2, $\cdots$, λ_m,

$\cdots, \lambda_M$. They are found by solving the equation $P(\lambda) = 0$. In our simple example, there would be two roots, λ_1 and λ_2, determined from

$$P(\lambda) = \lambda^2 + a_1\lambda + a_0 = 0 \tag{4.12}$$

and the solution to Eq. 4.11 is given by

$$u(t) = c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t} \tag{4.13}$$

where c_1 and c_2 are constants determined from initial conditions. The proof of this is simple and is found by substituting Eq. 4.13 into Eq. 4.11. One finds the result

$$c_1 e^{\lambda_1 t}(\lambda_1^2 + a_1\lambda_1 + a_0) + c_2 e^{\lambda_2 t}(\lambda_2^2 + a_1\lambda_2 + a_0)$$

which is identically zero for all c_1, c_2, and t if and only if the λ's satisfy Eq. 4.12.

4.2.3 Coupled First-Order ODE's

A Complete System

A set of coupled, first-order, homogeneous equations is given by

$$\begin{aligned} u_1' &= a_{11}\, u_1 + a_{12}\, u_2 \\ u_2' &= a_{21}\, u_1 + a_{22}\, u_2 \end{aligned} \tag{4.14}$$

which can be written

$$\vec{u}\,' = A\vec{u} \quad , \quad \vec{u} = [u_1, u_2]^T \quad , \quad A = (a_{ij}) = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

Consider the possibility that a solution is represented by

$$\begin{aligned} u_1 &= c_1\, x_{11} e^{\lambda_1 t} + c_2\, x_{12} e^{\lambda_2 t} \\ u_2 &= c_1\, x_{21} e^{\lambda_1 t} + c_2\, x_{22} e^{\lambda_2 t} \end{aligned} \tag{4.15}$$

By substitution, these are indeed solutions to Eq. 4.14 if and only if

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{21} \end{bmatrix} = \lambda_1 \begin{bmatrix} x_{11} \\ x_{21} \end{bmatrix} \quad , \quad \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_{12} \\ x_{22} \end{bmatrix} = \lambda_2 \begin{bmatrix} x_{12} \\ x_{22} \end{bmatrix} \tag{4.16}$$

Notice that a higher-order equation can be reduced to a coupled set of first-order equations by introducing a new set of dependent variables. Thus, by setting

$$u_1 = u' \quad , \quad u_2 = u \quad ,$$

we find Eq. 4.11 can be written

$$\begin{aligned} u_1' &= -a_1 u_1 - a_0 u_2 \\ u_2' &= u_1 \end{aligned} \tag{4.17}$$

which is a subset of Eq. 4.14.

A Derogatory System

Eq. 4.15 is still a solution to Eq. 4.14 if $\lambda_1 = \lambda_2 = \lambda$, provided two linearly independent vectors exist to satisfy Eq. 4.16 with $A = \Lambda$. In this case

$$\begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \lambda \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \text{and} \qquad \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

provide such a solution. This is the case where A has a complete set of eigenvectors and is not defective.

A Defective System

If A is defective, then it can be represented by the *Jordan canonical form*

$$\begin{bmatrix} u_1' \\ u_2' \end{bmatrix} = \begin{bmatrix} \lambda & 0 \\ 1 & \lambda \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{4.18}$$

whose solution is not obvious. However, in this case, one can solve the top equation first, giving $u_1(t) = u_1(0)e^{\lambda t}$. Then, substituting this result into the second equation, one finds

$$\frac{du_2}{dt} = \lambda u_2 + u_1(0)e^{\lambda t}$$

which is identical in form to Eq. 4.10 and has the solution

$$u_2(t) = [u_2(0) + u_1(0)t]e^{\lambda t}$$

From this the reader should be able to verify that

$$u_3(t) = \left(a + bt + ct^2\right)e^{\lambda t}$$

is a solution to

$$\begin{bmatrix} u_1' \\ u_2' \\ u_3' \end{bmatrix} = \begin{bmatrix} \lambda & & \\ 1 & \lambda & \\ & 1 & \lambda \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

if

$$a = u_3(0) \quad , \quad b = u_2(0) \quad , \quad c = \frac{1}{2}u_1(0) \tag{4.19}$$

The general solution to such defective systems is left as an exercise.

4.2.4 General Solution of Coupled ODE's with Complete Eigensystems

Let us consider a set of coupled, nonhomogeneous, linear, first-order ODE's with constant coefficients which might have been derived by space differencing a set of PDE's. Represent them by the equation

$$\frac{d\vec{u}}{dt} = A\vec{u} - \vec{f}(t) \tag{4.20}$$

Our assumption is that the $M \times M$ matrix A has a complete eigensystem[1] and can be transformed by the left and right eigenvector matrices, X^{-1} and X, to a diagonal matrix Λ having diagonal elements which are the eigenvalues of A, see Section 4.2.1. Now let us multiply Eq. 4.20 from the left by X^{-1} and insert the identity combination $XX^{-1} = I$ between A and $\vec{u}$. There results

$$X^{-1}\frac{d\vec{u}}{dt} = X^{-1}AX \cdot X^{-1}\vec{u} - X^{-1}\vec{f}(t) \tag{4.21}$$

Since A is independent of both $\vec{u}$ and t, the elements in X^{-1} and X are also independent of both $\vec{u}$ and t, and Eq. 4.21 can be modified to

$$\frac{d}{dt}X^{-1}\vec{u} = \Lambda X^{-1}\vec{u} - X^{-1}\vec{f}(t)$$

Finally, by introducing the new variables $\vec{w}$ and $\vec{g}$ such that

$$\vec{w} = X^{-1}\vec{u} \quad , \quad \vec{g}(t) = X^{-1}\vec{f}(t) \tag{4.22}$$

we reduce Eq. 4.20 to a new algebraic form

$$\frac{d\vec{w}}{dt} = \Lambda\vec{w} - \vec{g}(t) \tag{4.23}$$

It is important at this point to review the results of the previous paragraph. Notice that Eqs. 4.20 and 4.23 are expressing exactly the same equality. The only difference between them was brought about by algebraic manipulations which regrouped the variables. However, this regrouping is crucial for the solution process because Eqs.

[1] In the following, we exclude defective systems, not because they cannot be analyzed (the example at the conclusion of the previous section proves otherwise), but because they are only of limited interest in the general development of our theory.

4.23 are no longer coupled. They can be written line by line as a set of independent, single, first-order equations, thus

$$\begin{aligned} w_1' &= \lambda_1 w_1 - g_1(t) \\ &\vdots \\ w_m' &= \lambda_m w_m - g_m(t) \\ &\vdots \\ w_M' &= \lambda_M w_M - g_M(t) \end{aligned} \tag{4.24}$$

For any given set of $g_m(t)$ each of these equations can be solved separately and then recoupled, using the inverse of the relations given in Eqs. 4.22:

$$\begin{aligned} \vec{u}(t) &= X\vec{w}(t) \\ &= \sum_{m=1}^{M} w_m(t)\vec{x}_m \end{aligned} \tag{4.25}$$

where $\vec{x}_m$ is the m'th column of X, i.e., the eigenvector corresponding to λ_m.

We next focus on the very important subset of Eq. 4.20 when neither A nor $\vec{f}$ has any explicit dependence on t. In such a case, the g_m in Eqs. 4.23 and 4.24 are also time invariant and the solution to any line in Eq. 4.24 is

$$w_m(t) = c_m e^{\lambda_m t} + \frac{1}{\lambda_m} g_m$$

where the c_m are constants that depend on the initial conditions. Transforming back to the u-system gives

$$\begin{aligned} \vec{u}(t) &= X\vec{w}(t) \\ &= \sum_{m=1}^{M} w_m(t)\vec{x}_m \\ &= \sum_{m=1}^{M} c_m e^{\lambda_m t}\,\vec{x}_m + \sum_{m=1}^{M} \frac{1}{\lambda_m} g_m \vec{x}_m \\ &= \sum_{m=1}^{M} c_m e^{\lambda_m t}\,\vec{x}_m + X\Lambda^{-1}X^{-1}\vec{f} \\ &= \underbrace{\sum_{m=1}^{M} c_m e^{\lambda_m t}\,\vec{x}_m}_{\text{Transient}} + \underbrace{A^{-1}\vec{f}}_{\text{Steady-state}} \end{aligned} \tag{4.26}$$

Note that the steady-state solution is $A^{-1}\vec{f}$, as might be expected.

The first group of terms on the right side of this equation is referred to classically as the *complementary solution* or the solution of the homogeneous equations. The second group is referred to classically as the *particular solution* or the particular integral. In our application to fluid dynamics, it is more instructive to refer to these groups as the *transient* and *steady-state* solutions, respectively. An alternative, but entirely equivalent, form of the solution is

$$\vec{u}(t) = c_1 e^{\lambda_1 t}\,\vec{x}_1 + \cdots + c_m e^{\lambda_m t}\,\vec{x}_m + \cdots + c_M e^{\lambda_M t}\,\vec{x}_M \; + \; A^{-1}\vec{f} \tag{4.27}$$

4.3 Real Space and Eigenspace

4.3.1 Definition

Following the semi-discrete approach discussed in Section 4.1, we reduce the partial differential equations to a set of ordinary differential equations represented by the generic form

$$\frac{d\vec{u}}{dt} = A\vec{u} - \vec{f} \tag{4.28}$$

The dependent variable $\vec{u}$ represents some physical quantity or quantities which relate to the problem of interest. For the model problems on which we are focusing most of our attention, the elements of A are independent of both u and t. This permits us to say a great deal about such problems and serves as the basis for this section. In particular, we can develop some very important and fundamental concepts that underly the *global* properties of the numerical solutions to the model problems. How these relate to the numerical solutions of more practical problems depends upon the problem and, to a much greater extent, on the cleverness of the relator.

We begin by developing the concept of "spaces". That is, we identify different mathematical reference frames (spaces) and view our solutions from within each. In this way, we get different perspectives of the same solution, and this can add significantly to our understanding.

The most natural reference frame is the physical one. We say

> If a solution is expressed in terms of $\vec{u}$, it is said to be in *real space*.

There is, however, another very useful frame. We saw in Sections 4.2.1 and 4.2 that pre- and post-multiplication of A by the appropriate similarity matrices transforms A into a diagonal matrix, composed, in the most general case, of Jordan blocks or, in the simplest nondefective case, of scalars. Following Section 4.2 and, for simplicity, using only complete systems for our examples, we found that Eq. 4.28 had the alternative form

$$\frac{d\vec{w}}{dt} = \Lambda\vec{w} - \vec{g}$$

which is an uncoupled set of first-order ODE's that can be solved independently for the dependent variable vector $\vec{w}$. We say

> If a solution is expressed in terms of $\vec{w}$, it is said to be in *eigenspace* (often referred to as wave space).

The relations that transfer from one space to the other are:

$$\boxed{\begin{array}{rclcrl} \vec{w} & = & X^{-1}\vec{u} & \qquad & \vec{u} & = X\vec{w} \\ \vec{g} & = & X^{-1}\vec{f} & \qquad & \vec{f} & = X\vec{g} \end{array}}$$

The elements of $\vec{u}$ relate directly to the local physics of the problem. However, the elements of $\vec{w}$ are linear combinations of all of the elements of $\vec{u}$, and individually they have no direct local physical interpretation.

When the forcing function $\vec{f}$ is independent of t, the solutions in the two spaces are represented by

$$\vec{u}(t) = \sum c_m e^{\lambda_m t}\,\vec{x}_m + X\Lambda^{-1}X^{-1}\vec{f}$$

and

$$w_m(t) = c_m e^{\lambda_m t} + \frac{1}{\lambda_m} g_m \quad ; \quad m = 1, 2, \cdots, M$$

for real space and eigenspace, respectively. At this point we make two observations:

1. the transient portion of the solution in real space consists of a linear combination of contributions from each eigenvector, and

2. the transient portion of the solution in eigenspace provides the weighting of each eigenvector component of the transient solution in real space.

4.3.2 Eigenvalue Spectrums for Model ODE's

It is instructive to consider the eigenvalue spectrums of the ODE's formulated by central differencing the model equations for diffusion and biconvection. These model ODE's are presented in Section 4.1. Equations for the eigenvalues of the simple tridiagonals, $B(M : a, b, c)$ and $B_p(M : a, b, c)$, are given in Appendix B. From Section B.1, we find for the model diffusion equation with Dirichlet boundary conditions

$$\begin{aligned}\lambda_m &= \frac{\nu}{\Delta x^2}\left[-2 + 2\cos\left(\frac{m\pi}{M+1}\right)\right] \\ &= \frac{-4\nu}{\Delta x^2}\sin^2\left(\frac{m\pi}{2(M+1)}\right) \quad ; \quad m = 1, 2, \cdots, M \end{aligned} \tag{4.29}$$

and, from Section B.4, for the model biconvection equation

$$\begin{aligned}\lambda_m &= \frac{-ia}{\Delta x}\sin\left(\frac{2m\pi}{M}\right) \qquad m = 0, 1, \cdots, M-1 \\ &= -i\kappa_m^* a \qquad m = 0, 1, \cdots, M-1 \end{aligned} \tag{4.30}$$

where

$$\kappa_m^* = \frac{\sin \kappa_m \Delta x}{\Delta x} \qquad m = 0, 1, \cdots, M-1 \tag{4.31}$$

is the modified wavenumber from Section 3.5, $\kappa_m = m$, and $\Delta x = 2\pi/M$. Notice that the diffusion eigenvalues are real and negative while those representing periodic convection are all pure imaginary. The interpretation of this result plays a very important role later in our stability analysis.

4.3.3 Eigenvectors of the Model Equations

Next we consider the eigenvectors of the two model equations. These follow as special cases from the results given in Appendix B.

The Diffusion Model

Consider Eq. 4.4, the model ODE's for diffusion. First, to help visualize the matrix structure, we present results for a simple 4-point mesh and then we give the general case. The right-hand eigenvector matrix X is given by

$$\begin{bmatrix} \sin(x_1) & \sin(2x_1) & \sin(3x_1) & \sin(4x_1) \\ \sin(x_2) & \sin(2x_2) & \sin(3x_2) & \sin(4x_2) \\ \sin(x_3) & \sin(2x_3) & \sin(3x_3) & \sin(4x_3) \\ \sin(x_4) & \sin(2x_4) & \sin(3x_4) & \sin(4x_4) \end{bmatrix}$$

The columns of the matrix are the right eigenvectors. Recall that $x_j = j\Delta x = j\pi/(M+1)$, so in general the relation $\vec{u} = X\vec{w}$ can be written as

$$u_j = \sum_{m=1}^{M} w_m \sin m x_j \quad ; \quad j = 1, 2, \cdots, M \tag{4.32}$$

For the inverse, or left-hand eigenvector matrix X^{-1}, we find

$$\begin{bmatrix} \sin(x_1) & \sin(x_2) & \sin(x_3) & \sin(x_4) \\ \sin(2x_1) & \sin(2x_2) & \sin(2x_3) & \sin(2x_4) \\ \sin(3x_1) & \sin(3x_2) & \sin(3x_3) & \sin(3x_4) \\ \sin(4x_1) & \sin(4x_2) & \sin(4x_3) & \sin(4x_4) \end{bmatrix}$$

The rows of the matrix are the left eigenvectors. In general $\vec{w} = X^{-1}\vec{u}$ gives

$$w_m = \sum_{j=1}^{M} u_j \sin mx_j \quad ; \quad m = 1, 2, \cdots, M \tag{4.33}$$

In the field of harmonic analysis, Eq. 4.33 represents a sine transform of the function $u(x)$ for an M-point sample between the boundaries $x = 0$ and $x = \pi$ with the condition $u(0) = u(\pi) = 0$. Similarly, Eq. 4.32 represents the sine synthesis that companions the sine transform given by Eq. 4.33. In summary,

For the model diffusion equation:

$\vec{w} = X^{-1}\vec{u}$ is a sine transform from real space to (sine) wave space.

$\vec{u} = X\vec{w}$ is a sine synthesis from wave space back to real space.

The Biconvection Model

Next consider the model ODE's for periodic convection, Eq. 4.5. The coefficient matrices for these ODE's are always circulant. For our model ODE, the right-hand eigenvectors are given by

$$\vec{x}_m = e^{i\,j\,(2\pi m/M)}, \quad \begin{matrix} j & = 0, 1, \cdots, M-1 \\ m & = 0, 1, \cdots, M-1 \end{matrix}$$

With $x_j = j \cdot \Delta x = j \cdot 2\pi/M$, we can write $\vec{u} = X\vec{w}$ as

$$u_j = \sum_{m=0}^{M-1} w_m e^{imx_j} \quad ; \quad j = 0, 1, \cdots, M-1 \tag{4.34}$$

For a 4-point periodic mesh, we find the following left-hand eigenvector matrix from Appendix B.4:

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{-2i\pi/4} & e^{-4i\pi/4} & e^{-6i\pi/4} \\ 1 & e^{-4i\pi/4} & e^{-8i\pi/4} & e^{-12i\pi/4} \\ 1 & e^{-6i\pi/4} & e^{-12i\pi/4} & e^{-18i\pi/4} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = X^{-1}\vec{u}$$

In general

$$w_m = \frac{1}{M}\sum_{j=0}^{M-1} u_j e^{-imx_j} \quad ; \quad m = 0, 1, \cdots, M-1$$

This equation is identical to a discrete Fourier transform of the periodic dependent variable $\vec{u}$ using an M-point sample between and including $x = 0$ and $x = 2\pi - \Delta x$.

For circulant matrices, it is straightforward to establish the fact that the relation $\vec{u} = X\vec{w}$ represents the Fourier synthesis of the variable $\vec{w}$ back to $\vec{u}$. In summary,

For any circulant system:

$\vec{w} = X^{-1}\vec{u}$ is a complex Fourier transform from real space to wave space.

$\vec{u} = X\vec{w}$ is a complex Fourier synthesis from wave space back to real space.

4.3.4 Solutions of the Model ODE's

We can now combine the results of the previous sections to write the solutions of our model ODE's.

The Diffusion Equation

For the diffusion equation, Eq. 4.27 becomes

$$u_j(t) = \sum_{m=1}^{M} c_m e^{\lambda_m t} \sin mx_j + (A^{-1}f)_j, \qquad j = 1, 2, \cdots, M \tag{4.35}$$

where

$$\lambda_m = \frac{-4\nu}{\Delta x^2}\sin^2\left(\frac{m\pi}{2(M+1)}\right) \tag{4.36}$$

With the modified wavenumber defined as

$$\kappa_m^* = \frac{2}{\Delta x}\sin\left(\frac{\kappa_m \Delta x}{2}\right) \tag{4.37}$$

and using $\kappa_m = m$, we can write the ODE solution as

$$u_j(t) = \sum_{m=1}^{M} c_m e^{-\nu {\kappa_m^*}^2 t} \sin \kappa_m x_j + (A^{-1}f)_j, \qquad j = 1, 2, \cdots, M \tag{4.38}$$

This can be compared with the exact solution to the PDE, Eq. 2.37, evaluated at the nodes of the grid:

$$u_j(t) = \sum_{m=1}^{M} c_m e^{-\nu \kappa_m{}^2 t} \sin \kappa_m x_j + h(x_j), \qquad j = 1, 2, \cdots, M \tag{4.39}$$

We see that the solutions are identical except for the steady solution and the modified wavenumber in the transient term. The modified wavenumber is an approximation to the actual wavenumber. The difference between the modified wavenumber and the actual wavenumber depends on the differencing scheme and the grid resolution. This difference causes the various modes (or eigenvector components) to decay at rates which differ from the exact solution. With conventional differencing schemes, low wavenumber modes are accurately represented, while high wavenumber modes (if they have significant amplitudes) can have large errors.

The Convection Equation

For the biconvection equation, we obtain

$$u_j(t) = \sum_{m=0}^{M-1} c_m e^{\lambda_m t} e^{i \kappa_m x_j}, \qquad j = 0, 1, \cdots, M-1 \tag{4.40}$$

where

$$\lambda_m = -i \kappa_m^* a \tag{4.41}$$

with the modified wavenumber defined in Eq. 4.31. We can write this ODE solution as

$$u_j(t) = \sum_{m=0}^{M-1} c_m e^{-i \kappa_m^* a t} e^{i \kappa_m x_j}, \qquad j = 0, 1, \cdots, M-1 \tag{4.42}$$

and compare it to the exact solution of the PDE, Eq. 2.26, evaluated at the nodes of the grid:

$$u_j(t) = \sum_{m=0}^{M-1} f_m(0) e^{-i \kappa_m a t} e^{i \kappa_m x_j}, \qquad j = 0, 1, \cdots, M-1 \tag{4.43}$$

Once again the difference appears through the modified wavenumber contained in λ_m. As discussed in Section 3.5, this leads to an error in the speed with which various modes are convected, since κ^* is real. Since the error in the phase speed depends on the wavenumber, while the actual phase speed is independent of the wavenumber, the result is erroneous numerical dispersion. In the case of non-centered differencing, discussed in Chapter 11, the modified wavenumber is complex. The form of Eq. 4.42 shows that the imaginary portion of the modified wavenumber produces nonphysical decay or growth in the numerical solution.

4.4 The Representative Equation

In Section 4.3, we pointed out that Eqs. 4.20 and 4.23 express identical results but in terms of different groupings of the dependent variables, which are related by algebraic manipulation. This leads to the following important concept:

> The numerical solution to a set of linear ODE's (in which A is *not* a function of t) is entirely equivalent to the solution obtained if the equations are transformed to eigenspace, solved there in their uncoupled form, and then returned as a coupled set to real space.

The importance of this concept resides in its message that we can analyze time-marching methods by applying them to a single, uncoupled equation and our conclusions will apply in general. This is helpful both in analyzing the accuracy of time-marching methods and in studying their stability, topics which are covered in Chapters 6 and 7.

Our next objective is to find a "typical" single ODE to analyze. We found the uncoupled solution to a set of ODE's in Section 4.2. A typical member of the family is

$$\frac{dw_m}{dt} = \lambda_m w_m - g_m(t) \tag{4.44}$$

The goal in our analysis is to study typical behavior of general situations, not particular problems. For such a purpose Eq. 4.44 is not quite satisfactory. The role of λ_m is clear; it stands for some representative eigenvalue in the original A matrix. However, the question is: What should we use for $g_m(t)$ when the time dependence cannot be ignored? To answer this question, we note that, in principle, one can express any one of the forcing terms $g_m(t)$ as a finite Fourier series. For example

$$-g(t) = \sum_k a_k e^{ikt}$$

for which Eq. 4.44 has the exact solution:

$$w(t) = ce^{\lambda t} + \sum_k \frac{a_k e^{ikt}}{ik - \lambda}$$

From this we can extract the k'th term and replace ik with μ. This leads to

> The Representative ODE
>
> $$\frac{dw}{dt} = \lambda w + ae^{\mu t} \tag{4.45}$$

which can be used to evaluate all manner of time-marching methods. In such evaluations the parameters λ and μ must be allowed to take the worst possible combination of values that might occur in the ODE eigensystem. The exact solution of the representative ODE is (for $\mu \neq \lambda$):

$$w(t) = ce^{\lambda t} + \frac{ae^{\mu t}}{\mu - \lambda} \tag{4.46}$$

4.5 Problems

1. Consider the finite-difference operator derived in question 1 of Chapter 3. Using this operator to approximate the spatial derivative in the linear convection equation, write the semi-discrete form obtained with periodic boundary conditions on a 5-point grid ($M = 5$).

2. Write the semi-discrete form resulting from the application of second-order centered differences to the following equation on the domain $0 \leq x \leq 1$ with boundary conditions $u(0) = 0$, $u(1) = 1$:
$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - 6x$$

3. Consider a grid with 10 interior points spanning the domain $0 \leq x \leq \pi$. For initial conditions $u(x, 0) = \sin(mx)$ and boundary conditions $u(0, t) = u(\pi, t) = 0$, plot the exact solution of the diffusion equation with $\nu = 1$ at $t = 1$ with $m = 1$ and $m = 3$. (Plot the solution at the grid nodes only.) Calculate the corresponding modified wavenumbers for the second-order centered operator from Eq. 4.37. Calculate and plot the corresponding ODE solutions.

4. Consider the matrix
$$A = -B_p(10, -1, 0, 1)/(2\Delta x)$$
corresponding to the ODE form of the biconvection equation resulting from the application of second-order central differencing on a 10-point grid. Note that the domain is $0 \leq x \leq 2\pi$ and $\Delta x = 2\pi/10$. The grid nodes are given by $x_j = j\Delta x, \quad j = 0, 1, \ldots 9$. The eigenvalues of the above matrix A, as well as the matrices X and X^{-1}, can be found from Appendix B.4. Using these, compute and plot the ODE solution at $t = 2\pi$ for the initial condition $u(x, 0) = \sin x$. Compare with the exact solution of the PDE. Calculate the numerical phase speed from the modified wavenumber corresponding to this initial condition and show that it is consistent with the ODE solution. Repeat for the initial condition $u(x, 0) = \sin 2x$.

Chapter 6

TIME-MARCHING METHODS FOR ODE'S

After discretizing the spatial derivatives in the governing PDE's (such as the Navier-Stokes equations), we obtain a coupled system of nonlinear ODE's in the form

$$\frac{d\vec{u}}{dt} = \vec{F}(\vec{u}, t) \tag{6.1}$$

These can be integrated in time using a time-marching method to obtain a time-accurate solution to an *unsteady* flow problem. For a *steady* flow problem, spatial discretization leads to a coupled system of nonlinear algebraic equations in the form

$$\vec{F}(\vec{u}) = 0 \tag{6.2}$$

As a result of the nonlinearity of these equations, some sort of iterative method is required to obtain a solution. For example, one can consider the use of Newton's method, which is widely used for nonlinear algebraic equations (See Section 6.10.3.). This produces an iterative method in which a coupled system of linear algebraic equations must be solved at each iteration. These can be solved iteratively using relaxation methods, which will be discussed in Chapter 9, or directly using Gaussian elimination or some variation thereof.

Alternatively, one can consider a time-dependent path to the steady state and use a time-marching method to integrate the unsteady form of the equations until the solution is sufficiently close to the steady solution. The subject of the present chapter, time-marching methods for ODE's, is thus relevant to both steady and unsteady flow problems. When using a time-marching method to compute steady flows, the goal is simply to remove the transient portion of the solution as quickly as possible; time-accuracy is not required. This motivates the study of stability and stiffness, topics which are covered in the next two chapters.

Application of a spatial discretization to a PDE produces a coupled system of ODE's. Application of a time-marching method to an ODE produces an ordinary *difference* equation (OΔE). In earlier chapters, we developed exact solutions to our model PDE's and ODE's. In this chapter we will present some basic theory of linear OΔE's which closely parallels that for linear ODE's, and, using this theory, we will develop exact solutions for the model OΔE's arising from the application of time-marching methods to the model ODE's.

6.1 Notation

Using the semi-discrete approach, we reduce our PDE to a set of coupled ODE's represented in general by Eq. 4.1. However, for the purpose of this chapter, we need only consider the scalar case

$$\frac{du}{dt} = u' = F(u,t) \tag{6.3}$$

Although we use u to represent the dependent variable, rather than w, the reader should recall the arguments made in Chapter 4 to justify the study of a scalar ODE. Our first task is to find numerical approximations that can be used to carry out the time integration of Eq. 6.3 to some given accuracy, where accuracy can be measured either in a local or a global sense. We then face a further task concerning the numerical stability of the resulting methods, but we postpone such considerations to the next chapter.

In Chapter 2, we introduced the convention that the n subscript, or the (n) superscript, always points to a discrete time value, and h represents the time interval Δt. Combining this notation with Eq. 6.3 gives

$$u'_n = F_n = F(u_n, t_n) \quad ; \quad t_n = nh$$

Often we need a more sophisticated notation for intermediate time steps involving temporary calculations denoted by $\tilde{u}$, $\bar{u}$, etc. For these we use the notation

$$\tilde{u}'_{n+\alpha} = \tilde{F}_{n+\alpha} = F(\tilde{u}_{n+\alpha}, t_n + \alpha h)$$

The choice of u' or F to express the derivative in a scheme is arbitrary. They are both commonly used in the literature on ODE's.

The methods we study are to be applied to linear or nonlinear ODE's, but the methods themselves are formed by linear combinations of the dependent variable and its derivative at various time intervals. They are represented conceptually by

$$u_{n+1} = f\Big(\beta_1 u'_{n+1}, \beta_0 u'_n, \beta_{-1} u'_{n-1}, \cdots, \alpha_0 u_n, \alpha_{-1} u_{n-1}, \cdots\Big) \tag{6.4}$$

With an appropriate choice of the $\alpha' s$ and $\beta' s$, these methods can be constructed to give a local Taylor series accuracy of any order. The methods are said to be *explicit* if $\beta_1 = 0$ and *implicit* otherwise. An *explicit* method is one in which the new predicted solution is only a function of known data, for example, u'_n, u'_{n-1}, u_n, and u_{n-1} for a method using two previous time levels, and therefore the time advance is simple. For an *implicit* method, the new predicted solution is also a function of the time derivative at the new time level, that is, u'_{n+1}. As we shall see, for systems of ODE's and nonlinear problems, *implicit* methods require more complicated strategies to solve for u_{n+1} than *explicit* methods.

6.2 Converting Time-Marching Methods to OΔE's

Examples of some very common forms of methods used for time-marching general ODE's are:

$$u_{n+1} = u_n + hu'_n \tag{6.5}$$

$$u_{n+1} = u_n + hu'_{n+1} \tag{6.6}$$

and

$$\begin{aligned} \tilde{u}_{n+1} &= u_n + hu'_n \\ u_{n+1} &= \frac{1}{2}[u_n + \tilde{u}_{n+1} + h\tilde{u}'_{n+1}] \end{aligned} \tag{6.7}$$

According to the conditions presented under Eq. 6.4, the first and third of these are examples of explicit methods. We refer to them as the explicit Euler method and the MacCormack predictor-corrector method,[1] respectively. The second is implicit and referred to as the implicit (or backward) Euler method.

These methods are simple recipes for the time advance of a function in terms of its value and the value of its derivative, at given time intervals. The material presented in Chapter 4 develops a basis for evaluating such methods by introducing the concept of the representative equation

$$\frac{du}{dt} = u' = \lambda u + ae^{\mu t} \tag{6.8}$$

written here in terms of the dependent variable, u. The value of this equation arises from the fact that, by applying a time-marching method, we can analytically convert

[1] Here we give only MacCormack's *time-marching* method. The method commonly referred to as MacCormack's method, which is a fully-discrete method, will be presented in Section 11.3

such a linear ODE into a linear OΔE . The latter are subject to a whole body of analysis that is similar in many respects to, and just as powerful as, the theory of ODE's. We next consider examples of this conversion process and then go into the general theory on solving OΔE's.

Apply the simple explicit Euler scheme, Eq. 6.5, to Eq. 6.8. There results

$$u_{n+1} = u_n + h(\lambda u_n + ae^{\mu hn})$$

or

$$u_{n+1} - (1 + \lambda h)u_n = hae^{\mu hn} \tag{6.9}$$

Eq. 6.9 is a linear OΔE , with constant coefficients, expressed in terms of the dependent variable u_n and the independent variable n. As another example, applying the implicit Euler method, Eq. 6.6, to Eq. 6.8, we find

$$u_{n+1} = u_n + h\Big(\lambda u_{n+1} + ae^{\mu h(n+1)}\Big)$$

or

$$(1 - \lambda h)u_{n+1} - u_n = he^{\mu h} \cdot ae^{\mu hn} \tag{6.10}$$

As a final example, the predictor-corrector sequence, Eq. 6.7, gives

$$\begin{aligned} \tilde{u}_{n+1} - (1 + \lambda h)u_n &= ahe^{\mu hn} \\ -\frac{1}{2}(1 + \lambda h)\tilde{u}_{n+1} + u_{n+1} - \frac{1}{2}u_n &= \frac{1}{2}ahe^{\mu h(n+1)} \end{aligned} \tag{6.11}$$

which is a coupled set of linear OΔE's with constant coefficients. Note that the first line in Eq. 6.11 is identical to Eq. 6.9, since the predictor step in Eq. 6.7 is simply the explicit Euler method. The second line in Eq. 6.11 is obtained by noting that

$$\begin{aligned} \tilde{u}'_{n+1} &= F(\tilde{u}_{n+1}, t_n + h) \\ &= \lambda\tilde{u}_{n+1} + ae^{\mu h(n+1)} \end{aligned} \tag{6.12}$$

Now we need to develop techniques for analyzing these difference equations so that we can compare the merits of the time-marching methods that generated them.

6.3 Solution of Linear OΔE's With Constant Coefficients

The techniques for solving *linear difference equations* with constant coefficients is as well developed as that for ODE's and the theory follows a remarkably parallel path. This is demonstrated by repeating some of the developments in Section 4.2, but for difference rather than differential equations.

6.3.1 First- and Second-Order Difference Equations

First-Order Equations

The simplest nonhomogeneous OΔE of interest is given by the single first-order equation

$$u_{n+1} = \sigma u_n + ab^n \tag{6.13}$$

where σ, a, and b are, in general, complex parameters. The independent variable is now n rather than t, and since the equations are linear and have constant coefficients, σ is not a function of either n or u. The exact solution of Eq. 6.13 is

$$u_n = c_1\sigma^n + \frac{ab^n}{b-\sigma}$$

where c_1 is a constant determined by the initial conditions. In terms of the initial value of u it can be written

$$u_n = u_0\sigma^n + a\frac{b^n - \sigma^n}{b-\sigma}$$

Just as in the development of Eq. 4.10, one can readily show that the solution of the defective case, $(b = \sigma)$,

$$u_{n+1} = \sigma u_n + a\sigma^n$$

is

$$u_n = \left[u_0 + an\sigma^{-1}\right]\sigma^n$$

This can all be easily verified by substitution.

Second-Order Equations

The homogeneous form of a second-order difference equation is given by

$$u_{n+2} + a_1 u_{n+1} + a_0 u_n = 0 \tag{6.14}$$

Instead of the differential operator $\mathrm{D} \equiv \frac{d}{dt}$ used for ODE's, we use for OΔE's the difference operator E (commonly referred to as the displacement or shift operator) and defined formally by the relations

$$u_{n+1} = Eu_n \quad , \quad u_{n+k} = E^k u_n$$

Further notice that the displacement operator also applies to exponents, thus

$$b^\alpha \cdot b^n = b^{n+\alpha} = E^\alpha \cdot b^n$$

where α can be any fraction or irrational number.

The roles of D and E are the same insofar as once they have been introduced to the basic equations the value of $u(t)$ or u_n can be factored out. Thus Eq. 6.14 can now be re-expressed in an operational notion as

$$(E^2 + a_1 E + a_0)u_n = 0 \tag{6.15}$$

which must be zero for all u_n. Eq. 6.15 is known as the *operational form* of Eq. 6.14. The operational form contains a characteristic polynomial $P(E)$ which plays the same role for difference equations that $P(\mathrm{D})$ played for differential equations; that is, its roots determine the solution to the OΔE. In the analysis of OΔE's, we label these roots σ_1, σ_2, $\cdots$, etc, and refer to them as the *σ-roots.* They are found by solving the equation $P(\sigma) = 0$. In the simple example given above, there are just two σ roots and in terms of them the solution can be written

$$u_n = c_1(\sigma_1)^n + c_2(\sigma_2)^n \tag{6.16}$$

where c_1 and c_2 depend upon the initial conditions. The fact that Eq. 6.16 is a solution to Eq. 6.14 for all c_1 , c_2 and n should be verified by substitution.

6.3.2 Special Cases of Coupled First-Order Equations

A Complete System

Coupled, first-order, linear homogeneous difference equations have the form

$$\begin{aligned} u_1^{(n+1)} &= c_{11}u_1^{(n)} + c_{12}u_2^{(n)} \\ u_2^{(n+1)} &= c_{21}u_1^{(n)} + c_{22}u_2^{(n)} \end{aligned} \tag{6.17}$$

which can also be written

$$\vec{u}_{n+1} = C\vec{u}_n \quad , \quad \vec{u}_n = \left[u_1^{(n)}, u_2^{(n)}\right]^T \quad , \quad C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

The operational form of Eq. 6.17 can be written

$$\begin{bmatrix} (c_{11} - E) & c_{12} \\ c_{21} & (c_{22} - E) \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}^{(n)} = [\,C - E\ I\,]\vec{u}_n = 0$$

which must be zero for all u_1 and u_2. Again we are led to a characteristic polynomial, this time having the form $P(E) = \det[\,C - E\ I\,]$. The σ-roots are found from

$$P(\sigma) = \det \begin{bmatrix} (c_{11} - \sigma) & c_{12} \\ c_{21} & (c_{22} - \sigma) \end{bmatrix} = 0$$

Obviously the σ_k are the eigenvalues of C and, following the logic of Section 4.2, if $\vec{x}$ are its eigenvectors, the solution of Eq. 6.17 is

$$\vec{u}_n = \sum_{k=1}^{2} c_k(\sigma_k)^n \vec{x}_k$$

where c_k are constants determined by the initial conditions.

A Defective System

The solution of OΔE's with defective eigensystems follows closely the logic in Section 4.2.2 for defective ODE's. For example, one can show that the solution to

$$\begin{bmatrix} \bar{u}_{n+1} \\ \hat{u}_{n+1} \\ u_{n+1} \end{bmatrix} = \begin{bmatrix} \sigma & & \\ 1 & \sigma & \\ & 1 & \sigma \end{bmatrix} \begin{bmatrix} \bar{u}_n \\ \hat{u}_n \\ u_n \end{bmatrix}$$

is

$$\begin{aligned} \bar{u}_n &= \bar{u}_0 \sigma^n \\ \hat{u}_n &= \left[\hat{u}_0 + \bar{u}_0 n \sigma^{-1} \right] \sigma^n \\ u_n &= \left[u_0 + \hat{u}_0 n \sigma^{-1} + \bar{u}_0 \frac{n(n-1)}{2} \sigma^{-2} \right] \sigma^n \end{aligned} \tag{6.18}$$

6.4 Solution of the Representative OΔE's

6.4.1 The Operational Form and its Solution

Examples of the nonhomogeneous, linear, first-order ordinary difference equations, produced by applying a time-marching method to the representative equation, are given by Eqs. 6.9 to 6.11. Using the displacement operator, E, these equations can be written

$$[E - (1 + \lambda h)] u_n = h \cdot a e^{\mu h n} \tag{6.19}$$

$$[(1 - \lambda h)E - 1] u_n = h \cdot E \cdot a e^{\mu h n} \tag{6.20}$$

$$\begin{bmatrix} E & -(1 + \lambda h) \\ -\frac{1}{2}(1 + \lambda h)E & E - \frac{1}{2} \end{bmatrix} \begin{bmatrix} \tilde{u} \\ u \end{bmatrix}_n = h \cdot \begin{bmatrix} 1 \\ \frac{1}{2}E \end{bmatrix} \cdot a e^{\mu h n} \tag{6.21}$$

All three of these equations are subsets of the *operational form* of the representative OΔE

$$\boxed{P(E)u_n = Q(E) \cdot ae^{\mu hn}} \tag{6.22}$$

which is produced by applying time-marching methods to the representative ODE, Eq. 4.45. We can express in terms of Eq. 6.22 all manner of standard time-marching methods having multiple time steps and various types of intermediate predictor-corrector families. The terms $P(E)$ and $Q(E)$ are polynomials in E referred to as the *characteristic polynomial* and the *particular polynomial*, respectively.

The general solution of Eq. 6.22 can be expressed as

$$u_n = \sum_{k=1}^{K} c_k(\sigma_k)^n + ae^{\mu hn} \cdot \frac{Q(e^{\mu h})}{P(e^{\mu h})} \tag{6.23}$$

where σ_k are the K roots of the characteristic polynomial, $P(\sigma) = 0$. When determinants are involved in the construction of $P(E)$ and $Q(E)$, as would be the case for Eq. 6.21, the ratio $Q(E)/P(E)$ can be found by Cramer's rule. Keep in mind that for methods such as in Eq. 6.21 there are multiple (two in this case) solutions, one for u_n and $\tilde{u}_n$ and we are usually only interested in the final solution u_n. Notice also, the important subset of this solution which occurs when $\mu = 0$, representing a time invariant particular solution, or a steady state. In such a case

$$u_n = \sum_{k=1}^{K} c_k(\sigma_k)^n + a \cdot \frac{Q(1)}{P(1)}$$

6.4.2 Examples of Solutions to Time-Marching OΔE's

As examples of the use of Eqs. 6.22 and 6.23, we derive the solutions of Eqs. 6.19 to 6.21. For the explicit Euler method, Eq. 6.19, we have

$$\begin{aligned} P(E) &= E - 1 - \lambda h \\ Q(E) &= h \end{aligned} \tag{6.24}$$

and the solution of its representative OΔE follows immediately from Eq. 6.23:

$$u_n = c_1(1 + \lambda h)^n + ae^{\mu hn} \cdot \frac{h}{e^{\mu h} - 1 - \lambda h}$$

For the implicit Euler method, Eq. 6.20, we have

$$\begin{aligned} P(E) &= (1 - \lambda h)E - 1 \\ Q(E) &= hE \end{aligned} \tag{6.25}$$

so

$$u_n = c_1\left(\frac{1}{1-\lambda h}\right)^n + ae^{\mu hn}\cdot\frac{he^{\mu h}}{(1-\lambda h)e^{\mu h}-1}$$

In the case of the coupled predictor-corrector equations, Eq. 6.21, one solves for the final family u_n (one can also find a solution for the intermediate family $\tilde{u}$), and there results

$$P(E) = \det\begin{bmatrix} E & -(1+\lambda h) \\ -\frac{1}{2}(1+\lambda h)E & E-\frac{1}{2}\end{bmatrix} = E\left(E-1-\lambda h-\frac{1}{2}\lambda^2h^2\right)$$

$$Q(E) = \det\begin{bmatrix} E & h \\ -\frac{1}{2}(1+\lambda h)E & \frac{1}{2}hE\end{bmatrix} = \frac{1}{2}hE(E+1+\lambda h)$$

The σ-root is found from

$$P(\sigma) = \sigma\left(\sigma-1-\lambda h-\frac{1}{2}\lambda^2h^2\right) = 0$$

which has only one nontrivial root ($\sigma = 0$ is simply a shift in the reference index). The complete solution can therefore be written

$$u_n = c_1\left(1+\lambda h+\frac{1}{2}\lambda^2h^2\right)^n + ae^{\mu hn}\cdot\frac{\frac{1}{2}h\left(e^{\mu h}+1+\lambda h\right)}{e^{\mu h}-1-\lambda h-\frac{1}{2}\lambda^2h^2} \tag{6.26}$$

6.5 The $\lambda - \sigma$ Relation

6.5.1 Establishing the Relation

We have now been introduced to two basic kinds of roots, the λ-roots and the σ-roots. The former are the eigenvalues of the A matrix in the ODE's found by space differencing the original PDE, and the latter are the roots of the characteristic polynomial in a representative OΔE found by applying a time-marching method to the representative ODE. There is a fundamental relation between the two which can be used to identify many of the essential properties of a time-march method. This relation is first demonstrated by developing it for the explicit Euler method.

First we make use of the semi-discrete approach to find a system of ODE's and then express its solution in the form of Eq. 4.27. Remembering that $t = nh$, one can write

$$\vec{u}(t) = c_1\left(e^{\lambda_1 h}\right)^n\vec{x}_1 + \cdots + c_m\left(e^{\lambda_m h}\right)^n\vec{x}_m + \cdots + c_M\left(e^{\lambda_M h}\right)^n\vec{x}_M \;+\; P.S. \tag{6.27}$$

where for the present we are not interested in the form of the particular solution ($P.S.$). Now the explicit Euler method produces for each λ-root, one σ-root, which is given by $\sigma = 1 + \lambda h$. So if we use the Euler method for the time advance of the ODE's, the solution[2] of the resulting OΔE is

$$\vec{u}_n = c_1(\sigma_1)^n\,\vec{x}_1 + \cdots + c_m(\sigma_m)^n\,\vec{x}_m + \cdots + c_M(\sigma_M)^n\,\vec{x}_M \;+\; P.S. \tag{6.28}$$

where the c_m and the $\vec{x}_m$ in the two equations are identical and $\sigma_m = (1 + \lambda_m h)$. Comparing Eq. 6.27 and Eq. 6.28, we see a correspondence between σ_m and $e^{\lambda_m h}$. Since the value of $e^{\lambda h}$ can be expressed in terms of the series

$$e^{\lambda h} = 1 + \lambda h + \frac{1}{2}\lambda^2 h^2 + \frac{1}{6}\lambda^3 h^3 + \cdots + \frac{1}{n!}\lambda^n h^n + \cdots$$

the truncated expansion $\sigma = 1 + \lambda h$ is a reasonable[3] approximation for small enough λh.

Suppose, instead of the Euler method, we use the leapfrog method for the time advance, which is defined by

$$u_{n+1} \;= u_{n-1} + 2hu'_n \tag{6.29}$$

Applying Eq. 6.8 to Eq. 6.29, we have the characteristic polynomial $P(E) = E^2 - 2\lambda h E - 1$, so that for every λ the σ must satisfy the relation

$$\sigma_m^2 - 2\lambda_m h \sigma_m - 1 = 0 \tag{6.30}$$

Now we notice that each λ produces two σ-roots. For one of these we find

$$\begin{aligned} \sigma_m &= \lambda_m h + \sqrt{1 + \lambda_m^2 h^2} && (6.31)\\ &= 1 + \lambda_m h + \frac{1}{2}\lambda_m^2 h^2 - \frac{1}{8}\lambda_m^4 h^4 + \cdots && (6.32) \end{aligned}$$

This is an approximation to $e^{\lambda_m h}$ with an error $O(\lambda^3 h^3)$. The other root, $\lambda_m h - \sqrt{1 + \lambda_m^2 h^2}$, will be discussed in Section 6.5.3.

6.5.2 The Principal σ-Root

Based on the above we make the following observation:

> Application of the same time-marching method to all of the equations in a coupled system linear ODE's in the form of Eq. 4.6, always produces one σ-root for every λ-root that satisfies the relation
>
> $$\sigma = 1 + \lambda h + \frac{1}{2}\lambda^2 h^2 + \cdots + \frac{1}{k!}\lambda^k h^k + O\left(h^{k+1}\right)$$
>
> where k is the order of the time-marching method.

(6.33)

[2] Based on Section 4.4.
[3] The error is $O(\lambda^2 h^2)$.

We refer to the root that has the above property as the *principal* σ-root, and designate it $(\sigma_m)_1$. The above property can be stated regardless of the details of the time-marching method, knowing only that its leading error is $O\left(h^{k+1}\right)$. Thus the principal root is an approximation to $e^{\lambda h}$ up to $O\left(h^k\right)$.

Note that a second-order approximation to a derivative written in the form

$$(\delta_t u)_n = \frac{1}{2h}(u_{n+1} - u_{n-1}) \tag{6.34}$$

has a leading truncation error which is $O(h^2)$, while the second-order time-marching method which results from this approximation, which is the leapfrog method:

$$u_{n+1} = u_{n-1} + 2hu'_n \tag{6.35}$$

has a leading truncation error $O(h^3)$. This arises simply because of our notation for the time-marching method in which we have multiplied through by h to get an approximation for the *function* u_{n+1} rather than the *derivative* as in Eq. 6.34. The following example makes this clear. Consider a solution obtained at a given time T using a second-order time-marching method with a time step h. Now consider the solution obtained using the same method with a time step $h/2$. Since the error per time step is $O(h^3)$, this is reduced by a factor of eight (considering the leading term only). However, twice as many time steps are required to reach the time T. Therefore the error at the end of the simulation is reduced by a factor of four, consistent with a second-order approximation.

6.5.3 Spurious σ-Roots

We saw from Eq. 6.30 that the $\lambda - \sigma$ relation for the leapfrog method produces two σ-roots for each λ. One of these we identified as the principal root which always has the property given in 6.33. The other is referred to as a *spurious* σ-root and designated $(\sigma_m)_2$. In general, the $\lambda - \sigma$ relation produced by a time-marching scheme can result in multiple σ-roots all of which, except for the principal one, are spurious. All spurious roots are designated $(\sigma_m)_k$ where $k = 2, 3, \cdots$. No matter whether a σ-root is principal or spurious, it is always some algebraic function of the product λh. To express this fact we use the notation $\sigma = \sigma(\lambda h)$.

If a time-marching method produces spurious σ-roots, the solution for the OΔE in the form shown in Eq. 6.28 must be modified. Following again the message of Section 4.4, we have

$$\begin{aligned}\vec{u}_n &= c_{11}(\sigma_1)_1^n\, \vec{x}_1 + \cdots + c_{m1}(\sigma_m)_1^n\, \vec{x}_m + \cdots + c_{M1}(\sigma_M)_1^n\, \vec{x}_M + P.S. \\ &\quad + c_{12}(\sigma_1)_2^n\, \vec{x}_1 + \cdots + c_{m2}(\sigma_m)_2^n\, \vec{x}_m + \cdots + c_{M2}(\sigma_M)_2^n\, \vec{x}_M\end{aligned}$$

$$
\begin{aligned}
&+c_{13}(\sigma_1)_3^n\,\vec{x}_1+\cdots+c_{m3}(\sigma_m)_3^n\,\vec{x}_m+\cdots+c_{M3}(\sigma_M)_3^n\,\vec{x}_M\\
&+\text{etc., if there are more spurious roots}
\end{aligned}
\tag{6.36}
$$

Spurious roots arise if a method uses data from time level $n-1$ or earlier to advance the solution from time level n to $n+1$. Such roots originate entirely from the numerical approximation of the time-marching method and have nothing to do with the ODE being solved. However, generation of spurious roots does not, in itself, make a method inferior. In fact, many very accurate methods in practical use for integrating some forms of ODE's have spurious roots.

It should be mentioned that methods with spurious roots are not self starting. For example, if there is one spurious root to a method, all of the coefficients $(c_m)_2$ in Eq. 6.36 must be initialized by some starting procedure. The initial vector $\vec{u}_0$ does not provide enough data to initialize all of the coefficients. This results because methods which produce spurious roots require data from time level $n-1$ or earlier. For example, the leapfrog method requires $\vec{u}_{n-1}$ and thus cannot be started using only $\vec{u}_n$.

Presumably (i.e., if one starts the method properly) the spurious coefficients are all initialized with very small magnitudes, and presumably the magnitudes of the spurious roots themselves are all less than one (see Chapter 7). Then the presence of spurious roots does not contaminate the answer. That is, after some finite time the amplitude of the error associated with the spurious roots is even smaller then when it was initialized. Thus while spurious roots must be considered in *stability* analysis, they play virtually no role in *accuracy* analysis.

6.5.4 One-Root Time-Marching Methods

There are a number of time-marching methods that produce only one σ-root for each λ-root. We refer to them as *one-root* methods. They are also called one-step methods. They have the significant advantage of being self-starting which carries with it the very useful property that the time-step interval can be changed at will throughout the marching process. Three one-root methods were analyzed in Section 6.4.2. A popular method having this property, the so-called θ-method, is given by the formula

$$u_{n+1}=u_n+h\Big[(1-\theta)u'_n+\theta u'_{n+1}\Big]$$

The θ-method represents the explicit Euler ($\theta=0$), the trapezoidal ($\theta=\frac{1}{2}$), and the implicit Euler methods ($\theta=1$), respectively. Its $\lambda-\sigma$ relation is

$$\sigma=\frac{1+(1-\theta)\lambda h}{1-\theta\lambda h}$$

It is instructive to compare the exact solution to a set of ODE's (with a complete eigensystem) having time-invariant forcing terms with the exact solution to the OΔE's for one-root methods. These are

$$\begin{aligned}\vec{u}(t) &= c_1\left(e^{\lambda_1 h}\right)^n \vec{x}_1 + \cdots + c_m\left(e^{\lambda_m h}\right)^n \vec{x}_m + \cdots + c_M\left(e^{\lambda_M h}\right)^n \vec{x}_M + A^{-1}\vec{f} \\ \vec{u}_n &= c_1(\sigma_1)^n \vec{x}_1 + \cdots + c_m(\sigma_m)^n \vec{x}_m + \cdots + c_M(\sigma_M)^n \vec{x}_M + A^{-1}\vec{f}\end{aligned} \tag{6.37}$$

respectively. Notice that when t and $n = 0$, these equations are identical, so that all the constants, vectors, and matrices are identical except the $\vec{u}$ and the terms inside the parentheses on the right hand sides. The only error made by introducing the time marching is the error that σ makes in approximating $e^{\lambda h}$.

6.6 Accuracy Measures of Time-Marching Methods

6.6.1 Local and Global Error Measures

There are two broad categories of errors that can be used to derive and evaluate time-marching methods. One is the error made in each time step. This is a *local* error such as that found from a Taylor table analysis, see Section 3.4. It is usually used as the basis for establishing the order of a method. The other is the error determined at the end of a given *event* which has covered a specific interval of time composed of many time steps. This is a *global* error. It is useful for comparing methods, as we shall see in Chapter 8.

It is quite common to judge a time-marching method on the basis of results found from a Taylor table. However, a Taylor series analysis is a very limited tool for finding the more subtle properties of a numerical time-marching method. For example, it is of no use in:

- finding spurious roots.
- evaluating numerical stability and separating the errors in phase and amplitude.
- analyzing the particular solution of predictor-corrector combinations.
- finding the global error.

The latter three of these are of concern to us here, and to study them we make use of the material developed in the previous sections of this chapter. Our error measures are based on the difference between the exact solution to the representative ODE, given by

$$u(t) = ce^{\lambda t} + \frac{ae^{\mu t}}{\mu - \lambda} \tag{6.38}$$

and the solution to the representative OΔE's, including only the contribution from the principal root, which can be written as

$$u_n = c_1(\sigma_1)^n + ae^{\mu hn} \cdot \frac{Q(e^{\mu h})}{P(e^{\mu h})} \tag{6.39}$$

6.6.2 Local Accuracy of the Transient Solution $(er_\lambda, |\sigma|, er_\omega)$

Transient error

The particular choice of an error measure, either local or global, is to some extent arbitrary. However, a necessary condition for the choice should be that the measure can be used consistently for all methods. In the discussion of the λ-σ relation we saw that all time-marching methods produce a principal σ-root for every λ-root that exists in a set of linear ODE's. Therefore, a very natural local error measure for the transient solution is the value of the difference between solutions based on these two roots. We designate this by er_λ and make the following definition

$$\boxed{er_\lambda \equiv e^{\lambda h} - \sigma_1}$$

The leading error term can be found by expanding in a Taylor series and choosing the first nonvanishing term. This is similar to the error found from a Taylor table. The *order* of the method is the last power of λh matched exactly.

Amplitude and Phase Error

Suppose a λ eigenvalue is imaginary. Such can indeed be the case when we study the equations governing periodic convection which produces harmonic motion. For such cases it is more meaningful to express the error in terms of amplitude and phase. Let $\lambda = i\omega$ where ω is a real number representing a frequency. Then the numerical method must produce a principal σ-root that is complex and expressible in the form

$$\sigma_1 = \sigma_r + i\sigma_i \approx e^{i\omega h} \tag{6.40}$$

From this it follows that the local error in amplitude is measured by the deviation of $|\sigma_1|$ from unity, that is

$$\boxed{er_a = 1 - |\sigma_1| = 1 - \sqrt{(\sigma_1)_r^2 + (\sigma_1)_i^2}}$$

and the local error in phase can be defined as

$$\boxed{er_\omega \equiv \omega h - \tan^{-1}\left[(\sigma_1)_i/(\sigma_1)_r)\right]} \tag{6.41}$$

Amplitude and phase errors are important measures of the suitability of time-marching methods for convection and wave propagation phenomena.

The approach to error analysis described in Section 3.5 can be extended to the combination of a spatial discretization and a time-marching method applied to the linear convection equation. The principal root, $\sigma_1(\lambda h)$, is found using $\lambda = -ia\kappa^*$, where κ^* is the modified wavenumber of the spatial discretization. Introducing the Courant number, $C_n = ah/\Delta x$, we have $\lambda h = -iC_n\kappa^*\Delta x$. Thus one can obtain values of the principal root over the range $0 \leq \kappa\Delta x \leq \pi$ for a given value of the Courant number. The above expression for er_ω can be normalized to give the error in the phase speed, as follows

$$er_p = \frac{er_\omega}{\omega h} = 1 + \frac{\tan^{-1}\left[(\sigma_1)_i/(\sigma_1)_r)\right]}{C_n\kappa\Delta x} \tag{6.42}$$

where $\omega = -a\kappa$. A positive value of er_p corresponds to phase lag (the numerical phase speed is too small), while a negative value corresponds to phase lead (the numerical phase speed is too large).

6.6.3 Local Accuracy of the Particular Solution (er_μ)

The numerical error in the particular solution is found by comparing the particular solution of the ODE with that for the OΔE. We have found these to be given by

$$P.S._{(ODE)} = ae^{\mu t} \cdot \frac{1}{(\mu - \lambda)}$$

and

$$P.S._{(O\Delta E)} = ae^{\mu t} \cdot \frac{Q(e^{\mu h})}{P(e^{\mu h})}$$

respectively. For a measure of the *local* error in the particular solution we introduce the definition

$$er_\mu \equiv h\left\{\frac{P.S._{(O\Delta E)}}{P.S._{(ODE)}} - 1\right\} \tag{6.43}$$

The multiplication by h converts the error from a global measure to a local one, so that the order of er_λ and er_μ are consistent. In order to determine the leading error term, Eq. 6.43 can be written in terms of the characteristic and particular polynomials as

$$\boxed{er_\mu = \frac{c_o}{\mu - \lambda} \cdot \left\{ (\mu - \lambda) Q\left(e^{\mu h}\right) - P\left(e^{\mu h}\right) \right\}} \tag{6.44}$$

and expanded in a Taylor series, where

$$c_o = \lim_{h \to 0} \frac{h(\mu - \lambda)}{P(e^{\mu h})}$$

The value of c_o is a method-dependent constant that is often equal to one. If the forcing function is independent of time, μ is equal to zero, and for this case, many numerical methods generate an er_μ that is also zero.

The algebra involved in finding the order of er_μ can be quite tedious. However, this order is quite important in determining the true order of a time-marching method by the process that has been outlined. An illustration of this is given in the section on Runge-Kutta methods.

6.6.4 Time Accuracy For Nonlinear Applications

In practice, time-marching methods are usually applied to nonlinear ODE's, and it is necessary that the advertised order of accuracy be valid for the nonlinear cases as well as for the linear ones. A necessary condition for this to occur is that the local accuracies of *both the transient and the particular solutions be of the same order.* More precisely, a time-marching method is said to be of order k if

$$\begin{aligned} er_\lambda &= c_1 \cdot (\lambda h)^{k_1+1} && (6.45) \\ er_\mu &= c_2 \cdot (\lambda h)^{k_2+1} && (6.46) \\ \text{where } k &= \text{smallest of}(k_1, k_2) && (6.47) \end{aligned}$$

The reader should be aware that this is not sufficient. For example, to derive all of the necessary conditions for the fourth-order Runge-Kutta method presented later in this chapter the derivation must be performed for a nonlinear ODE. However, the analysis based on a linear nonhomogeneous ODE produces the appropriate conditions for the majority of time-marching methods used in CFD.

6.6.5 Global Accuracy

In contrast to the local error measures which have just been discussed, we can also define global error measures. These are useful when we come to the evaluation of time-marching methods for specific purposes. This subject is covered in Chapter 8 after our introduction to stability in Chapter 7.

Suppose we wish to compute some time-accurate phenomenon over a fixed interval of time using a constant time step. We refer to such a computation as an "event". Let T be the fixed time of the event and h be the chosen step size. Then the required number of time steps, is N, given by the relation

$$T = Nh$$

Global error in the transient

A natural extension of er_λ to cover the error in an entire event is given by

$$Er_\lambda \equiv e^{\lambda T} - (\sigma_1(\lambda h))^N \tag{6.48}$$

Global error in amplitude and phase

If the event is periodic, we are more concerned with the global error in amplitude and phase. These are given by

$$Er_a = 1 - \left(\sqrt{(\sigma_1)_r^2 + (\sigma_1)_i^2}\right)^N \tag{6.49}$$

and

$$\begin{aligned} Er_\omega &\equiv N\left[\omega h - \tan^{-1}\left(\frac{(\sigma_1)_i}{(\sigma_1)_r}\right)\right] \\ &= \omega T - N\tan^{-1}\left[(\sigma_1)_i/(\sigma_1)_r\right] \end{aligned} \tag{6.50}$$

Global error in the particular solution

Finally, the global error in the particular solution follows naturally by comparing the solutions to the ODE and the OΔE. It can be measured by

$$Er_\mu \equiv (\mu - \lambda)\frac{Q\left(e^{\mu h}\right)}{P\left(e^{\mu h}\right)} - 1$$

6.7 Linear Multistep Methods

In the previous sections, we have developed the framework of error analysis for time advance methods and have randomly introduced a few methods without addressing motivational, developmental or design issues. In the subsequent sections, we introduce classes of methods along with their associated error analysis. We shall not spend much time on development or design of these methods, since most of them have historic origins from a wide variety of disciplines and applications. The Linear Multistep Methods (LMM's) are probably the most natural extension to time marching of the space differencing schemes introduced in Chapter 3 and can be analyzed for accuracy or designed using the Taylor table approach of Section 3.4.

6.7.1 The General Formulation

When applied to the nonlinear ODE

$$\frac{du}{dt} = u' = \mathrm{F}(u, t)$$

all linear multistep methods can be expressed in the general form

$$\sum_{k=1-K}^{1} \alpha_k u_{n+k} = h \sum_{k=1-K}^{1} \beta_k \mathrm{F}_{n+k} \tag{6.51}$$

where the notation for F is defined in Section 6.1. The methods are said to be linear because the α's and β's are independent of u and n, and they are said to be K-step because K time-levels of data are required to marching the solution one time-step, h. They are *explicit* if $\beta_1 = 0$ and *implicit* otherwise.

When Eq. 6.51 is applied to the representative equation, Eq. 6.8, and the result is expressed in operational form, one finds

$$\left(\sum_{k=1-K}^{1} \alpha_k E^k \right) u_n = h \left(\sum_{k=1-K}^{1} \beta_k E^k \right) (\lambda u_n + a e^{\mu h n}) \tag{6.52}$$

We recall from Section 6.5.2 that a time-marching method when applied to the representative equation must provide a σ-root, labeled σ_1, that approximates $e^{\lambda h}$ through the order of the method. The condition referred to as *consistency* simply means that $\sigma \to 1$ as $h \to 0$, and it is certainly a necessary condition for the accuracy of any time marching method. We can also agree that, to be of any value in time accuracy, a method should at least be first-order accurate, that is $\sigma \to (1 + \lambda h)$ as $h \to 0$. One can show that these conditions are met by any method represented by Eq. 6.51 if

$$\sum_k \alpha_k = 0 \quad \text{and} \quad \sum_k \beta_k = \sum_k (K + k - 1)\alpha_k$$

Since both sides of Eq. 6.51 can be multiplied by an arbitrary constant, these methods are often "normalized" by requiring

$$\sum_k \beta_k = 1$$

Under this condition $c_o = 1$ in Eq. 6.44.

6.7.2 Examples

There are many special explicit and implicit forms of linear multistep methods. Two well-known families of them, referred to as Adams-Bashforth (*explicit*) and Adams-Moulton (*implicit*), can be designed using the Taylor table approach of Section 3.4. The Adams-Moulton family is obtained from Eq. 6.51 with

$$\alpha_1 = 1, \qquad \alpha_0 = -1, \qquad \alpha_k = 0, \qquad k = -1, -2, \cdots \tag{6.53}$$

The Adams-Bashforth family has the same α's with the additional constraint that $\beta_1 = 0$. The three-step Adams-Moulton method can be written in the following form

$$u_{n+1} = u_n + h(\beta_1 u'_{n+1} + \beta_0 u'_n + \beta_{-1} u'_{n-1} + \beta_{-2} u'_{n-2}) \tag{6.54}$$

A Taylor table for Eq. 6.54 can be generated as

	u_n	$h \cdot u'_n$	$h^2 \cdot u''_n$	$h^3 \cdot u'''_n$	$h^4 \cdot u''''_n$
u_{n+1}	1	1	$\frac{1}{2}$	$\frac{1}{6}$	$\frac{1}{24}$
$-u_n$	-1				
$-h\beta_1 u'_{n+1}$		$-\beta_1$	$-\beta_1$	$-\beta_1 \frac{1}{2}$	$-\beta_1 \frac{1}{6}$
$-h\beta_0 u'_n$		$-\beta_0$			
$-h\beta_{-1} u'_{n-1}$		$-\beta_{-1}$	β_{-1}	$-\beta_{-1} \frac{1}{2}$	$\beta_{-1} \frac{1}{6}$
$-h\beta_{-2} u'_{n-2}$		$-(-2)^0 \beta_{-2}$	$-(-2)^1 \beta_{-2}$	$-(-2)^2 \beta_{-2} \frac{1}{2}$	$-(-2)^3 \beta_{-2} \frac{1}{6}$

Table 6.1. Taylor table for the Adams-Moulton three-step linear multistep method.

This leads to the linear system

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 0 & -2 & -4 \\ 3 & 0 & 3 & 12 \\ 4 & 0 & -4 & -32 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_0 \\ \beta_{-1} \\ \beta_{-2} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{6.55}$$

to solve for the β's, resulting in

$$\beta_1 = 9/24, \qquad \beta_0 = 19/24, \qquad \beta_{-1} = -5/24, \qquad \beta_{-2} = 1/24 \tag{6.56}$$

which produces a method which is fourth-order accurate.[4] With $\beta_1 = 0$ one obtains

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & -2 & -4 \\ 0 & 3 & 12 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_{-1} \\ \beta_{-2} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{6.57}$$

[4] Recall from Section 6.5.2 that a kth-order time-marching method has a leading truncation error term which is $O(h^{k+1})$.

giving

$$\beta_0 = 23/12, \qquad \beta_{-1} = -16/12, \qquad \beta_{-2} = 5/12 \tag{6.58}$$

This is the third-order Adams-Bashforth method.

A list of simple methods, some of which are very common in CFD applications, is given below together with identifying names that are sometimes associated with them. In the following material AB(n) and AM(n) are used as abbreviations for the (n)th order Adams-Bashforth and (n)th order Adams-Moulton methods. One can verify that the Adams type schemes given below satisfy Eqs. 6.55 and 6.57 up to the order of the method.

Explicit Methods

$$\begin{array}{lll} u_{n+1} & = u_n + hu'_n & \text{Euler} \\ u_{n+1} & = u_{n-1} + 2hu'_n & \text{Leapfrog} \\ u_{n+1} & = u_n + \frac{1}{2}h\Big[3u'_n - u'_{n-1}\Big] & \text{AB2} \\ u_{n+1} & = u_n + \frac{h}{12}\Big[23u'_n - 16u'_{n-1} + 5u'_{n-2}\Big] & \text{AB3} \end{array}$$

Implicit Methods

$$\begin{array}{lll} u_{n+1} & = u_n + hu'_{n+1} & \text{Implicit Euler} \\ u_{n+1} & = u_n + \frac{1}{2}h\Big[u'_n + u'_{n+1}\Big] & \text{Trapezoidal (AM2)} \\ u_{n+1} & = \frac{1}{3}\Big[4u_n - u_{n-1} + 2hu'_{n+1}\Big] & \text{2nd-order Backward} \\ u_{n+1} & = u_n + \frac{h}{12}\Big[5u'_{n+1} + 8u'_n - u'_{n-1}\Big] & \text{AM3} \end{array}$$

6.7.3 Two-Step Linear Multistep Methods

High resolution CFD problems usually require very large data sets to store the spatial information from which the time derivative is calculated. This limits the interest in multistep methods to about two time levels. The most general two-step linear multistep method (i.e., K=2 in Eq. 6.51), that is at least first-order accurate, can be written as

$$(1+\xi)u_{n+1} = [(1+2\xi)u_n - \xi u_{n-1}] + h\Big[\theta u'_{n+1} + (1-\theta+\varphi)u'_n - \varphi u'_{n-1}\Big] \tag{6.59}$$

Clearly the methods are explicit if $\theta = 0$ and implicit otherwise. A list of methods contained in Eq. 6.59 is given in Table 6.2. Notice that the Adams methods have $\xi = 0$, which corresponds to $\alpha_{-1} = 0$ in Eq. 6.51. Methods with $\xi = -1/2$, which corresponds to $\alpha_0 = 0$ in Eq. 6.51, are known as *Milne* methods.

θ	ξ	φ	Method	Order
0	0	0	Euler	1
1	0	0	Implicit Euler	1
1/2	0	0	Trapezoidal or AM2	2
1	1/2	0	2nd Order Backward	2
3/4	0	−1/4	Adams type	2
1/3	−1/2	−1/3	Lees Type	2
1/2	−1/2	−1/2	Two–step trapezoidal	2
5/9	−1/6	−2/9	A–contractive	2
0	−1/2	0	Leapfrog	2
0	0	1/2	AB2	2
0	−5/6	−1/3	Most accurate explicit	3
1/3	−1/6	0	Third–order implicit	3
5/12	0	1/12	AM3	3
1/6	−1/2	−1/6	Milne	4

Table 6.2. Some linear one- and two-step methods, see Eq. 6.59.

One can show after a little algebra that both er_μ and er_λ are reduced to $0(h^3)$ (i.e., the methods are 2nd-order accurate) if

$$\varphi = \xi - \theta + \frac{1}{2}$$

The class of all 3rd-order methods is determined by imposing the additional constraint

$$\xi = 2\theta - \frac{5}{6}$$

Finally a unique fourth-order method is found by setting $\theta = -\varphi = -\xi/3 = \frac{1}{6}$.

6.8 Predictor-Corrector Methods

There are a wide variety of predictor-corrector schemes created and used for a variety of purposes. Their use in solving ODE's is relatively easy to illustrate and understand. Their use in solving PDE's can be much more subtle and demands concepts[5] which have no counterpart in the analysis of ODE's.

Predictor-corrector methods constructed to time-march linear or nonlinear ODE's are composed of sequences of linear multistep methods, each of which is referred to

[5] Such as alternating direction, fractional-step, and hybrid methods.

as a family in the solution process. There may be many families in the sequence, and usually the final family has a higher Taylor-series order of accuracy than the intermediate ones. Their use is motivated by ease of application and increased efficiency, where measures of efficiency are discussed in the next two chapters.

A simple one-predictor, one-corrector example is given by

$$\begin{aligned} \tilde{u}_{n+\alpha} &= u_n + \alpha h u'_n \\ u_{n+1} &= u_n + h\Big[\beta \tilde{u}'_{n+\alpha} + \gamma u'_n\Big] \end{aligned} \tag{6.60}$$

where the parameters α, β and γ are arbitrary parameters to be determined. One can analyze this sequence by applying it to the representative equation and using the operational techniques outlined in Section 6.4. It is easy to show, following the example leading to Eq. 6.26, that

$$P(E) = E^\alpha \cdot \Big[E - 1 - (\gamma + \beta)\lambda h - \alpha\beta\lambda^2 h^2\Big] \tag{6.61}$$

$$Q(E) = E^\alpha \cdot h \cdot [\beta E^\alpha + \gamma + \alpha\beta\lambda h] \tag{6.62}$$

Considering only local accuracy, one is led, by following the discussion in Section 6.6, to the following observations. For the method to be second-order accurate *both* er_λ and er_μ must be $O(h^3)$. For this to hold for er_λ, it is obvious from Eq. 6.61 that

$$\gamma + \beta = 1 \quad ; \quad \alpha\beta = \frac{1}{2}$$

which provides two equations for three unknowns. The situation for er_μ requires some algebra, but it is not difficult to show using Eq. 6.44 that the same conditions also make it $O(h^3)$. One concludes, therefore, that the predictor-corrector sequence

$$\begin{aligned} \tilde{u}_{n+\alpha} &= u_n + \alpha h u'_n \\ u_{n+1} &= u_n + \frac{1}{2}h\Big[\Big(\frac{1}{\alpha}\Big)\tilde{u}'_{n+\alpha} + \Big(\frac{2\alpha - 1}{\alpha}\Big)u'_n\Big] \end{aligned} \tag{6.63}$$

is a second-order accurate method for any α.

A classical predictor-corrector sequence is formed by following an Adams-Bashforth predictor of any order with an Adams-Moulton corrector having an order one higher. The order of the combination is then equal to the order of the corrector. If the order of the corrector is (k), we refer to these as ABM(k) methods. The Adams-Bashforth-Moulton sequence for $k = 3$ is

$$\begin{aligned} \tilde{u}_{n+1} &= u_n + \frac{1}{2}h\Big[3u'_n - u'_{n-1}\Big] \\ u_{n+1} &= u_n + \frac{h}{12}\Big[5\tilde{u}'_{n+1} + 8u'_n - u'_{n-1}\Big] \end{aligned} \tag{6.64}$$

Some simple, specific, second-order accurate methods are given below. The Gazdag method, which we discuss in Chapter 8, is

$$\begin{aligned}\tilde{u}_{n+1} &= u_n + \frac{1}{2}h\Big[3\tilde{u}'_n - \tilde{u}'_{n-1}\Big] \\ u_{n+1} &= u_n + \frac{1}{2}h\Big[\tilde{u}'_n + \tilde{u}'_{n+1}\Big]\end{aligned} \tag{6.65}$$

The Burstein method, obtained from Eq. 6.63 with $\alpha = 1/2$ is

$$\begin{aligned}\tilde{u}_{n+1/2} &= u_n + \frac{1}{2}hu'_n \\ u_{n+1} &= u_n + h\tilde{u}'_{n+1/2}\end{aligned} \tag{6.66}$$

and, finally, MacCormack's method, presented earlier in this chapter, is

$$\begin{aligned}\tilde{u}_{n+1} &= u_n + hu'_n \\ u_{n+1} &= \frac{1}{2}[u_n + \tilde{u}_{n+1} + h\tilde{u}'_{n+1}]\end{aligned} \tag{6.67}$$

Note that MacCormack's method can also be written as

$$\begin{aligned}\tilde{u}_{n+1} &= u_n + hu'_n \\ u_{n+1} &= u_n + \frac{1}{2}h[u'_n + \tilde{u}'_{n+1}]\end{aligned} \tag{6.68}$$

from which it is clear that it is obtained from Eq. 6.63 with $\alpha = 1$.

6.9 Runge-Kutta Methods

There is a special subset of predictor-corrector methods, referred to as Runge-Kutta methods,[6] that produce just one σ-root for each λ-root such that $\sigma(\lambda h)$ corresponds to the Taylor series expansion of $e^{\lambda h}$ out through the order of the method and then truncates. Thus for a Runge-Kutta method of order k (up to 4th order), the principal (and only) σ-root is given by

$$\sigma = 1 + \lambda h + \frac{1}{2}\lambda^2 h^2 + \cdots + \frac{1}{k!}\lambda^k h^k \tag{6.69}$$

It is not particularly difficult to build this property into a method, but, as we pointed out in Section 6.6.4, it is not sufficient to guarantee k'th order accuracy for the solution

[6] Although implicit and multi-step Runge-Kutta methods exist, we will consider only single-step, explicit Runge-Kutta methods here.

of $u' = F(u,t)$ or for the representative equation. To ensure k'th order accuracy, the method must further satisfy the constraint that

$$er_\mu = O(h^{k+1}) \tag{6.70}$$

and this is much more difficult.

The most widely publicized Runge-Kutta process is the one that leads to the fourth-order method. We present it below in some detail. It is usually introduced in the form

$$\begin{aligned}
k_1 &= hF(u_n, t_n)\\
k_2 &= hF(u_n + \beta k_1, t_n + \alpha h)\\
k_3 &= hF(u_n + \beta_1 k_1 + \gamma_1 k_2, t_n + \alpha_1 h)\\
k_4 &= hF(u_n + \beta_2 k_1 + \gamma_2 k_2 + \delta_2 k_3, t_n + \alpha_2 h)
\end{aligned}$$

followed by

$$u(t_n + h) - u(t_n) = \mu_1 k_1 + \mu_2 k_2 + \mu_3 k_3 + \mu_4 k_4 \tag{6.71}$$

However, we prefer to present it using predictor-corrector notation. Thus, a scheme entirely equivalent to 6.71 is

$$\begin{aligned}
\widehat{u}_{n+\alpha} &= u_n + \beta h u'_n\\
\tilde{u}_{n+\alpha_1} &= u_n + \beta_1 h u'_n + \gamma_1 h \widehat{u}'_{n+\alpha}\\
\overline{u}_{n+\alpha_2} &= u_n + \beta_2 h u'_n + \gamma_2 h \widehat{u}'_{n+\alpha} + \delta_2 h \tilde{u}'_{n+\alpha_1}\\
u_{n+1} &= u_n + \mu_1 h u'_n + \mu_2 h \widehat{u}'_{n+\alpha} + \mu_3 h \tilde{u}'_{n+\alpha_1} + \mu_4 h \overline{u}'_{n+\alpha_2}
\end{aligned} \tag{6.72}$$

Appearing in Eqs. 6.71 and 6.72 are a total of 13 parameters which are to be determined such that the method is fourth-order according to the requirements in Eqs. 6.69 and 6.70. First of all, the choices for the time samplings, α, α_1, and α_2, are not arbitrary. They must satisfy the relations

$$\begin{aligned}
\alpha &= \beta\\
\alpha_1 &= \beta_1 + \gamma_1\\
\alpha_2 &= \beta_2 + \gamma_2 + \delta_2
\end{aligned} \tag{6.73}$$

The algebra involved in finding algebraic equations for the remaining 10 parameters is not trivial, but the equations follow directly from finding $P(E)$ and $Q(E)$ and then satisfying the conditions in Eqs. 6.69 and 6.70. Using Eq. 6.73 to eliminate the β's we find from Eq. 6.69 the four conditions

$$\begin{array}{rcll}
\mu_1 + \mu_2 + \mu_3 + \mu_4 &=& 1 & (1)\\
\mu_2\alpha + \mu_3\alpha_1 + \mu_4\alpha_2 &=& 1/2 & (2)\\
\mu_3\alpha\gamma_1 + \mu_4(\alpha\gamma_2 + \alpha_1\delta_2) &=& 1/6 & (3)\\
\mu_4\alpha\gamma_1\delta_2 &=& 1/24 & (4)
\end{array} \tag{6.74}$$

These four relations guarantee that the five terms in σ exactly match the first 5 terms in the expansion of $e^{\lambda h}$. To satisfy the condition that $er_\mu = O(h^5)$, we have to fulfill four more conditions

$$\begin{array}{rcll}
\mu_2\alpha^2 + \mu_3\alpha_1^2 + \mu_4\alpha_2^2 & = & 1/3 & (3) \\
\mu_2\alpha^3 + \mu_3\alpha_1^3 + \mu_4\alpha_2^3 & = & 1/4 & (4) \\
\mu_3\alpha^2\gamma_1 + \mu_4(\alpha^2\gamma_2 + \alpha_1^2\delta_2) & = & 1/12 & (4) \\
\mu_3\alpha\alpha_1\gamma_1 + \mu_4\alpha_2(\alpha\gamma_2 + \alpha_1\delta_2) & = & 1/8 & (4)
\end{array} \tag{6.75}$$

The number in parentheses at the end of each equation indicates the order that is the basis for the equation. Thus if the first 3 equations in 6.74 and the first equation in 6.75 are all satisfied, the resulting method would be third-order accurate. As discussed in Section 6.6.4, the fourth condition in Eq. 6.75 cannot be derived using the methodology presented here, which is based on a linear nonhomogenous representative ODE. A more general derivation based on a nonlinear ODE can be found in several books.[7]

There are eight equations in 6.74 and 6.75 which must be satisfied by the 10 unknowns. Since the equations are overdetermined, two parameters can be set arbitrarily. Several choices for the parameters have been proposed, but the most popular one is due to Runge. It results in the "standard" fourth-order Runge-Kutta method expressed in predictor-corrector form as

$$\begin{aligned}
\hat{u}_{n+1/2} &= u_n + \frac{1}{2}hu'_n \\
\tilde{u}_{n+1/2} &= u_n + \frac{1}{2}h\hat{u}'_{n+1/2} \\
\overline{u}_{n+1} &= u_n + h\tilde{u}'_{n+1/2} \\
u_{n+1} &= u_n + \frac{1}{6}h\left[u'_n + 2\left(\hat{u}'_{n+1/2} + \tilde{u}'_{n+1/2}\right) + \overline{u}'_{n+1}\right]
\end{aligned} \tag{6.76}$$

Notice that this represents the simple sequence of conventional linear multistep methods referred to, respectively, as

$$\left.\begin{array}{l}
\text{Euler Predictor} \\
\text{Euler Corrector} \\
\text{Leapfrog Predictor} \\
\text{Milne Corrector}
\end{array}\right\} \equiv RK4$$

One can easily show that both the Burstein and the MacCormack methods given by Eqs. 6.66 and 6.67 are second-order Runge-Kutta methods, and third-order methods

[7] The present approach based on a linear inhomogeneous equation provides all of the necessary conditions for Runge-Kutta methods of up to third order.

can be derived from Eqs. 6.72 by setting $\mu_4 = 0$ and satisfying only Eqs. 6.74 and the first equation in 6.75. It is clear that for orders one through four, RK methods of order k require k evaluations of the derivative function to advance the solution one time step. We shall discuss the consequences of this in Chapter 8. Higher-order Runge-Kutta methods can be developed, but they require more derivative evaluations than their order. For example, a fifth-order method requires six evaluations to advance the solution one step. In any event, storage requirements reduce the usefulness of Runge-Kutta methods of order higher than four for CFD applications.

6.10 Implementation of Implicit Methods

We have presented a wide variety of time-marching methods and shown how to derive their $\lambda - \sigma$ relations. In the next chapter, we will see that these methods can have widely different properties with respect to stability. This leads to various trade-offs which must be considered in selecting a method for a specific application. Our presentation of the time-marching methods in the context of a linear scalar equation obscures some of the issues involved in implementing an implicit method for systems of equations and nonlinear equations. These are covered in this Section.

6.10.1 Application to Systems of Equations

Consider first the numerical solution of our representative ODE

$$u' = \lambda u + ae^{\mu t} \tag{6.77}$$

using the implicit Euler method. Following the steps outlined in Section 6.2, we obtained

$$(1 - \lambda h)u_{n+1} - u_n = he^{\mu h} \cdot ae^{\mu hn} \tag{6.78}$$

Solving for u_{n+1} gives

$$u_{n+1} = \frac{1}{1 - \lambda h}(u_n + he^{\mu h} \cdot ae^{\mu hn}) \tag{6.79}$$

This calculation does not seem particularly onerous in comparison with the application of an explicit method to this ODE, requiring only an additional division.

Now let us apply the implicit Euler method to our generic system of equations given by

$$\vec{u}' = A\vec{u} - \vec{f}(t) \tag{6.80}$$

where $\vec{u}$ and $\vec{f}$ are vectors and we still assume that A is not a function of $\vec{u}$ or t. Now the equivalent to Eq. 6.78 is

$$(I - hA)\vec{u}_{n+1} - \vec{u}_n = -h\vec{f}(t + h) \tag{6.81}$$

or

$$\vec{u}_{n+1} = (I - hA)^{-1}[\vec{u}_n - h\vec{f}(t + h)] \tag{6.82}$$

The inverse is not actually performed, but rather we solve Eq. 6.81 as a linear system of equations. For our one-dimensional examples, the system of equations which must be solved is tridiagonal (e.g., for biconvection, $A = -aB_p(-1, 0, 1)/2\Delta x$), and hence its solution is inexpensive, but in multidimensions the bandwidth can be very large. In general, the cost per time step of an implicit method is larger than that of an explicit method. The primary area of application of implicit methods is in the solution of *stiff* ODE's, as we shall see in Chapter 8.

6.10.2 Application to Nonlinear Equations

Now consider the general *nonlinear* scalar ODE given by

$$\frac{du}{dt} = F(u, t) \tag{6.83}$$

Application of the implicit Euler method gives

$$u_{n+1} = u_n + hF(u_{n+1}, t_{n+1}) \tag{6.84}$$

This is a nonlinear difference equation. As an example, consider the nonlinear ODE

$$\frac{du}{dt} + \frac{1}{2}u^2 = 0 \tag{6.85}$$

solved using implicit Euler time marching, which gives

$$u_{n+1} + h\frac{1}{2}u_{n+1}^2 = u_n \tag{6.86}$$

which requires a nontrivial method to solve for u_{n+1}. There are several different approaches one can take to solving this nonlinear difference equation. An iterative method, such as Newton's method (see below), can be used. In practice, the "initial guess" for this nonlinear problem can be quite close to the solution, since the "initial guess" is simply the solution at the previous time step, which implies that a linearization approach may be quite successful. Such an approach is described in the next Section.

6.10.3 Local Linearization for Scalar Equations

General Development

Let us start the process of local linearization by considering Eq. 6.83. In order to implement the linearization, we expand $F(u,t)$ about some reference point in time. Designate the reference value by t_n and the corresponding value of the dependent variable by u_n. A Taylor series expansion about these reference quantities gives

$$\begin{aligned} F(u,t) \;&=\; F(u_n,t_n) + \left(\frac{\partial F}{\partial u}\right)_n (u-u_n) + \left(\frac{\partial F}{\partial t}\right)_n (t-t_n) \\ &+\frac{1}{2}\left(\frac{\partial^2 F}{\partial u^2}\right)_n (u-u_n)^2 + \left(\frac{\partial^2 F}{\partial u \partial t}\right)_n (u-u_n)(t-t_n) \\ &+\frac{1}{2}\left(\frac{\partial^2 F}{\partial t^2}\right)_n (t-t_n)^2 + \cdots \end{aligned} \tag{6.87}$$

On the other hand, the expansion of $u(t)$ in terms of the independent variable t is

$$u(t) = u_n + (t-t_n)\left(\frac{\partial u}{\partial t}\right)_n + \frac{1}{2}(t-t_n)^2\left(\frac{\partial^2 u}{\partial t^2}\right)_n + \cdots \tag{6.88}$$

If t is within h of t_n, both $(t-t_n)^k$ and $(u-u_n)^k$ are $O(h^k)$, and Eq. 6.87 can be written

$$F(u,t) = F_n + \left(\frac{\partial F}{\partial u}\right)_n (u-u_n) + \left(\frac{\partial F}{\partial t}\right)_n (t-t_n) + O(h^2) \tag{6.89}$$

Notice that this is an expansion of the *derivative* of the function. Thus, relative to the order of expansion of the function, it represents a second-order-accurate, locally-linear approximation to $F(u,t)$ that is valid in the vicinity of the reference station t_n and the corresponding $u_n = u(t_n)$. With this we obtain the locally (in the neighborhood of t_n) time-linear representation of Eq. 6.83, namely

$$\frac{du}{dt} = \left(\frac{\partial F}{\partial u}\right)_n u + \left(F_n - \left(\frac{\partial F}{\partial u}\right)_n u_n\right) + \left(\frac{\partial F}{\partial t}\right)_n (t-t_n) + O(h^2) \tag{6.90}$$

Implementation of the Trapezoidal Method

As an example of how such an expansion can be used, consider the mechanics of applying the trapezoidal method for the time integration of Eq. 6.83. The trapezoidal method is given by

$$u_{n+1} = u_n + \frac{1}{2}h[F_{n+1} + F_n] + hO(h^2) \tag{6.91}$$

where we write $hO(h^2)$ to emphasize that the method is second order accurate. Using Eq. 6.89 to evaluate $F_{n+1} = F(u_{n+1}, t_{n+1})$, one finds

$$\begin{aligned} u_{n+1} &= u_n + \frac{1}{2}h\left[F_n + \left(\frac{\partial F}{\partial u}\right)_n (u_{n+1} - u_n) + h\left(\frac{\partial F}{\partial t}\right)_n + O(h^2) + F_n\right] \\ &\quad + hO(h^2) \end{aligned} \tag{6.92}$$

Note that the $O(h^2)$ term within the brackets (which is due to the local linearization) is multiplied by h and therefore is the same order as the $hO(h^2)$ error from the trapezoidal method. The use of local time linearization updated at the end of each time step, and the trapezoidal time march, combine to make a *second-order-accurate* numerical integration process. There are, of course, other second-order implicit time-marching methods that can be used. The important point to be made here is that *local linearization updated at each time step has not reduced the order of accuracy* of a second-order time-marching process.

A very useful reordering of the terms in Eq. 6.92 results in the expression

$$\left[1 - \frac{1}{2}h\left(\frac{\partial F}{\partial u}\right)_n\right]\Delta u_n = hF_n + \frac{1}{2}h^2\left(\frac{\partial F}{\partial t}\right)_n \tag{6.93}$$

which is now in the delta form which will be formally introduced in Section 12.6. In many fluid mechanic applications the nonlinear function F is not an *explicit* function of t. In such cases the partial derivative of $F(u)$ with respect to t is zero and Eq. 6.93 simplifies to the second-order accurate expression

$$\left[1 - \frac{1}{2}h\left(\frac{\partial F}{\partial u}\right)_n\right]\Delta u_n = hF_n \tag{6.94}$$

Notice that the RHS is extremely simple. It is the product of h and the RHS of the basic equation evaluated at the previous time step. In this example, the basic equation was the simple scalar equation 6.83, but for our applications, it is generally the space-differenced form of the steady-state equation of some fluid flow problem.

A numerical time-marching procedure using Eq. 6.94 is usually implemented as follows:

1. Solve for the elements of $h\vec{F}_n$, store them in an array say $\vec{R}$, and save $\vec{u}_n$.

2. Solve for the elements of the matrix multiplying $\Delta\vec{u}_n$ and store in some appropriate manner making use of sparseness or bandedness of the matrix if possible. Let this storage area be referred to as B.

3. Solve the coupled set of linear equations

$$B\Delta\vec{u}_n = \vec{R}$$

for $\Delta\vec{u}_n$. (Very seldom does one find B^{-1} in carrying out this step).

4. Find $\vec{u}_{n+1}$ by adding $\Delta\vec{u}_n$ to $\vec{u}_n$, thus

$$\vec{u}_{n+1} = \Delta\vec{u}_n + \vec{u}_n$$

The solution for $\vec{u}_{n+1}$ is generally stored such that it overwrites the value of $\vec{u}_n$ and the process is repeated.

Implementation of the Implicit Euler Method

We have seen that the first-order implicit Euler method can be written

$$u_{n+1} = u_n + hF_{n+1} \tag{6.95}$$

if we introduce Eq. 6.90 into this method, rearrange terms, and remove the explicit dependence on time, we arrive at the form

$$\left[1 - h\left(\frac{\partial F}{\partial u}\right)_n\right]\Delta u_n = hF_n \tag{6.96}$$

We see that the only difference between the implementation of the trapezoidal method and the implicit Euler method is the factor of $\frac{1}{2}$ in the brackets of the left side of Eqs. 6.94 and 6.96. Omission of this factor degrades the method in time accuracy by one order of h. We shall see later that this method is an excellent choice for steady problems.

Newton's Method

Consider the limit $h \rightarrow \infty$ of Eq. 6.96 obtained by dividing both sides by h and setting $1/h = 0$. There results

$$-\left(\frac{\partial F}{\partial u}\right)_n \Delta u_n = F_n \tag{6.97}$$

or

$$u_{n+1} = u_n - \left[\left(\frac{\partial F}{\partial u}\right)_n\right]^{-1} F_n \tag{6.98}$$

This is the well-known Newton method for finding the roots of a nonlinear equation $F(u) = 0$. The fact that it has quadratic convergence is verified by a glance at Eqs. 6.87 and 6.88 (remember the dependence on t has been eliminated for this case). By quadratic convergence, we mean that the error after a given iteration is proportional to the square of the error at the previous iteration, where the error is the difference between the current solution and the converged solution. Quadratic convergence is thus a very powerful property. Use of a finite value of h in Eq. 6.96 leads to linear convergence, i.e., the error at a given iteration is some multiple of the error at the previous iteration. The reader should ponder the meaning of letting $h \rightarrow \infty$ for the trapezoidal method, given by Eq. 6.94.

6.10.4 Local Linearization for Coupled Sets of Nonlinear Equations

In order to present this concept, let us consider an example involving some simple boundary-layer equations. We choose the Falkner-Skan equations from classical boundary-layer theory. Our task is to apply the implicit trapezoidal method to the equations

$$\frac{d^3 f}{dt^3} + f\frac{d^2 f}{dt^2} + \beta\left(1 - \left(\frac{df}{dt}\right)^2\right) = 0 \tag{6.99}$$

Here f represents a dimensionless stream function, and β is a scaling factor.

First of all we reduce Eq. 6.99 to a set of first-order nonlinear equations by the transformations

$$u_1 = \frac{d^2 f}{dt^2} \quad , \quad u_2 = \frac{df}{dt} \quad , \quad u_3 = f \tag{6.100}$$

This gives the coupled set of three nonlinear equations

$$
\begin{aligned}
u_1' &= F_1 = -u_1\, u_3 - \beta\left(1 - u_2^2\right) \\
u_2' &= F_2 = u_1 \\
u_3' &= F_3 = u_2
\end{aligned} \tag{6.101}
$$

and these can be represented in vector notation as

$$
\frac{d\vec{u}}{dt} = \vec{F}(\vec{u}) \tag{6.102}
$$

Now we seek to make the same local expansion that derived Eq. 6.90, except that this time we are faced with a nonlinear vector function, rather than a simple nonlinear scalar function. The required extension requires the evaluation of a matrix, called the Jacobian matrix.[8] Let us refer to this matrix as A. It is derived from Eq. 6.102 by the following process

$$
A = (a_{ij}) = \partial F_i \,/ \partial u_j \tag{6.103}
$$

For the general case involving a third order matrix this is

$$
A = \begin{bmatrix}
\frac{\partial F_1}{\partial u_1} & \frac{\partial F_1}{\partial u_2} & \frac{\partial F_1}{\partial u_3} \\[2ex]
\frac{\partial F_2}{\partial u_1} & \frac{\partial F_2}{\partial u_2} & \frac{\partial F_2}{\partial u_3} \\[2ex]
\frac{\partial F_3}{\partial u_1} & \frac{\partial F_3}{\partial u_2} & \frac{\partial F_3}{\partial u_3}
\end{bmatrix} \tag{6.104}
$$

The expansion of $\vec{F}(\vec{u})$ about some reference state $\vec{u}_n$ can be expressed in a way similar to the scalar expansion given by eq 6.87. Omitting the explicit dependency on the independent variable t, and defining $\vec{F}_n$ as $\vec{F}(\vec{u}_n)$, one has [9]

$$
\vec{F}(\vec{u}) = \vec{F}_n + A_n\left(\vec{u} - \vec{u}_n\right) + O(h^2) \tag{6.105}
$$

where the argument for $O(h^2)$ is the same as in the derivation of Eq. 6.88. Using this we can write the local linearization of Eq. 6.102 as

[8] Recall that we derived the Jacobian matrices for the two-dimensional Euler equations in Section 2.2

[9] The Taylor series expansion of a vector contains a vector for the first term, a matrix times a vector for the second term, and tensor products for the terms of higher order.

$$\frac{d\vec{u}}{dt} = A_n\vec{u} + \underbrace{\left[\vec{F}_n - A_n\vec{u}_n\right]}_{\text{"constant"}} + O(h^2) \tag{6.106}$$

which is a locally-linear, second-order-accurate approximation to a set of coupled nonlinear ordinary differential equations that is valid for $t \leq t_n + h$. Any first- or second-order time-marching method, explicit or implicit, could be used to integrate the equations without loss in accuracy with respect to order. The number of times, and the manner in which, the terms in the Jacobian matrix are updated as the solution proceeds depends, of course, on the nature of the problem.

Returning to our simple boundary-layer example, which is given by Eq. 6.101, we find the Jacobian matrix to be

$$A = \begin{bmatrix} -u_3 & 2\beta u_2 & -u_1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{6.107}$$

The student should be able to derive results for this example that are equivalent to those given for the scalar case in Eq. 6.93. Thus for the Falkner-Skan equations the trapezoidal method results in

$$\begin{bmatrix} 1 + \frac{h}{2}(u_3)_n & -\beta h(u_2)_n & \frac{h}{2}(u_1)_n \\ -\frac{h}{2} & 1 & 0 \\ 0 & -\frac{h}{2} & 1 \end{bmatrix} \begin{bmatrix} (\Delta u_1)_n \\ (\Delta u_2)_n \\ (\Delta u_3)_n \end{bmatrix} = h \begin{bmatrix} -(u_1u_3)_n - \beta(1-u_2^2)_n \\ (u_1)_n \\ (u_2)_n \end{bmatrix}$$

We find $\vec{u}_{n+1}$ from $\Delta\vec{u}_n + \vec{u}_n$, and the solution is now advanced one step. Re-evaluate the elements using $\vec{u}_{n+1}$ and continue. Without any iterating within a step advance, the solution will be second-order-accurate in time.

6.11 Problems

1. Find an expression for the nth term in the Fibonacci series, which is given by $1, 1, 2, 3, 5, 8, \ldots$ Note that the series can be expressed as the solution to a difference equation of the form $u_{n+1} = u_n + u_{n-1}$. What is u_{25}? (Let the first term given above be u_0.)

2. The trapezoidal method $u_{n+1} = u_n + \frac{1}{2}h(u'_{n+1} + u'_n)$ is used to solve the representative ODE.

   (a) What is the resulting OΔE?

(b) What is its exact solution?

(c) How does the exact steady-state solution of the OΔE compare with the exact steady-state solution of the ODE if $\mu = 0$?

3. The 2nd-order backward method is given by

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

(a) Write the OΔE for the representative equation. Identify the polynomials $P(E)$ and $Q(E)$.

(b) Derive the λ-σ relation. Solve for the σ-roots and identify them as principal or spurious.

(c) Find er_λ and the first two nonvanishing terms in a Taylor series expansion of the spurious root.

(d) Perform a σ-root trace relative to the unit circle for both diffusion and convection.

4. Consider the time-marching scheme given by

$$u_{n+1} = u_{n-1} + \frac{2h}{3}(u'_{n+1} + u'_n + u'_{n-1})$$

(a) Write the OΔE for the representative equation. Identify the polynomials $P(E)$ and $Q(E)$.

(b) Derive the $\lambda - \sigma$ relation.

(c) Find er_λ.

5. Find the difference equation which results from applying the Gazdag predictor-corrector method (Eq. 6.65) to the representative equation. Find the λ-σ relation.

6. Consider the following time-marching method:

$$\begin{aligned} \tilde{u}_{n+1/3} &= u_n + hu'_n/3 \\ \bar{u}_{n+1/2} &= u_n + h\tilde{u}'_{n+1/3}/2 \\ u_{n+1} &= u_n + h\bar{u}'_{n+1/2} \end{aligned}$$

Find the difference equation which results from applying this method to the representative equation. Find the λ-σ relation. Find the solution to the difference equation, including the homogeneous and particular solutions. Find er_λ and er_μ. What order is the homogeneous solution? What order is the particular solution? Find the particular solution if the forcing term is fixed.

7. Write a computer program to solve the one-dimensional linear convection equation with periodic boundary conditions and $a = 1$ on the domain $0 \leq x \leq 1$. Use 2nd-order centered differences in space and a grid of 50 points. For the initial condition, use

$$u(x, 0) = e^{-0.5[(x-0.5)/\sigma]^2}$$

with $\sigma = 0.08$. Use the explicit Euler, 2nd-order Adams-Bashforth (AB2), implicit Euler, trapezoidal, and 4th-order Runge-Kutta methods. For the explicit Euler and AB2 methods, use a Courant number, $ah/\Delta x$, of 0.1; for the other methods, use a Courant number of unity. Plot the solutions obtained at $t = 1$ compared to the exact solution (which is identical to the initial condition).

8. Repeat problem 7 using 4th-order (noncompact) differences in space. Use only 4th-order Runge-Kutta time marching at a Courant number of unity. Show solutions at $t = 1$ and $t = 10$ compared to the exact solution.

9. Using the computer program written for problem 7, compute the solution at $t = 1$ using 2nd-order centered differences in space coupled with the 4th-order Runge-Kutta method for grids of 100, 200, and 400 nodes. On a log-log scale, plot the error given by

$$\sqrt{\sum_{j=1}^{M} \frac{(u_j - u_j^{exact})^2}{M}}$$

where M is the number of grid nodes and u^{exact} is the exact solution. Find the global order of accuracy from the plot.

10. Using the computer program written for problem 8, repeat problem 9 using 4th-order (noncompact) differences in space.

11. Write a computer program to solve the one-dimensional linear convection equation with inflow-outflow boundary conditions and $a = 1$ on the domain $0 \leq x \leq 1$. Let $u(0, t) = \sin \omega t$ with $\omega = 10\pi$. Run until a periodic steady state is reached which is independent of the initial condition and plot your solution compared with the exact solution. Use 2nd-order centered differences in space with a 1st-order backward difference at the outflow boundary (as in Eq. 3.69) together with 4th-order Runge-Kutta time marching. Use grids with 100, 200, and 400 nodes and plot the error vs. the number of grid nodes, as described in problem 9. Find the global order of accuracy.

12. Repeat problem 11 using 4th-order (noncompact) centered differences. Use a third-order forward-biased operator at the inflow boundary (as in Eq. 3.67). At the last grid node, derive and use a 3rd-order backward operator (using nodes $j-3$, $j-2$, $j-1$, and j) and at the second last node, use a 3rd-order backward-biased operator (using nodes $j-2$, $j-1$, j, and $j+1$; see problem 1 in Chapter 3).

13. Using the approach described in Section 6.6.2, find the phase speed error, er_p, and the amplitude error, er_a, for the combination of second-order centered differences and 1st, 2nd, 3rd, and 4th-order Runge-Kutta time-marching at a Courant number of unity. Also plot the phase speed error obtained using exact integration in time, i.e., that obtained using the spatial discretization alone. Note that the required σ-roots for the various Runge-Kutta methods can be deduced from Eq. 6.69, without actually deriving the methods. Explain your results.

Chapter 7

STABILITY OF LINEAR SYSTEMS

A general definition of stability is neither simple nor universal and depends on the particular phenomenon being considered. In the case of the nonlinear ODE's of interest in fluid dynamics, stability is often discussed in terms of fixed points and attractors. In these terms a system is said to be stable in a certain domain if, from within that domain, some norm of its solution is always attracted to the same fixed point. These are important and interesting concepts, but we do not dwell on them in this work. Our basic concern is with time-dependent ODE's and OΔE 's in which the coefficient matrices are independent of both u and t; see Section 4.2. We will refer to such matrices as *stationary*. Chapters 4 and 6 developed the representative forms of ODE's generated from the basic PDE's by the semidiscrete approach, and then the OΔE's generated from the representative ODE's by application of time-marching methods. These equations are represented by

$$\frac{d\vec{u}}{dt} = A\vec{u} - \vec{f}(t) \tag{7.1}$$

and

$$\vec{u}_{n+1} = \mathrm{C}\vec{u}_n - \vec{g}_n \tag{7.2}$$

respectively. For a one-step method, the latter form is obtained by applying a time-marching method to the generic ODE form in a fairly straightforward manner. For example, the explicit Euler method leads to $C = I + hA$, and $\vec{g}_n = h\vec{f}(nh)$. Methods involving two or more steps can always be written in the form of Eq. 7.2 by introducing new dependent variables. Note also that only methods in which the time and space discretizations are treated separately can be written in an intermediate semi-discrete form such as Eq. 7.1. The fully-discrete form, Eq. 7.2 and the associated stability definitions and analysis are applicable to all methods.

7.1 Dependence on the Eigensystem

Our definitions of stability are based entirely on the behavior of the homogeneous parts of Eqs. 7.1 and 7.2. The stability of Eq. 7.1 depends entirely on the eigensystem[1] of A. The stability of Eq. 7.2 can often also be related to the eigensystem of its matrix. However, in this case the situation is not quite so simple since, in our applications to partial differential equations (especially hyperbolic ones), a stability definition can depend on both the time and space differencing. This is discussed in Section 7.4. Analysis of these eigensystems has the important added advantage that it gives an estimate of the *rate* at which a solution approaches a steady-state if a system is stable. Consideration will be given to matrices that have both complete and defective eigensystems, see Section 4.2.3, with a reminder that a complete system can be arbitrarily close to a defective one, in which case practical applications can make the properties of the latter appear to dominate.

If A and C are stationary, we can, in theory at least, estimate their fundamental properties. For example, in Section 4.3.2, we found from our model ODE's for diffusion and periodic convection what could be expected for the eigenvalue spectrums of practical physical problems containing these phenomena. These expectations are referred to many times in the following analysis of stability properties. They are important enough to be summarized by the following:

- For diffusion dominated flows the λ-eigenvalues tend to lie along the negative real axis.

- For periodic convection-dominated flows the λ-eigenvalues tend to lie along the imaginary axis.

In many interesting cases, the eigenvalues of the matrices in Eqs. 7.1 and 7.2 are sufficient to determine the stability. In previous chapters, we designated these eigenvalues as λ_m and σ_m for Eqs. 7.1 and 7.2, respectively, and we will find it convenient to examine the stability of various methods in both the complex λ and complex σ planes.

[1]This is *not* the case if the coefficient matrix depends on t even if it is linear.

7.2 Inherent Stability of ODE's

7.2.1 The Criterion

Here we state the standard stability criterion used for ordinary differential equations.

$$\boxed{\text{For a } \textit{stationary} \text{ matrix } A\text{, Eq. 7.1 is } \textit{inherently stable} \text{ if, when } \vec{f} \text{ is constant, } \vec{u} \text{ remains bounded as } t \to \infty.} \tag{7.3}$$

Note that inherent stability depends only on the transient solution of the ODE's.

7.2.2 Complete Eigensystems

If a matrix has a complete eigensystem, all of its eigenvectors are linearly independent, and the matrix can be diagonalized by a similarity transformation. In such a case it follows at once from Eq. 6.27, for example, that the ODE's are inherently stable if and only if

$$\boxed{\Re(\lambda_m) \le 0 \quad \text{for all} \quad m} \tag{7.4}$$

This states that, for inherent stability, all of the λ eigenvalues must lie on, or to the left of, the imaginary axis in the complex λ plane. This criterion is satisfied for the model ODE's representing both diffusion and biconvection. It should be emphasized (as it is an important practical consideration in convection-dominated systems) that the special case for which $\lambda = \pm i$ is *included* in the domain of stability. In this case it is true that $\vec{u}$ does not decay as $t \to \infty$, but neither does it grow, so the above condition is met. Finally we note that for ODE's with complete eigensystems the eigenvectors play no role in the inherent stability criterion.

7.2.3 Defective Eigensystems

In order to understand the stability of ODE's that have defective eigensystems, we inspect the nature of their solutions in eigenspace. For this we draw on the results in Sections 4.2.3 and especially on Eqs. 4.18 to 4.19 in that section. In an eigenspace related to defective systems the form of the representative equation changes from a single equation to a Jordan block. For example, instead of Eq. 4.45 a typical form of the homogeneous part might be

$$\begin{bmatrix} u_1' \\ u_2' \\ u_3' \end{bmatrix} = \begin{bmatrix} \lambda & & \\ 1 & \lambda & \\ & 1 & \lambda \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

for which one finds the solution

$$\begin{aligned}
u_1(t) &= u_1(0)e^{\lambda t} \\
u_2(t) &= [u_2(0) + u_1(0)t]e^{\lambda t} \\
u_3(t) &= \left[u_3(0) + u_2(0)t + \frac{1}{2}u_1(0)t^2\right]e^{\lambda t}
\end{aligned} \tag{7.5}$$

Inspecting this solution, we see that for such cases condition 7.4 must be modified to the form

$$\boxed{\Re(\lambda_m) < 0 \quad \text{for all} \quad m} \tag{7.6}$$

since for pure imaginary λ, u_2 and u_3 would grow without bound (linearly or quadratically) if $u_2(0) \neq 0$ or $u_1(0) \neq 0$. Theoretically this condition is sufficient for stability in the sense of Statement 7.3 since $t^k e^{-|\epsilon| t} \rightarrow 0$ as $t \rightarrow \infty$ for all non-zero ϵ. However, in practical applications the criterion may be worthless since there may be a very large growth of the polynomial before the exponential "takes over" and brings about the decay. Furthermore, on a computer such a growth might destroy the solution process before it could be terminated.

Note that the stability condition 7.6 *excludes* the imaginary axis which tends to be occupied by the eigenvalues related to biconvection problems. However, condition 7.6 is of little or no practical importance if significant amounts of dissipation are present.

7.3 Numerical Stability of OΔE 's

7.3.1 The Criterion

The OΔE companion to Statement 7.3 is

For a *stationary* matrix C, Eq. 7.2 is *numerically stable* if, when $\vec{g}$ is constant, $\vec{u}_n$ remains bounded as $n \rightarrow \infty$.	(7.7)

We see that numerical stability depends only on the transient solution of the OΔE 's. This definition of stability is sometimes referred to as asymptotic or time stability.

As we stated at the beginning of this chapter, stability definitions are not unique. A definition often used in CFD literature stems from the development of PDE solutions that do not necessarily follow the semidiscrete route. In such cases it is appropriate to consider simultaneously the effects of both the time and space approximations. A time-space domain is fixed and stability is defined in terms of what happens to some norm of the solution within this domain as the mesh intervals go to zero at some constant ratio. We discuss this point of view in Section 7.4.

7.3.2 Complete Eigensystems

Consider a set of O∆E 's governed by a complete eigensystem. The stability criterion, according to the condition set in Eq. 7.7, follows at once from a study of Eq. 6.28 and its companion for multiple σ-roots, Eq. 6.36. Clearly, for such systems a time-marching method is numerically stable if and only if

$$\boxed{|(\sigma_m)_k| \leq 1 \quad \text{for all } m \text{ and } k} \tag{7.8}$$

This condition states that, for numerical stability, all of the σ eigenvalues (both principal and spurious, if there are any) must lie on or inside the unit circle in the complex σ-plane.

This definition of stability for O∆E 's is consistent with the stability definition for ODE's. Again the sensitive case occurs for the periodic-convection model which places the "correct" location of the principal σ–root precisely on the unit circle where the solution is only neutrally stable. Further, for a complete eigensystem, the eigenvectors play no role in the numerical stability assessment.

7.3.3 Defective Eigensystems

The discussion for these systems parallels the discussion for defective ODE's. Examine Eq. 6.18 and note its similarity with Eq. 7.5. We see that for defective O∆E's the required modification to 7.8 is

$$\boxed{|(\sigma_m)_k| < 1 \quad \text{for all } m \text{ and } k} \tag{7.9}$$

since defective systems do not guarantee boundedness for $|\sigma| = 1$, for example in Eq. 7.5 if $|\sigma| = 1$ and either $u_2(0)$ or $u_1(0) \neq 0$ we get linear or quadratic growth.

7.4 Time-Space Stability and Convergence of O∆E's

Let us now examine the concept of stability in a different way. In the previous discussion we considered in some detail the following approach:

1. The PDE's are converted to ODE's by approximating the space derivatives on a finite mesh.

2. Inherent stability of the ODE's is established by guaranteeing that $\Re(\lambda) \leq 0$.

3. Time-march methods are developed which guarantee that $|\sigma(\lambda h)| \leq 1$ and this is taken to be the condition for numerical stability.

This *does* guarantee that a stationary system, generated from a PDE on some *fixed* space mesh, will have a numerical solution that is bounded as $t = nh \to \infty$. This *does not* guarantee that desirable solutions are generated in the time march process as both the time and space mesh intervals approach zero.

Now let us define stability in the time-space sense. First construct a finite time-space domain lying within $0 \leq x \leq L$ and $0 \leq t \leq T$. Cover this domain with a grid that is equispaced in both time and space and fix the mesh *ratio* by the equation[2]

$$c_n = \frac{\Delta t}{\Delta x}$$

Next reduce our OΔE approximation of the PDE to a two-level (i.e., two time-planes) formula in the form of Eq. 7.2. The homogeneous part of this formula is

$$\vec{u}_{n+1} = \mathrm{C}\vec{u}_n \tag{7.10}$$

Eq. 7.10 is said to be stable if any bounded initial vector, $\vec{u}_0$, produces a bounded solution vector, $\vec{u}_n$, as the mesh shrinks to zero for a fixed c_n. This is the classical definition of stability. It is often referred to as Lax or Lax-Richtmyer stability. Clearly as the mesh intervals go to zero, the number of time steps, N, must go to infinity in order to cover the entire fixed domain, so the criterion in 7.7 is a necessary condition for this stability criterion.

The significance of this definition of stability arises through *Lax's Theorem*, which states that, if a numerical method is *stable* (in the sense of Lax) and *consistent* then it is *convergent*. A method is *consistent* if it produces no error (in the Taylor series sense) in the limit as the mesh spacing and the time step go to zero (with c_n fixed, in the hyperbolic case). This is further discussed in Section 7.8. A method is *convergent* if it converges to the exact solution as the mesh spacing and time step go to zero in this manner.[3] Clearly, this is an important property.

Applying simple recursion to Eq. 7.10, we find

$$\vec{u}_n = \mathrm{C}^n\vec{u}_0$$

and using vector and matrix p-norms (see Appendix A) and their inequality relations, we have

$$||\vec{u}_n|| = ||\mathrm{C}^n\vec{u}_0|| \leq ||\mathrm{C}^n|| \cdot ||\vec{u}_0|| \leq ||\mathrm{C}||^n \cdot ||\vec{u}_0|| \tag{7.11}$$

[2]This ratio is appropriate for hyperbolic problems; a different ratio may be needed for parabolic problems such as the model diffusion equation.

[3]In the CFD literature, the word *converge* is used with two entirely different meanings. Here we refer to a numerical solution *converging* to the exact solution of the PDE. Later we will refer to the numerical solution *converging* to a steady state solution.

Since the initial data vector is bounded, the solution vector is bounded if

$$||\mathrm{C}|| \leq 1 \tag{7.12}$$

where $||\mathrm{C}||$ represents any p-norm of C. This is often used as a *sufficient* condition for stability.

Now we need to relate the stability definitions given in Eqs. 7.8 and 7.9 with that given in Eq. 7.12. In Eqs. 7.8 and 7.9, stability is related to the *spectral radius* of C, i.e., its eigenvalue of maximum magnitude. In Eq. 7.12, stability is related to a p-norm of C. It is clear that *the criteria are the same when the spectral radius is a true p-norm.*

Two facts about the relation between spectral radii and matrix norms are well known:

1. The spectral radius of a matrix is its L_2 norm when the matrix is normal, i.e., it commutes with its transpose.

2. The spectral radius is the *lower bound* of all norms.

Furthermore, when C is normal, the second inequality in Eq. 7.11 becomes an equality. In this case, Eq. 7.12 becomes both necessary and sufficient for stability. From these relations we draw two important conclusions about the numerical stability of methods used to solve PDE's.

- The stability criteria in Eqs. 7.8 and 7.12 are identical for stationary systems when the governing matrix is normal. This includes symmetric, asymmetric, and circulant matrices. These criteria are both necessary and sufficient for methods that generate such matrices and depend solely upon the eigenvalues of the matrices.

- If the spectral radius of *any* governing matrix is greater than one, the method is unstable by any criterion. Thus for general matrices, the spectral radius condition is necessary[4] but not sufficient for stability.

[4]Actually the necessary condition is that the spectral radius of C be less than or equal to $1 + O(\Delta t)$, but this distinction is not critical for our purposes here.

7.5 Numerical Stability Concepts in the Complex σ-Plane

7.5.1 σ-Root Traces Relative to the Unit Circle

Whether or not the semi-discrete approach was taken to find the differencing approximation of a set of PDE's, the final difference equations can be represented by

$$\vec{u}_{n+1} = \mathrm{C}\vec{u}_n - \vec{g}_n$$

Furthermore if C has a complete[5] eigensystem, the solution to the homogeneous part can always be expressed as

$$\vec{u}_n = c_1\sigma_1^n\vec{x}_1 + \cdots + c_m\sigma_m^n\vec{x}_m + \cdots + c_M\sigma_M^n\vec{x}_M$$

where the σ_m are the eigenvalues of C. If the semi-discrete approach *is* used, we can find a relation between the σ and the λ eigenvalues. This serves as a very convenient guide as to where we might expect the σ-roots to lie relative to the unit circle in the complex σ-plane. For this reason we will proceed to trace the locus of the σ-roots as a function of the parameter λh for the equations modeling diffusion and periodic convection[6].

Locus of the exact trace

Figure 7.1 shows the exact trace of the σ-root if it is generated by $e^{\lambda h}$ representing either diffusion or biconvection. In both cases the $\bullet$ represents the starting value where $h = 0$ and $\sigma = 1$. For the diffusion model, λh is real and negative. As the magnitude of λh increases, the trace representing the dissipation model heads towards the origin as $\lambda h =\rightarrow -\infty$. On the other hand, for the biconvection model, $\lambda h = i\omega h$ is always imaginary. As the magnitude of ωh increases, the trace representing the biconvection model travels around the circumference of the unit circle, which it never leaves. We must be careful in interpreting σ when it is representing $e^{i\omega h}$. The fact that it *lies on* the unit circle means only that the *amplitude* of the representation is correct, it tells us nothing of the *phase* error (see Eq. 6.41). The phase error relates to the *position on* the unit circle.

[5]The subject of defective eigensystems has been addressed. From now on we will omit further discussion of this special case.

[6]Or, if you like, the parameter h for fixed values of λ equal to -1 and i for the diffusion and biconvection cases, respectively.

Examples of some methods

Now let us compare the exact σ-root traces with some that are produced by actual time-marching methods. Table 7.1 shows the λ-σ relations for a variety of methods. Figures 7.2 and 7.3 illustrate the results produced by various methods when they are applied to the model ODE's for diffusion and periodic-convection, Eqs. 4.4 and 4.5. It is implied that the behavior shown is typical of what will happen if the methods are applied to diffusion- (or dissipation-) dominated or periodic convection-dominated problems as well as what does happen in the model cases. Most of the important possibilities are covered by the illustrations.

1	$\sigma - 1 - \lambda h = 0$	Explicit Euler
2	$\sigma^2 - 2\lambda h\sigma - 1 = 0$	Leapfrog
3	$\sigma^2 - (1 + \frac{3}{2}\lambda h)\sigma + \frac{1}{2}\lambda h = 0$	AB2
4	$\sigma^3 - (1 + \frac{23}{12}\lambda h)\sigma^2 + \frac{16}{12}\lambda h\sigma - \frac{5}{12}\lambda h = 0$	AB3
5	$\sigma^(1 - \lambda h) - 1 = 0$	Implicit Euler
6	$\sigma(1 - \frac{1}{2}\lambda h) - (1 + \frac{1}{2}\lambda h) = 0$	Trapezoidal
7	$\sigma^2(1 - \frac{2}{3}\lambda h) - \frac{4}{3}\sigma + \frac{1}{3} = 0$	2nd O Backward
8	$\sigma^2(1 - \frac{5}{12}\lambda h) - (1 + \frac{8}{12}\lambda h)\sigma + \frac{1}{12}\lambda h = 0$	AM3
9	$\sigma^2 - (1 + \frac{13}{12}\lambda h + \frac{15}{24}\lambda^2 h^2)\sigma + \frac{1}{12}\lambda h(1 + \frac{5}{2}\lambda h) = 0$	ABM3
10	$\sigma^3 - (1 + 2\lambda h)\sigma^2 + \frac{3}{2}\lambda h\sigma - \frac{1}{2}\lambda h = 0$	Gazdag
11	$\sigma - 1 - \lambda h - \frac{1}{2}\lambda^2 h^2 = 0$	RK2
12	$\sigma - 1 - \lambda h - \frac{1}{2}\lambda^2 h^2 - \frac{1}{6}\lambda^3 h^3 - \frac{1}{24}\lambda^4 h^4 = 0$	RK4
13	$\sigma^2(1 - \frac{1}{3}\lambda h) - \frac{4}{3}\lambda h\sigma - (1 + \frac{1}{3}\lambda h) = 0$	Milne 4th

Table 7.1. Some $\lambda - \sigma$ Relations

a. Explicit Euler Method

Figure 7.2 shows results for the explicit Euler method. When used for dissipation-dominated cases it is stable for the range -2$\leq \lambda h \leq$0. (Usually the magnitude of λ has to be estimated and often it is found by trial and error). When used for biconvection the σ- trace falls outside the unit circle for all finite h, and the method has no range of stability in this case.

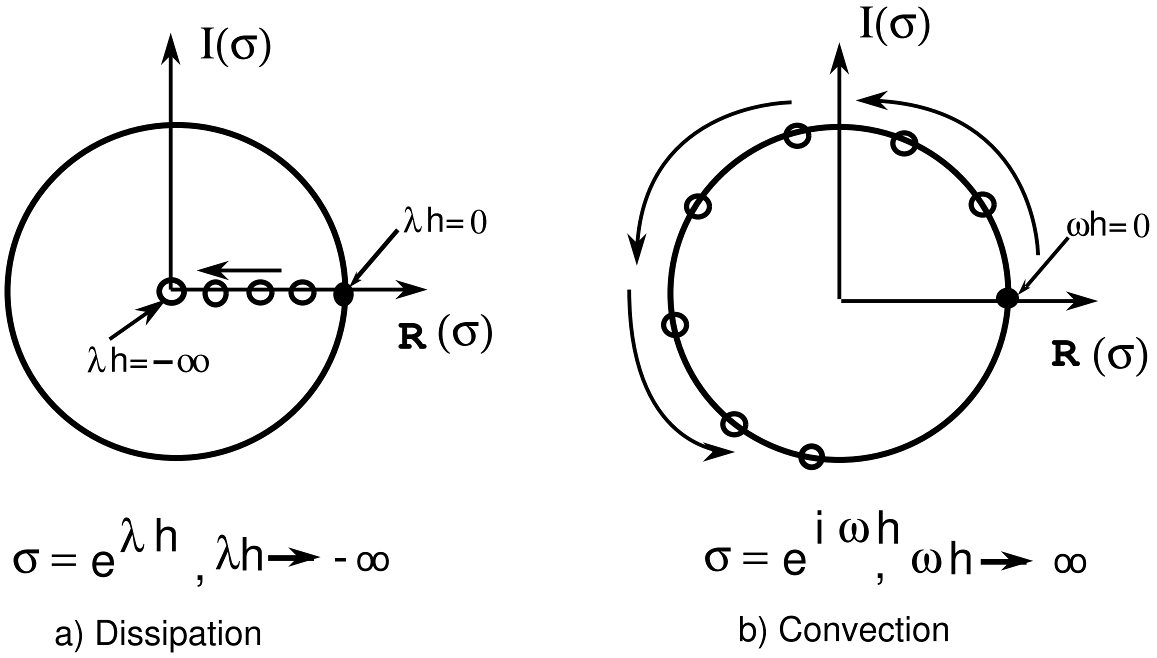

Figure 7.1: Exact traces of σ-roots for model equations.

b. Leapfrog Method

This is a two-root method, since there are two σ's produced by every λ. When applied to dissipation dominated problems we see from Fig. 7.2 that the principal root is stable for a range of λh, but the spurious root is not. In fact, the spurious root starts on the unit circle and falls outside of it for *all* $\Re(\lambda h) < 0$. However, for biconvection cases, when λ is *pure imaginary*, the method is not only stable, but it also produces a σ that falls precisely on the unit circle in the range $0 \leq \omega h \leq 1$. As was pointed out above, this does not mean, that the method is without error. Although the figure shows that there is a range of ωh in which the leapfrog method produces no error in amplitude, it says nothing about the error in *phase*. More is said about this in Chapter 8.

c. Second-Order Adams-Bashforth Method

This is also a two-root method but, unlike the leapfrog scheme, the spurious root starts at the origin, rather than on the unit circle, see Fig. 7.2. Therefore, there is a range of real negative λh for which the method will be stable. The figure shows that the range ends when $\lambda h < -1.0$ since at that point the spurious root leaves the circle and $|\sigma_2|$ becomes greater than one. The situation is quite different when the

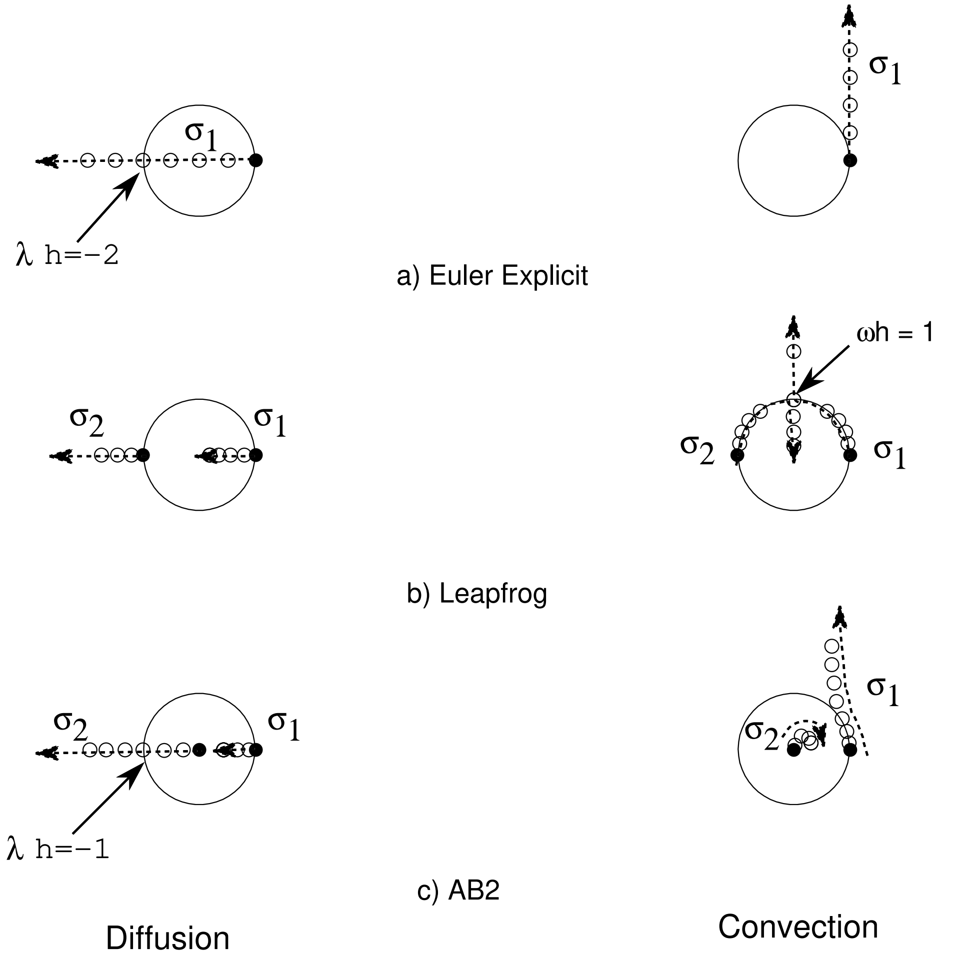

Figure 7.2: Traces of σ-roots for various methods.

λ-root is pure imaginary. In that case as ωh increases away from zero the spurious root remains inside the circle and remains stable for a range of ωh. However, the principal root falls outside the unit circle for all $\omega h > 0$, and for the biconvection model equation the method is unstable for all h.

d. Trapezoidal Method

The trapezoidal method is a very popular one for reasons that are partially illustrated in Fig. 7.3. Its σ-roots fall on or inside the unit circle for both the dissipating and the periodic convecting case and, in fact, *it is stable for all values of λh for which λ itself is inherently stable.* Just like the leapfrog method it has the capability of producing only phase error for the periodic convecting case, but there is a major difference between the two since the trapezoidal method produces no amplitude error for *any* ωh — not just a limited range between $0 \leq \omega h \leq 1$.

e. Gazdag Method

The Gazdag method was designed to produce low phase error. Since its characteristic polynomial for σ is a cubic (Table 7.1, no. 10), it must have two spurious roots in addition to the principal one. These are shown in Fig. 7.3. In both the dissipation and biconvection cases, a spurious root limits the stability. For the dissipating case, a spurious root leaves the unit circle when $\lambda h < -\frac{1}{2}$, and for the biconvecting case, when $\omega h > \frac{2}{3}$. Note that both spurious roots are located at the origin when $\lambda = 0$.

f,g. Second- and Fourth-Order Runge-Kutta Methods, RK2 and RK4

Traces of the σ-roots for the second- and fourth-order Runge-Kutta methods are shown in Fig. 7.3. The figures show that both methods are stable for a range of λh when λh is real and negative, but that the range of stability for RK4 is greater, going almost all the way to -2.8, whereas RK2 is limited to -2. On the other hand for biconvection RK2 is unstable for all ωh, whereas RK4 remains inside the unit circle for $0 \leq \omega h \leq 2\sqrt{2}$. One can show that the RK4 stability limit is about $|\lambda h| < 2.8$ for all complex λh for which $\Re(\lambda) \leq 0$.

7.5.2 Stability for Small Δt

It is interesting to pursue the question of stability when the time step size, h, is small so that accuracy of all the λ-roots is of importance. Situations for which this is not the case are considered in Chapter 8.

Mild instability

All conventional time-marching methods produce a principal root that is very close to $e^{\lambda h}$ for small values of λh. Therefore, on the basis of the principal root, the stability of a method that is required to resolve a transient solution over a relatively short time span may be a moot issue. Such cases are typified by the AB2 and RK2 methods when they are applied to a biconvection problem. Figs. 7.2c and 7.3f show that for both methods the principal root falls outside the unit circle and is unstable for all ωh. However, if the transient solution of interest can be resolved in a limited number of time steps that are small in the sense of the figure, the error caused by this instability may be relatively unimportant. If the root had fallen inside the circle the

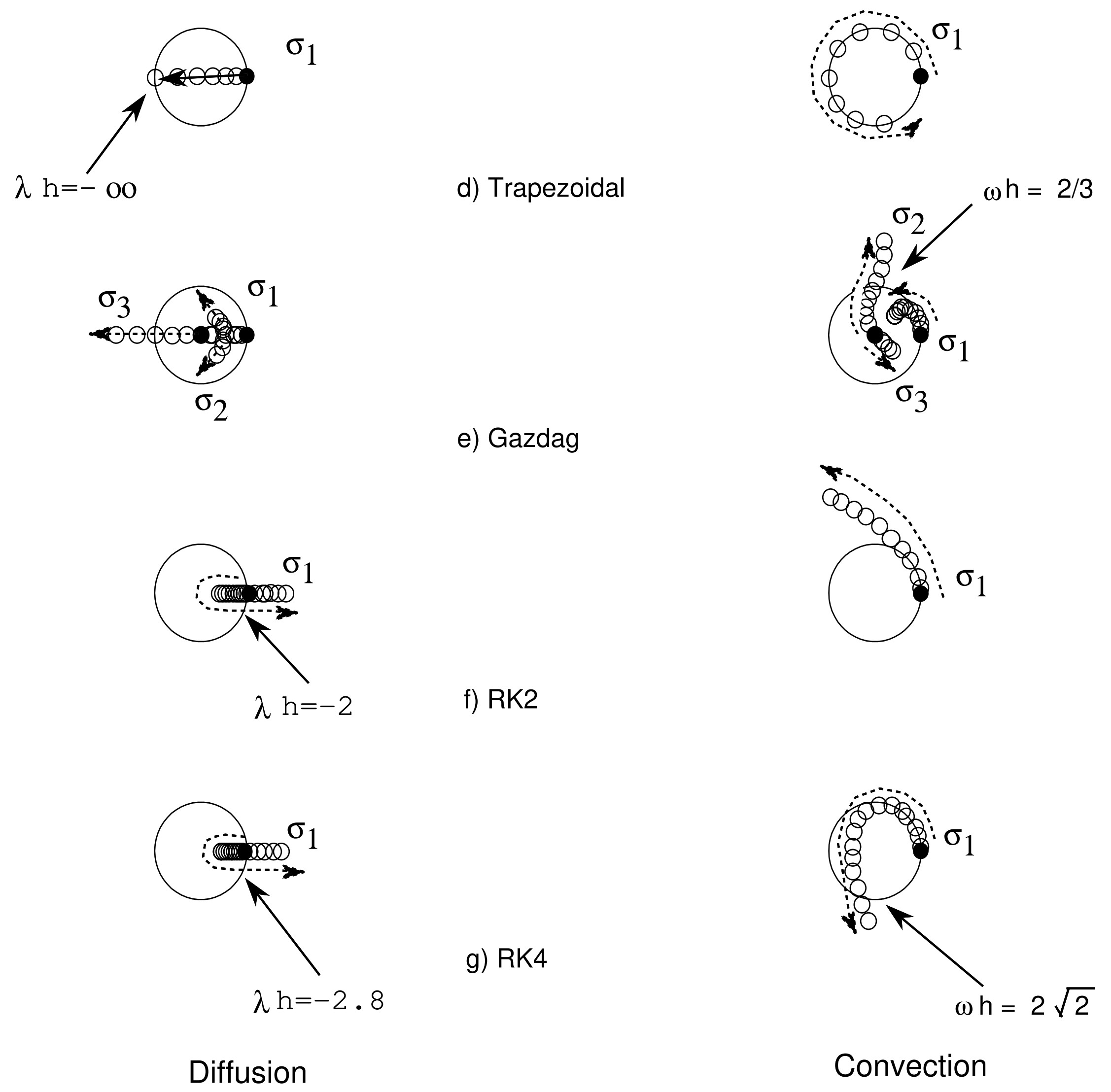

Figure 7.3: Traces of σ-roots for various methods (cont'd).

method would have been declared stable but an error of the same *magnitude* would have been committed, just in the opposite direction. For this reason the AB2 and the RK2 methods have both been used in serious quantitative studies involving periodic convection. This kind of instability is referred to as mild instability and is not a serious problem under the circumstances discussed.

Catastrophic instability

There is a much more serious stability problem for small h that can be brought about by the existence of certain types of spurious roots. One of the best illustrations of this kind of problem stems from a critical study of the most accurate, explicit, two-step, linear multistep method (see Table 7.1):

$$u_{n+1} = -4u_n + 5u_{n-1} + 2h\Big(2u_n' + u_{n-1}'\Big) \tag{7.13}$$

One can show, using the methods given in Section 6.6, that this method is third-order accurate both in terms of er_λ and er_μ, so from an accuracy point of view it is attractive. However, let us inspect its stability even for very small values of λh. This can easily be accomplished by studying its characteristic polynomial when $\lambda h \to 0$. From Eq. 7.13 it follows that for $\lambda h = 0$, $P(E) = E^2 + 4E - 5$. Factoring $P(\sigma) = 0$ we find $P(\sigma) = (\sigma - 1)(\sigma + 5) = 0$. There are two σ-roots; σ_1, the principal one, equal to 1, and σ_2, a spurious one, equal to -5!!

In order to evaluate the consequences of this result, one must understand how methods with spurious roots work in practice. We know that they are not self starting, and the special procedures chosen to start them initialize the coefficients of the spurious roots, the c_{mk} for $k > 1$ in Eq. 6.36. If the starting process is well designed these coefficients are forced to be very small, and if the method is stable, they get smaller with increasing n. However, if the magnitude of one of the spurious σ is equal to 5, one can see disaster is imminent because $(5)^{10} \approx 10^7$. Even a very small initial value of c_{mk} is quickly overwhelmed. Such methods are called catastrophically unstable and are worthless for most, if not all, computations.

Milne and Adams type methods

If we inspect the σ-root traces of the multiple root methods in Figs. 7.2 and 7.3, we find them to be of two types. One type is typified by the leapfrog method. In this case a spurious root falls *on the unit circle* when $h \to 0$. The other type is exemplified by the 2nd-order Adams-Bashforth and Gazdag methods. In this case all spurious roots fall *on the origin* when $h \to 0$.

The former type is referred to as a *Milne Method*. Since at least one spurious root for a Milne method always starts on the unit circle, the method is likely to become unstable for some complex λ as h proceeds away from zero. On the basis of a Taylor series expansion, however, these methods are generally the most accurate insofar as they minimize *the coefficient* in the leading term for er_t.

The latter type is referred to as an *Adams Method*. Since for these methods all spurious methods start at the origin for $h = 0$, they have a guaranteed range of stability for small enough h. However, on the basis of the magnitude of the coefficient in the leading Taylor series error term, they suffer, relatively speaking, from accuracy.

For a given amount of computational work, the order of accuracy of the two types is generally equivalent, and stability requirements in CFD applications generally override the (usually small) increase in accuracy provided by a coefficient with lower magnitude.

7.6 Numerical Stability Concepts in the Complex λh Plane

7.6.1 Stability for Large h.

The reason to study stability for small values of h is fairly easy to comprehend. Presumably we are seeking to resolve some transient and, since the accuracy of the transient solution for all of our methods depends on the smallness of the time-step, we seek to make the size of this step as small as possible. On the other hand, the cost of the computation generally depends on the number of steps taken to compute a solution, and to minimize this we wish to make the step size as large as possible. In the compromise, stability can play a part. Aside from ruling out catastrophically unstable methods, however, the situation in which all of the transient terms are resolved constitutes a rather minor role in stability considerations.

By far the most important aspect of numerical stability occurs under conditions when:

- One has inherently stable, coupled systems with λ–eigenvalues having widely separated magnitudes.

or

- We seek only to find a steady-state solution using a path that includes the unwanted transient.

In both of these cases there exist in the eigensystems relatively large values of $|\lambda h|$ associated with eigenvectors that we wish to drive through the solution process without any regard for their individual accuracy in eigenspace. This situation is the major motivation for the study of numerical stability. It leads to the subject of stiffness discussed in the next chapter.

7.6.2 Unconditional Stability, A-Stable Methods

Inherent stability of a set of ODE's was defined in Section 7.2 and, for coupled sets with a complete eigensystem, it amounted to the requirement that the real parts of all λ eigenvalues must lie on, or to the left of, the imaginary axis in the complex λ plane. This serves as an excellent reference frame to discuss and define the general stability features of time-marching methods. For example, we start with the definition:

> A numerical method is *unconditionally stable* if it is stable for all ODE's that are inherently stable.

A method with this property is said to be *A-stable*. A method is A_o*-stable* if the region of stability contains the negative real axis in the complex λh plane, and *I-stable* if it contains the entire imaginary axis. By applying a fairly simple test for A-stability in terms of positive real functions to the class of two-step LMM's given in Section 6.7.3, one finds these methods to be A-stable if and only if

$$\theta \geq \varphi + \frac{1}{2} \tag{7.14}$$

$$\xi \geq -\frac{1}{2} \tag{7.15}$$

$$\xi \leq \theta + \varphi - \frac{1}{2} \tag{7.16}$$

A set of A-stable implicit methods is shown in Table 7.2.

θ	ξ	φ	Method	Order
1	0	0	Implicit Euler	1
1/2	0	0	Trapezoidal	2
1	1/2	0	2nd O Backward	2
3/4	0	$-1/4$	Adams type	2
1/3	$-1/2$	$-1/3$	Lees	2
1/2	$-1/2$	$-1/2$	Two-step trapezoidal	2
5/8	$-1/6$	$-2/9$	A-contractive	2

Table 7.2. Some unconditionally stable (A-stable) implicit methods.

Notice that none of these methods has an accuracy higher than second-order. It can be proved that the order of an A-stable LMM *cannot exceed two*, and, furthermore that of all 2nd-order A-stable methods, the trapezoidal method has the smallest truncation error.

Returning to the stability test using positive real functions one can show that a two-step LMM is A_o-stable if and only if

$$\theta \geq \varphi + \frac{1}{2} \tag{7.17}$$

$$\xi \geq -\frac{1}{2} \tag{7.18}$$

$$0 \leq \theta - \varphi \tag{7.19}$$

For first-order accuracy, the inequalities 7.17 to 7.19 are less stringent than 7.14 to 7.16. For second-order accuracy, however, the parameters (θ, ξ, φ) are related by the condition

$$\varphi = \xi - \theta + \frac{1}{2}$$

and the two sets of inequalities reduce to the same set which is

$$\xi \leq 2\theta - 1 \tag{7.20}$$

$$\xi \geq -\frac{1}{2} \tag{7.21}$$

Hence, two-step, second-order accurate LMM's that are A-stable and A_o-stable share the same (φ, ξ, θ) parameter space. Although the order of accuracy of an A-stable method cannot exceed two, A_o-stable LMM methods exist which have an accuracy of arbitrarily high order.

It has been shown that for a method to be I-stable it must also be A-stable. Therefore, no further discussion is necessary for the special case of I-stability.

It is not difficult to prove that methods having a characteristic polynomial for which the coefficient of the highest order term in E is unity[7] can never be unconditionally stable. This includes all explicit methods and predictor-corrector methods made up of explicit sequences. Such methods are referred to, therefore, as *conditionally stable* methods.

7.6.3 Stability Contours in the Complex λh Plane.

A very convenient way to present the stability properties of a time-marching method is to plot the locus of the complex λh for which $|\sigma| = 1$, such that the resulting

[7] Or can be made equal to unity by a trivial normalization (division by a constant independent of λh). The proof follows from the fact that the coefficients of such a polynomial are sums of various combinations of products of all its roots.

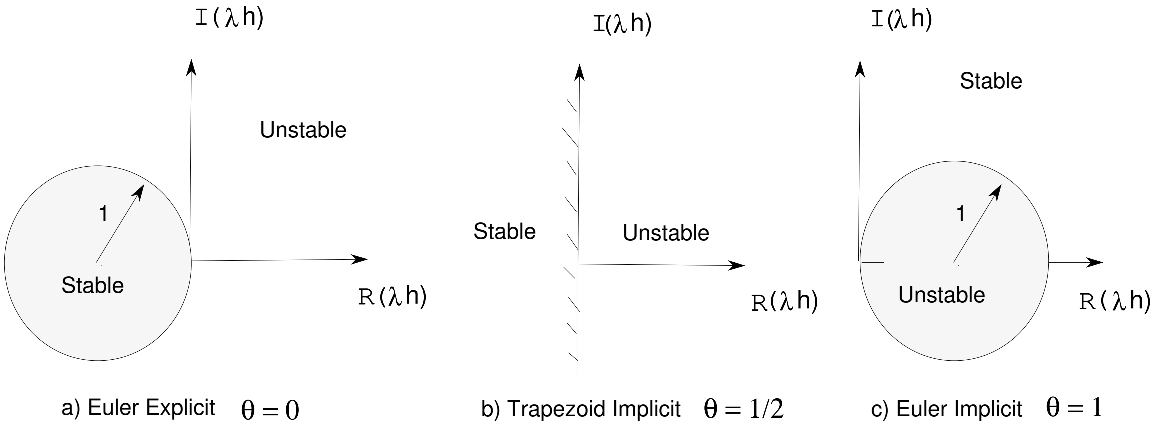

Figure 7.4: Stability contours for the θ-method.

contour goes through the point $\lambda h = 0$. Here $|\sigma|$ refers to the maximum absolute value of any σ, principal or spurious, that is a root to the characteristic polynomial for a given λh. It follows from Section 7.3 that on one side of this contour the numerical method is stable while on the other, it is unstable. We refer to it, therefore, as a *stability contour.*

Typical stability contours for both explicit and implicit methods are illustrated in Fig. 7.4, which is derived from the one-root θ-method given in Section 6.5.4.

Contours for explicit methods

Fig. 7.4a shows the stability contour for the explicit Euler method. In the following two ways it is typical of all stability contours for explicit methods:

1. The contour encloses a finite portion of the left-half complex λh-plane.

2. The region of stability is *inside* the boundary, and therefore, it is conditional.

However, this method includes no part of the imaginary axis (except for the origin) and so it is unstable for the model biconvection problem. Although several explicit methods share this deficiency (e.g., AB2, RK2), several others do not (e.g., leapfrog, Gazdag, RK3, RK4), see Figs. 7.5 and 7.6. Notice in particular that the third- and fourth-order Runge-Kutta methods, Fig. 7.6, include a portion of the imaginary axis out to $\pm 1.9i$ and $\pm 2\sqrt{2}i$, respectively.

Contours for unconditionally stable implicit methods

Fig. 7.4c shows the stability contour for the implicit Euler method. It is typical of many stability contours for unconditionally stable implicit methods. Notice that the

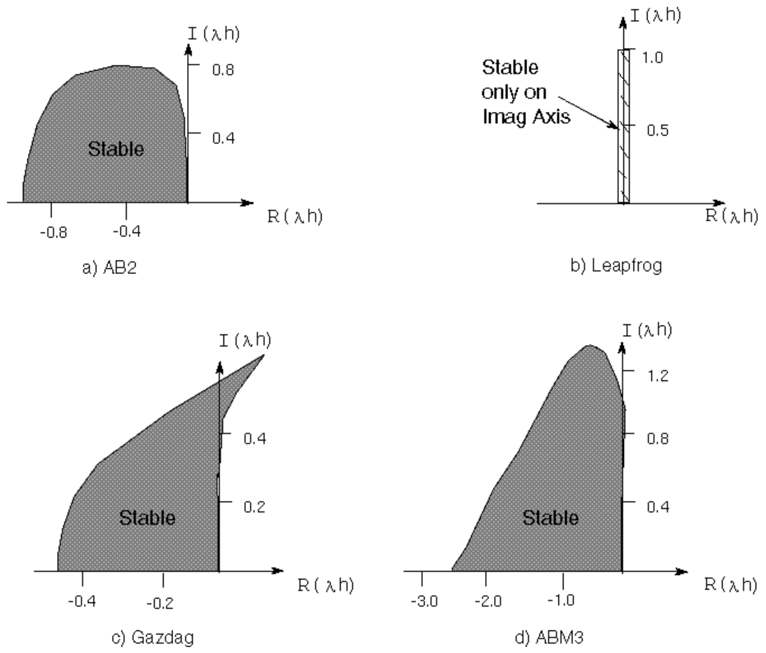

Figure 7.5: Stability contours for some explicit methods.

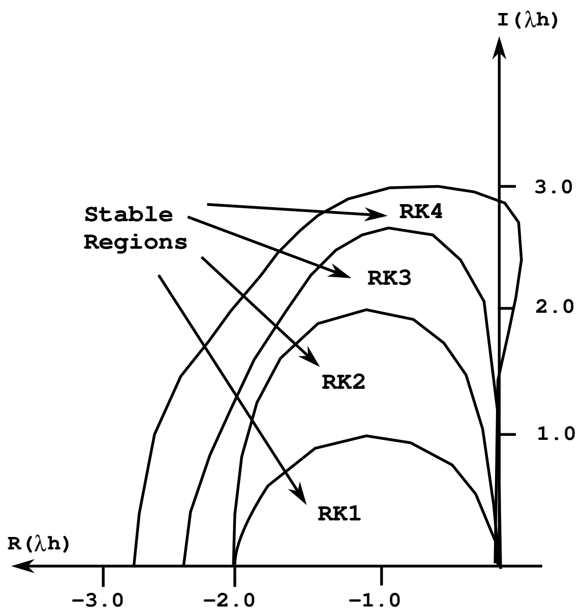

Figure 7.6: Stability contours for Runge-Kutta methods.

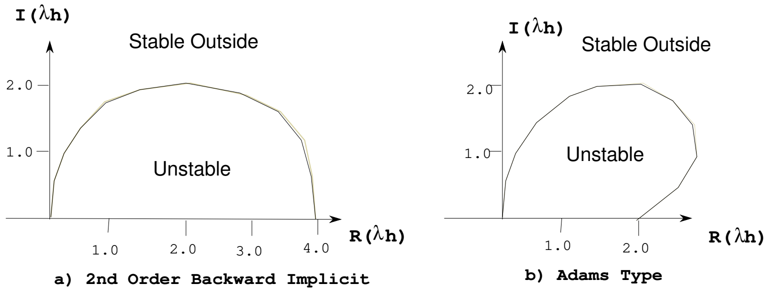

Figure 7.7: Stability contours for the 2 unconditionally stable implicit methods.

method is stable for the entire range of complex λh that fall *outside* the boundary. This means that the method is numerically stable *even when the ODE's that it is being used to integrate are inherently unstable.* Some other implicit unconditionally stable methods with the same property are shown in Fig. 7.7. In all of these cases the imaginary axis is part of the *stable* region.

Not all unconditionally stable methods are stable in some regions where the ODE's they are integrating are inherently unstable. The classic example of a method that is stable *only* when the generating ODE's are themselves inherently stable is the trapezoidal method, i.e., the special case of the θ-method for which $\theta = \frac{1}{2}$. The stability boundary for this case is shown in Fig. 7.4b. The boundary is the imaginary axis and the numerical method is stable for λh lying on or to the left of this axis. Two other methods that have this property are the two-step trapezoidal method

$$u_{n+1} = u_{n-1} + h\Big(u'_{n+1} + u'_{n-1}\Big)$$

and a method due to Lees

$$u_{n+1} = u_{n-1} + \frac{2}{3}h\Big(u'_{n+1} + u'_n + u'_{n-1}\Big)$$

Notice that both of these methods are of the Milne type.

Contours for conditionally stable implicit methods

Just because a method is implicit does not mean that it is unconditionally stable. Two illustrations of this are shown in Fig. 7.8. One of these is the Adams-Moulton 3rd-order method (no. 8, Table 7.1). Another is the 4th-order Milne method given by the point operator

$$u_{n+1} = u_{n-1} + \frac{1}{3}h\Big(u'_{n+1} + 4u'_n + u'_{n-1}\Big)$$

and shown in Table 7.1 as no. 13. It is stable *only* for $\lambda = \pm i\omega$ when $0 \leq \omega \leq \sqrt{3}$. Its stability boundary is very similar to that for the leapfrog method (see Fig. 7.5b).

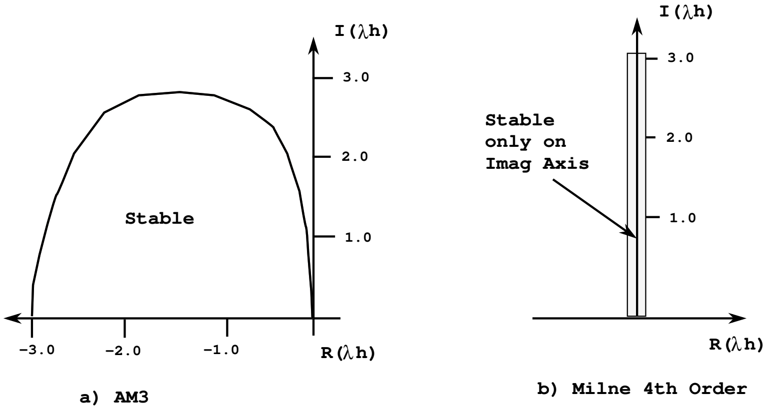

Figure 7.8: Stability contours for the 2 conditionally stable implicit methods.

7.7 Fourier Stability Analysis

By far the most popular form of stability analysis for numerical schemes is the Fourier or von Neumann approach. This analysis is usually carried out on point operators and it does not depend on an intermediate stage of ODE's. *Strictly speaking* it applies only to difference approximations of PDE's that produce OΔE's *which are linear, have no space or time varying coefficients, and have periodic boundary conditions.*[8] In practical application it is often used as a guide for estimating the worthiness of a method for more general problems. It serves as a fairly reliable *necessary* stability condition, but it is by no means a *sufficient* one.

7.7.1 The Basic Procedure

One takes data from a "typical" point in the flow field and uses this as constant throughout time and space according to the assumptions given above. Then one imposes a spatial harmonic as an initial value on the mesh and asks the question: Will its amplitude grow or decay in time? The answer is determined by finding the conditions under which

$$u(x,t) = e^{\alpha t} \cdot e^{i\kappa x} \tag{7.22}$$

is a solution to the *difference* equation, where κ is real and $\kappa\Delta x$ lies in the range $0 \le \kappa\Delta x \le \pi$. Since, for the general term,

$$u_{j+m}^{(n+\ell)} = e^{\alpha(t+\ell\Delta t)} \cdot e^{i\kappa(x+m\Delta x)} = e^{\alpha\ell\Delta t} \cdot e^{i\kappa m\Delta x} \cdot u_j^{(n)}$$

[8]Another way of viewing this is to consider it as an initial value problem on an infinite space domain.

the quantity $u_j^{(n)}$ is common to every term and can be factored out. In the remaining expressions, we find the term $e^{\alpha \Delta t}$ which we represent by σ, thus:

$$\sigma \equiv e^{\alpha \Delta t}$$

Then, since $e^{\alpha t} = \left(e^{\alpha \Delta t}\right)^n = \sigma^n$, it is clear that

$$\boxed{\text{For numerical stability } |\sigma| \leq 1} \tag{7.23}$$

and the problem is to solve for the σ's produced by any given method and, as a necessary condition for stability, make sure that, in the worst possible combination of parameters, condition 7.23 is satisfied[9].

7.7.2 Some Examples

The procedure can best be explained by examples. Consider as a first example the finite difference approximation to the model diffusion equation known as Richardson's method of overlapping steps. This was mentioned in Chapter 4 and given as Eq. 4.2:

$$u_j^{(n+1)} = u_j^{(n-1)} + \nu \frac{2\Delta t}{\Delta x^2}\left(u_{j+1}^{(n)} - 2u_j^{(n)} + u_{j-1}^{(n)}\right) \tag{7.24}$$

Substitution of Eq. 7.22 into Eq. 7.24 gives the relation

$$\sigma = \sigma^{-1} + \nu \frac{2\Delta t}{\Delta x^2}\left(e^{i\kappa \Delta x} - 2 + e^{-i\kappa \Delta x}\right)$$

or

$$\sigma^2 + \underbrace{\left[\frac{4\nu \Delta t}{\Delta x^2}(1 - \cos \kappa \Delta x)\right]}_{2b} \sigma - 1 = 0 \tag{7.25}$$

Thus Eq. 7.22 is a solution of Eq. 7.24 if σ is a root of Eq. 7.25. The two roots of 7.25 are

$$\sigma_{1,2} = -b \pm \sqrt{b^2 + 1}$$

from which it is clear that one $|\sigma|$ is always > 1. We find, therefore, that by the Fourier stability test, Richardson's method of overlapping steps is unstable for all ν, κ and Δt.

[9]If boundedness is required in a *finite* time domain, the condition is often presented as $|\sigma| \leq 1 + O(\Delta t)$.

As another example consider the finite-difference approximation for the model biconvection equation

$$u_j^{(n+1)} = u_j^{(n)} - \frac{a\Delta t}{2\Delta x}\left(u_{j+1}^{(n)} - u_{j-1}^{(n)}\right) \tag{7.26}$$

In this case

$$\sigma = 1 - \frac{a\Delta t}{\Delta x} \cdot i \cdot \sin \kappa \Delta x$$

from which it is clear that $|\sigma| > 1$ for all nonzero a and κ. Thus we have another finite-difference approximation that, by the Fourier stability test, is unstable for any choice of the free parameters.

7.7.3 Relation to Circulant Matrices

The underlying assumption in a Fourier stability analysis is that the C matrix, determined when the differencing scheme is put in the form of Eq. 7.2, is circulant. Such being the case, the $e^{i\kappa x}$ in Eq. 7.22 represents an *eigenvector* of the system, and the two examples just presented outline a simple procedure for finding the *eigenvalues* of the circulant matrices formed by application of the two methods to the model problems. The choice of σ for the stability parameter in the Fourier analysis, therefore, is not an accident. It is exactly the same σ we have been using in all of our previous discussions, but arrived at from a different perspective.

If we examine the preceding examples from the viewpoint of circulant matrices and the semi-discrete approach, the results present rather obvious conclusions. The space differencing in Richardson's method produces the matrix $s \cdot B_p(1, -2, 1)$ where s is a positive scalar coefficient. From Appendix B we find that the eigenvalues of this matrix are real negative numbers. Clearly, the time-marching is being carried out by the leapfrog method and, from Fig. 7.5, this method is unstable for *all* eigenvalues with negative real parts. On the other hand, the space matrix in Eq. 7.26 is $B_p(-1, 0, 1)$, and according to Appendix B, this matrix has pure imaginary eigenvalues. However, in this case the explicit Euler method is being used for the time-march and, according to Fig. 7.4, this method is always unstable for such conditions.

7.8 Consistency

Consider the model equation for diffusion analysis

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2} \tag{7.27}$$

Many years before computers became available (1910, in fact), Lewis F. Richardson proposed a method for integrating equations of this type. We presented his method in Eq. 4.2 and analyzed its stability by the Fourier method in Section 7.7.

In Richardson's time, the concept of numerical instability was not known. However, the concept is quite clear today and we now know immediately that his approach would be unstable. As a semi-discrete method it can be expressed in matrix notation as the system of ODE's:

$$\frac{d\vec{u}}{dt} = \frac{\nu}{\Delta x^2} B(1,-2,1)\vec{u} + (\vec{bc}) \tag{7.28}$$

with the leapfrog method used for the time march. Our analysis in this Chapter revealed that this is numerically unstable since the λ-roots of B(1,-2,1) are all real and negative and the spurious σ-root in the leapfrog method is unstable for all such cases, see Fig. 7.5b.

The method was used by Richardson for weather prediction, and this fact can now be a source of some levity. In all probability, however, the hand calculations (the only approach available at the time) were not carried far enough to exhibit strange phenomena. We could, of course, use the 2nd-order Runge-Kutta method to integrate Eq. 7.28 since it is stable for real negative λ's. It is, however, conditionally stable and for this case we are rather severely limited in time step size by the requirement $\Delta t \le \Delta x^2/(2\nu)$.

There are many ways to manipulate the numerical stability of algorithms. One of them is to introduce mixed time and space differencing, a possibility we have not yet considered. For example, we introduced the DuFort-Frankel method in Chapter 4:

$$u_j^{(n+1)} = u_j^{(n-1)} + \frac{2\nu\Delta t}{\Delta x^2}\left[u_{j-1}^{(n)} - 2\left(\frac{u_j^{(n+1)} + u_j^{(n-1)}}{2}\right) + u_{j+1}^{(n)}\right] \tag{7.29}$$

in which the central term in the space derivative in Eq. 4.2 has been replaced by its average value at two different time levels. Now let

$$\alpha \equiv \frac{2\nu\Delta t}{\Delta x^2}$$

and rearrange terms

$$(1+\alpha)u_j^{(n+1)} = (1-\alpha)u_j^{(n-1)} + \alpha\left[u_{j-1}^{(n)} + u_{j+1}^{(n)}\right]$$

There is no obvious ODE between the basic PDE and this final OΔE. Hence, there is no intermediate λ-root structure to inspect. Instead one proceeds immediately to the σ-roots.

The simplest way to carry this out is by means of the Fourier stability analysis introduced in Section 7.7. This leads at once to

$$(1+\alpha)\sigma = (1-\alpha)\sigma^{-1} + \alpha\Big(e^{i\kappa\Delta x} + e^{-i\kappa\Delta x}\Big)$$

or

$$(1+\alpha)\sigma^2 - 2\alpha\sigma\cos(k\Delta x) - (1-\alpha) = 0$$

The solution of the quadratic is

$$\sigma = \frac{\alpha\cos\kappa\Delta x \pm \sqrt{1-\alpha^2\sin^2\kappa\Delta x}}{1+\alpha}$$

There are 2M σ-roots all of which are ≤ 1 for any real α in the range $0 \leq \alpha \leq \infty$. This means that the method is *unconditionally stable*!

The above result seems too good to be true, since we have found an unconditionally stable method using an *explicit* combination of Lagrangian interpolation polynomials.

The price we have paid for this is the loss of *consistency* with the original PDE.

To prove this, we expand the terms in Eq. 7.29 in a Taylor series and reconstruct the partial differential equation we are actually solving as the mesh size becomes very small. For the time derivative we have

$$\frac{1}{2\Delta t}\Big[u_j^{(n+1)} - u_j^{(n-1)}\Big] = (\partial_t u)_j^{(n)} + \frac{1}{6}\Delta t^2(\partial_{ttt}u)_j^{(n)} + \cdots$$

and for the mixed time and space differences

$$\begin{aligned}\frac{u_{j-1}^{(n)} - u_j^{(n+1)} - u_j^{(n-1)} + u_{j+1}^{(n)}}{\Delta x^2} \quad = \quad & (\partial_{xx}u)_j^{(n)} - \left(\frac{\Delta t}{\Delta x}\right)^2(\partial_{tt}u)_j^{(n)} + \\ & \frac{1}{12}\Delta x^2(\partial_{xxxx}u)_j^{(n)} - \\ & \frac{1}{12}\Delta t^2\left(\frac{\Delta t}{\Delta x}\right)^2(\partial_{tttt}u)_j^{(n)} + \cdots \end{aligned} \tag{7.30}$$

Replace the terms in Eq. 7.29 with the above expansions and take the limit as $\Delta t\ ,\ \Delta x \to 0$. We find

$$\frac{\partial u}{\partial t} = \nu\frac{\partial^2 u}{\partial x^2} - \nu r^2\frac{\partial^2 u}{\partial t^2} \tag{7.31}$$

where

$$r \equiv \frac{\Delta t}{\Delta x}$$

Eq. 7.27 is *parabolic*. Eq. 7.31 is *hyperbolic*. Thus if $\Delta t \to 0$ and $\Delta x \to 0$ in such a way that $\frac{\Delta t}{\Delta x}$ remains constant, the equation we actually solve by the method in Eq. 7.29 is a wave equation, not a diffusion equation. In such a case *Eq. 7.29 is not uniformly consistent with the equation we set out to solve* even for vanishingly small step sizes. The situation is summarized in Table 7.3. If $\Delta t \to 0$ and $\Delta x \to 0$ in such a way that $\frac{\Delta t}{\Delta x^2}$ remains constant, then r^2 is $O(\Delta t)$, and the method is consistent. However, this leads to a constraint on the time step which is just as severe as that imposed by the stability condition of the second-order Runge-Kutta method shown in the table.

2nd-order Runge-Kutta		DuFort–Frankel
	For Stability	
$\Delta t \le \frac{\Delta x^2}{2\nu}$		$\Delta t \le \infty$
Conditionally Stable		Unconditionally Stable
	For Consistency	
Uniformly Consistent		Conditionally Consistent,
with $\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2}$		Approximates $\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2}$ only if $\nu \left(\frac{\Delta t}{\Delta x}\right)^2 < \epsilon$ Therefore $\Delta t < \Delta x \sqrt{\frac{\epsilon}{\nu}}$

Table 7.3: Summary of accuracy and consistency conditions for RK2 and Du Fort-Frankel methods. $\epsilon =$ an arbitrary error bound.

7.9 Problems

1. Consider the ODE

$$u' = \frac{du}{dt} = Au + f$$

with

$$A = \begin{bmatrix} -10 & -0.1 & -0.1 \\ 1 & -1 & 1 \\ 10 & 1 & -1 \end{bmatrix} \quad , \quad f = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}$$

(a) Find the eigenvalues of A using a numerical package. What is the steady-state solution? How does the ODE solution behave in time?

(b) Write a code to integrate from the initial condition $u(0) = [1, 1, 1]^T$ from time $t = 0$ using the explicit Euler, implicit Euler, and MacCormack methods. In all three cases, use $h = 0.1$ for 1000 time steps, $h = 0.2$ for 500 time steps, $h = 0.4$ for 250 time steps and $h = 1.0$ for 100 time steps. Compare the computed solution with the exact steady solution.

(c) Using the λ-σ relations for these three methods, what are the expected bounds on h for stability? Are your results consistent with these bounds?

2. (a) Compute a table of the numerical values of the σ-roots of the 2nd-order Adams-Bashforth method when $\lambda = i$. Take h in intervals of 0.05 from 0 to 0.80 and compute the absolute values of the roots to at least 6 places.

   (b) Plot the trace of the roots in the complex σ-plane and draw the unit circle on the same plot.

   (c) Repeat the above for the RK2 method.

3. When applied to the linear convection equation, the widely known Lax–Wendroff method gives:

$$u_j^{n+1} = u_j^n - \frac{1}{2}C_n(u_{j+1}^n - u_{j-1}^n) + \frac{1}{2}C_n^2(u_{j+1}^n - 2u_j^n + u_{j-1}^n)$$

where C_n, known as the Courant (or CFL) number, is $ah/\Delta x$. Using Fourier stability analysis, find the range of C_n for which the method is stable.

4. Determine and plot the stability contours for the Adams-Bashforth methods of order 1 through 4. Compare with the Runge-Kutta methods of order 1 through 4.

5. Recall the compact centered 3-point approximation for a first derivative:

$$(\delta_x u)_{j-1} + 4(\delta_x u)_j + (\delta_x u)_{j+1} = \frac{3}{\Delta x}(u_{j+1} - u_{j-1})$$

By replacing the spatial index j by the temporal index n, obtain a time-marching method using this formula. What order is the method? Is it explicit or implicit? Is it a two-step LMM? If so, to what values of ξ, θ, and ϕ (in Eq. 6.59) does it correspond? Derive the λ-σ relation for the method. Is it A-stable, A_o-stable, or I-stable?

6. Write the ODE system obtained by applying the 2nd-order centered difference approximation to the spatial derivative in the model diffusion equation with periodic boundary conditions. Using Appendix B.4, find the eigenvalues of the spatial operator matrix. Given that the λ-σ relation for the 2nd-order Adams-Bashforth method is

$$\sigma^2 - (1 + 3\lambda h/2)\sigma + \lambda h/2 = 0$$

show that the maximum stable value of $|\lambda h|$ for real negative λ, i.e., the point where the stability contour intersects the negative real axis, is obtained with $\lambda h = -1$. Using the eigenvalues of the spatial operator matrix, find the maximum stable time step for the combination of 2nd-order centered differences and the 2nd-order Adams-Bashforth method applied to the model diffusion equation. Repeat using Fourier analysis.

7. Consider the following PDE:

$$\frac{\partial u}{\partial t} = i\frac{\partial^2 u}{\partial x^2}$$

where $i = \sqrt{-1}$. Which explicit time-marching methods would be suitable for integrating this PDE, if 2nd-order centered differencing is used for the spatial differences and the boundary conditions are periodic? Find the stability condition for one of these methods.

8. Using Fourier analysis, analyze the stability of first-order backward differencing coupled with explicit Euler time marching applied to the linear convection equation with positive a. Find the maximum Courant number for stability.

9. Consider the linear convection with a positive wave speed as in problem 8. Apply a Dirichlet boundary condition at the left boundary. No boundary condition is permitted at the right boundary. Write the system of ODE's which results from first-order backward spatial differencing in matrix-vector form. Using Appendix B.1, find the λ-eigenvalues. Write the OΔE which results from the application of explicit Euler time marching in matrix-vector form, i.e.,

$$\vec{u}_{n+1} = C\vec{u}_n - \vec{g}_n$$

Write C in banded matrix notation and give the entries of $\vec{g}$. Using the λ-σ relation for the explicit Euler method, find the σ-eigenvalues. Based on these, what is the maximum Courant number allowed for asymptotic stability? Explain why this differs from the answer to problem 8. Hint: is C normal?

Chapter 8

CHOOSING A TIME-MARCHING METHOD

In this chapter we discuss considerations involved in selecting a time-marching method for a specific application. Examples are given showing how time-marching methods can be compared in a given context. An important concept underlying much of this discussion is *stiffness*, which is defined in the next section.

8.1 Stiffness Definition for ODE's

8.1.1 Relation to λ-Eigenvalues

The introduction of the concept referred to as "stiffness" comes about from the numerical analysis of mathematical models constructed to simulate dynamic phenomena containing widely different time scales. Definitions given in the literature are not unique, but fortunately we now have the background material to construct a definition which is entirely sufficient for our purposes.

We start with the assumption that our CFD problem is modeled with sufficient accuracy by a coupled set of ODE's producing an A matrix typified by Eq. 7.1. Any definition of stiffness requires a *coupled* system with at least two eigenvalues, and the decision to use some numerical time-marching or iterative method to solve it. The difference between the dynamic scales in physical space is represented by the difference in the magnitude of the eigenvalues in eigenspace. In the following discussion we concentrate on the transient part of the solution. The forcing function may also be time varying in which case it would also have a time scale. However, we assume that this scale would be adequately resolved by the chosen time-marching method, and, since this part of the ODE has no effect on the numerical stability of

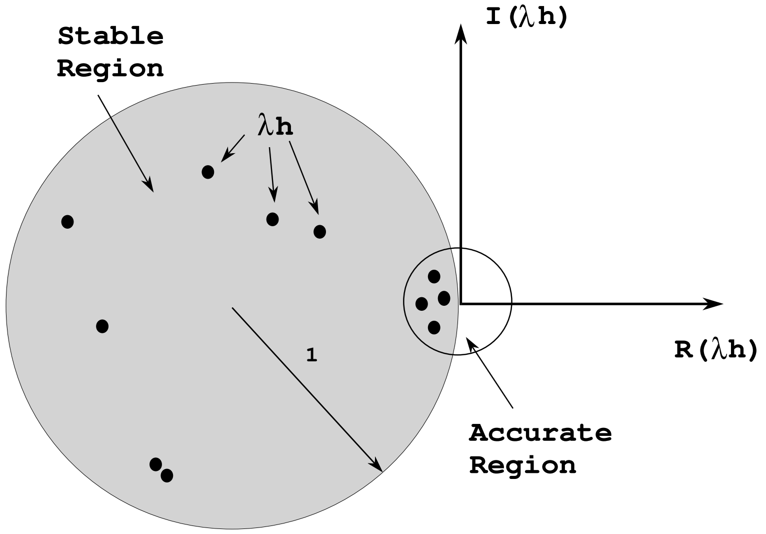

Figure 8.1: Stable and accurate regions for the explicit Euler method.

the homogeneous part, we exclude the forcing function from further discussion in this section.

Consider now the form of the exact solution of a system of ODE's with a complete eigensystem. This is given by Eq. 6.27 and its solution using a one-root, time-marching method is represented by Eq. 6.28. For a given time step, the time integration is an approximation in eigenspace that is different for every eigenvector $\vec{x}_m$. In many numerical applications the eigenvectors associated with the small $|\lambda_m|$ are well resolved and those associated with the large $|\lambda_m|$ are resolved much less accurately, if at all. The situation is represented in the complex λh plane in Fig. 8.1. In this figure the time step has been chosen so that time accuracy is given to the eigenvectors associated with the eigenvalues lying in the small circle and stability without time accuracy is given to those associated with the eigenvalues lying outside of the small circle but still inside the large circle.

The whole concept of stiffness in CFD arises from the fact that we often do not need the *time resolution* of eigenvectors associated with the large $|\lambda_m|$ in the transient solution, although these eigenvectors must remain coupled into the system to maintain a high accuracy of the *spatial resolution.*

8.1.2 Driving and Parasitic Eigenvalues

For the above reason it is convenient to subdivide the transient solution into two parts. First we order the eigenvalues by their magnitudes, thus

$$|\lambda_1| \le |\lambda_2| \le \cdots \le |\lambda_M| \tag{8.1}$$

Then we write

$$\begin{matrix}\text{Transient}\\ \text{Solution}\end{matrix} = \underbrace{\sum_{m=1}^{p} c_m e^{\lambda_m t}\, \vec{x}_m}_{\text{Driving}} + \underbrace{\sum_{m=p+1}^{M} c_m e^{\lambda_m t}\, \vec{x}_m}_{\text{Parasitic}} \tag{8.2}$$

This concept is crucial to our discussion. Rephrased, it states that we can separate our eigenvalue spectrum into two groups; one $[\lambda_1 \to \lambda_p]$ called the driving eigenvalues (our choice of a time-step and marching method must accurately approximate the time variation of the eigenvectors associated with these), and the other, $[\lambda_{p+1} \to \lambda_M]$, called the parasitic eigenvalues (no time accuracy whatsoever is required for the eigenvectors associated with these, but their presence must not contaminate the accuracy of the complete solution). Unfortunately, we find that, although time accuracy requirements are dictated by the driving eigenvalues, numerical stability requirements are dictated by the parasitic ones.

8.1.3 Stiffness Classifications

The following definitions are somewhat useful. An inherently stable set of ODE's is stiff if

$$|\lambda_p| \ll |\lambda_M|$$

In particular we define the ratio

$$C_r = |\lambda_M| \, / \, |\lambda_p|$$

and form the categories

$$\begin{array}{crcl} \text{Mildly-stiff} & & C_r & < 10^2 \\ \text{Strongly-stiff} & 10^3 < & C_r & < 10^5 \\ \text{Extremely-stiff} & 10^6 < & C_r & < 10^8 \\ \text{Pathologically-stiff} & 10^9 < & C_r & \end{array}$$

It should be mentioned that the gaps in the stiff category definitions are intentional because the bounds are arbitrary. It is important to notice that these definitions make no distinction between real, complex, and imaginary eigenvalues.

8.2 Relation of Stiffness to Space Mesh Size

Many flow fields are characterized by a few regions having high spatial gradients of the dependent variables and other domains having relatively low gradient phenomena. As a result it is quite common to cluster mesh points in certain regions of space and spread them out otherwise. Examples of where this clustering might occur are at a shock wave, near an airfoil leading or trailing edge, and in a boundary layer.

One quickly finds that this grid clustering can strongly affect the eigensystem of the resulting A matrix. In order to demonstrate this, let us examine the eigensystems of the model problems given in Section 4.3.2. The simplest example to discuss relates to the model diffusion equation. In this case the eigenvalues are all real, negative numbers that automatically obey the ordering given in Eq. 8.1. Consider the case when *all* of the eigenvalues are parasitic, i.e., we are interested only in the converged steady-state solution. Under these conditions, the stiffness is determined by the ratio λ_M/λ_1. A simple calculation shows that

$$\lambda_1 = -\frac{4\nu}{\Delta x^2}\,\sin^2\left(\frac{\pi}{2(M+1)}\right) \approx -\left(\frac{4\nu}{\Delta x^2}\right)\left(\frac{\Delta x}{2}\right)^2 = -\nu$$

$$\lambda_M \approx -\frac{4\nu}{\Delta x^2}\,\sin^2\left(\frac{\pi}{2}\right) = -\frac{4\nu}{\Delta x^2}$$

and the ratio is

$$\lambda_M/\lambda_1 \approx \frac{4}{\Delta x^2} = 4\left(\frac{M+1}{\pi}\right)^2$$

The most important information found from this example is the fact that the stiffness of the transient solution is directly related to the grid spacing. Furthermore, in diffusion problems this stiffness is proportional to the reciprocal of the space mesh size *squared*. For a mesh size $M = 40$, this ratio is about 680. Even for a mesh of this moderate size the problem is already approaching the category of strongly stiff.

For the biconvection model a similar analysis shows that

$$|\lambda_M|\,/\,|\lambda_1| \approx \frac{1}{\Delta x}$$

Here the stiffness parameter is still space-mesh dependent, but much less so than for diffusion-dominated problems.

We see that in both cases we are faced with the rather annoying fact that the more we try to increase the resolution of our spatial gradients, the stiffer our equations tend to become. Typical CFD problems without chemistry vary between the mildly and strongly stiff categories, and are *greatly* affected by the resolution of a boundary layer since it is a diffusion process. Our brief analysis has been limited to equispaced problems, but in general the stiffness of CFD problems is proportional to the mesh intervals in the manner shown above where *the critical interval is the smallest one in the physical domain.*

8.3 Practical Considerations for Comparing Methods

We have presented relatively simple and reliable measures of stability and both the local and global accuracy of time-marching methods. Since there are an endless number of these methods to choose from, one can wonder how this information is to be used to pick a "best" choice for a particular problem. There is no unique answer to such a question. For example, it is, among other things, highly dependent upon the speed, capacity, and architecture of the available computer, and technology influencing this is undergoing rapid and dramatic changes as this is being written. Nevertheless, if certain ground rules are agreed upon, relevant conclusions can be reached. Let us now examine some ground rules that might be appropriate. It should then be clear how the analysis can be extended to other cases.

Let us consider the problem of measuring the efficiency of a time–marching method for computing, over a fixed interval of time, an accurate transient solution of a coupled set of ODE's. The length of the time interval, T, and the accuracy required of the solution are dictated by the physics of the particular problem involved. For example, in calculating the amount of turbulence in a homogeneous flow, the time interval would be that required to extract a reliable statistical sample, and the accuracy would be related to how much the energy of certain harmonics would be permitted to distort from a given level. Such a computation we refer to as an *event.*

The appropriate error measures to be used in comparing methods for calculating an event are the *global* ones, Er_a, Er_λ and Er_ω, discussed in Section 6.6.5, rather than the local ones er_λ, er_a, and er_p discussed earlier.

The actual form of the coupled ODE's that are produced by the semi-discrete approach is

$$\frac{d\vec{u}}{dt} = \vec{F}(\vec{u}, t)$$

At every time step we must evaluate the function $\vec{F}(\vec{u}, t)$ at least once. This function is usually nonlinear, and its computation *usually consumes the major portion of the computer time required to make the simulation.* We refer to a single calculation of the vector $\vec{F}(\vec{u}, t)$ as a *function evaluation* and denote the total number of such evaluations by F_{ev}.

8.4 Comparing the Efficiency of Explicit Methods

8.4.1 Imposed Constraints

As mentioned above, the efficiency of methods can be compared only if one accepts a set of limiting constraints within which the comparisons are carried out. The following assumptions bound the considerations made in this Section:

1. The time-march method is explicit.

2. Implications of computer storage capacity and access time are ignored. In some contexts, this can be an important consideration.

3. The calculation is to be time-accurate, must simulate an entire event which takes a total time T, and must use a constant time step size, h, so that

$$T = Nh$$

where N is the total number of time steps.

8.4.2 An Example Involving Diffusion

Let the event be the numerical solution of

$$\frac{du}{dt} = -u \tag{8.3}$$

from $t = 0$ to $T = -\ln(0.25)$ with $u(0) = 1$. Eq. 8.3 is obtained from our representative ODE with $\lambda = -1$, $a = 0$. Since the exact solution is $u(t) = u(0)e^{-t}$, this makes the exact value of u at the end of the event equal to 0.25, i.e., $u(T) = 0.25$. To the constraints imposed above, let us set the additional requirement

- The error in u at the end of the event, i.e., the *global* error, must be $< 0.5\%$.

We judge the most efficient method as the one that satisfies these conditions and has the smallest number of function evaluations, F_{ev}. Three methods are compared — explicit Euler, AB2, and RK4.

First of all, the allowable error constraint means that the global error in the amplitude, see Eq. 6.48, must have the property:

$$\left| \frac{Er_\lambda}{e^{\lambda T}} \right| < 0.005$$

Then, since $h = T/N = -\ln(0.25)/N$, it follows that

$$\left|1 - (\sigma_1(\ln(.25)/N))^N/.25\right| < 0.005$$

where σ_1 is found from the characteristic polynomials given in Table 7.1. The results shown in Table 8.1 were computed using a simple iterative procedure.

Method	N	h	σ_1	F_{ev}	Er_λ	
Euler	193	.00718	.99282	193	.001248	worst
AB2	16	.0866	.9172	16	.001137	
RK4	2	.6931	.5012	8	.001195	best

Table 8.1: Comparison of time-marching methods for a simple diffusion problem.

In this example we see that, for a given global accuracy, the method with the highest local accuracy is the most efficient on the basis of the expense in terms of function evaluations. Thus the second-order Adams-Bashforth method is much better than the first-order Euler method, and the fourth-order Runge-Kutta method is the best of all. The main purpose of this exercise is to show the (usually) great superiority of second-order over first-order time-marching methods.

8.4.3 An Example Involving Periodic Convection

Let us use as a basis for this example the study of homogeneous turbulence simulated by the numerical solution of the incompressible Navier-Stokes equations inside a cube with periodic boundary conditions on all sides. In this numerical experiment the function evaluations contribute overwhelmingly to the CPU time, and the number of these evaluations must be kept to an absolute minimum because of the magnitude of the problem. On the other hand, a complete event must be established in order to obtain meaningful statistical samples which are the essence of the solution. In this case, in addition to the constraints given in Section 8.4.1, we add the following:

- The number of evaluations of $\vec{F}(\vec{u}, t)$ is fixed.

Under these conditions a method is judged as best when it has the highest global accuracy for resolving eigenvectors with imaginary eigenvalues. The above constraint has led to the invention of schemes that omit the function evaluation in the corrector step of a predictor-corrector combination, leading to the so-called incomplete predictor-corrector methods. The presumption is, of course, that more efficient methods will result from the omission of the second function evaluation. An example is the method of Gazdag, given in Section 6.8. Basically this is composed of an AB2 predictor and a trapezoidal corrector. However, the derivative of the fundamental family is never found so there is only one evaluation required to complete each cycle. The λ-σ relation for the method is shown as entry 10 in Table 7.1.

In order to discuss our comparisions we introduce the following definitions:

- Let a k-evaluation method be defined as one that requires k evaluations of $\vec{F}(\vec{u},t)$ to advance one step using that method's time interval, h.
- Let K represent the total number of allowable function evaluations.
- Let h_1 be the time interval advanced in one step of a one-evaluation method.

The Gazdag, leapfrog, and AB2 schemes are all 1-evaluation methods. The second and fourth order RK methods are 2- and 4-evaluation methods, respectively. For a 1-evaluation method the total number of time steps, N, and the number of evaluations, K, are the same, one evaluation being used for each step, so that for these methods $h = h_1$. For a 2-evaluation method $N = K/2$ since two evaluations are used for each step. However, in this case, in order to arrive at the same time T after K evaluations, the time step must be twice that of a one–evaluation method so $h = 2h_1$. For a 4-evaluation method the time interval must be $h = 4h_1$, etc. Notice that as k increases, the time span required for one application of the method increases. *However, notice also that as k increases, the power to which σ_1 is raised to arrive at the final destination decreases;* see the Figure below. This is the key to the true comparison of time-march methods for this type of problem.

$$
\begin{array}{cccccccccccc}
 & 0 & & & & & & & & & T & u_N \\
k=1 & | & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & & | & [\sigma(\lambda h_1)]^8 \\
k=2 & | & 2h_1 & \bullet & & \bullet & & \bullet & & & | & [\sigma(2\lambda h_1)]^4 \\
k=4 & | & & 4h_1 & & \bullet & & & & & | & [\sigma(4\lambda h_1)]^2
\end{array}
$$

Step sizes and powers of σ for k-evaluation methods used to get to the same value of T if 8 evaluations are allowed.

In general, after K evaluations, the global amplitude and phase error for k-evaluation methods applied to systems with pure imaginary λ-roots can be written[1]

$$Er_a = 1 - |\sigma_1(ik\omega h_1)|^{K/k} \tag{8.4}$$

[1]See Eqs. 6.49 and 6.50.

$$Er_\omega = \omega T - \frac{K}{k}\tan^{-1}\left[\frac{[\sigma_1(ik\omega h_1)]_{imaginary}}{[\sigma_1(ik\omega h_1)]_{real}}\right] \tag{8.5}$$

Consider a convection-dominated event for which the function evaluation is very time consuming. We idealize to the case where $\lambda = i\omega$ and set ω equal to one. The event must proceed to the time t = T = 10. We consider two maximum evaluation limits $K = 50$ and $K = 100$ and choose from four possible methods, leapfrog, AB2, Gazdag, and RK4. The first three of these are one-evaluation methods and the last one is a four-evaluation method. It is not difficult to show that on the basis of local error (made in a single step) the Gazdag method is superior to the RK4 method in both amplitude and phase. For example, for $\omega h = 0.2$ the Gazdag method produces a $|\sigma_1| = 0.9992276$ whereas for $\omega h = 0.8$ (which must be used to keep the number of evaluations the same) the RK4 method produces a $|\sigma_1| = 0.998324$. However, we are making our comparisons on the basis of global error for a fixed number of evaluations.

First of all we see that for a one-evaluation method $h_1 = T/K$. Using this, and the fact that $\omega = 1$, we find, by some rather simple calculations[2] made using Eqs. 8.4 and 8.5, the results shown in Table 8.2. Notice that to find global error the Gazdag root must be raised to the power of 50 while the RK4 root is raised only to the power of 50/4. On the basis of *global error* the Gazdag method is not superior to RK4 in either amplitude or phase, although, in terms of phase error (for which it was designed) it is superior to the other two methods shown.

	K	leapfrog	AB2	Gazdag	RK4
$\omega h_1 = .1$	100	1.0	1.003	.995	.999
$\omega h_1 = .2$	50	1.0	1.022	.962	.979

a. Amplitude, exact = 1.0.

	K	leapfrog	AB2	Gazdag	RK4
$\omega h_1 = .1$	100	$-.96$	-2.4	.45	.12
$\omega h_1 = .2$	50	-3.8	-9.8	1.5	1.5

b. Phase error in degrees.

Table 8.2: Comparison of global amplitude and phase errors for four methods.

[2]The σ_1 root for the Gazdag method can be found using a numerical root finding routine to trace the three roots in the σ-plane, see Fig. 7.3e.

Using analysis such as this (and also considering the stability boundaries) the RK4 method is recommended as a basic first choice for any explicit time-accurate calculation of a convection-dominated problem.

8.5 Coping With Stiffness

8.5.1 Explicit Methods

The ability of a numerical method to cope with stiffness can be illustrated quite nicely in the complex λh plane. A good example of the concept is produced by studying the Euler method applied to the representative equation. The transient solution is $u_n = (1+\lambda h)^n$ and the trace of the complex value of λh which makes $|1+\lambda h| = 1$ gives the whole story. In this case the trace forms a circle of unit radius centered at $(-1,0)$ as shown in Fig. 8.1. If h is chosen so that all λh in the ODE eigensystem fall inside this circle the integration will be numerically stable. Also shown by the small circle centered at the origin is the region of Taylor series accuracy. If some λh fall outside the small circle but stay within the stable region, these λh are stiff, but stable. We have defined these λh as parasitic eigenvalues. Stability boundaries for some explicit methods are shown in Figs. 7.5 and 7.6.

For a specific example, consider the mildly stiff system composed of a coupled two-equation set having the two eigenvalues $\lambda_1 = -100$ and $\lambda_2 = -1$. If uncoupled and evaluated in wave space, the time histories of the two solutions would appear as a rapidly decaying function in one case, and a relatively slowly decaying function in the other. Analytical evaluation of the time histories poses no problem since e^{-100t} quickly becomes very small and can be neglected in the expressions when time becomes large. Numerical evaluation is altogether different. Numerical solutions, of course, depend upon $[\sigma(\lambda_m h)]^n$ and no $|\sigma_m|$ can exceed one for any λ_m in the coupled system or else the process is numerically unstable.

Let us choose the simple explicit Euler method for the time march. The coupled equations in real space are represented by

$$\begin{aligned} u_1(n) &= c_1(1-100h)^n x_{11} + c_2(1-h)^n x_{12} + (PS)_1 \\ u_2(n) &= c_1(1-100h)^n x_{21} + c_2(1-h)^n x_{22} + (PS)_2 \end{aligned} \tag{8.6}$$

We will assume that our accuracy requirements are such that sufficient accuracy is obtained as long as $|\lambda h| \le 0.1$. This defines a time step limit based on *accuracy* considerations of $h = 0.001$ for λ_1 and $h = 0.1$ for λ_2. The time step limit based on *stability*, which is determined from λ_1, is $h = 0.02$. We will also assume that $c_1 = c_2 = 1$ and that an amplitude less than 0.001 is negligible. We first run 66 time steps with $h = 0.001$ in order to resolve the λ_1 term. With this time step the

λ_2 term is resolved exceedingly well. After 66 steps, the amplitude of the λ_1 term (i.e., $(1-100h)^n$) is less than 0.001 and that of the λ_2 term (i.e., $(1-h)^n$) is 0.9361. Hence the λ_1 term can now be considered negligible. To drive the $(1-h)^n$ term to zero (i.e., below 0.001), we would like to change the step size to $h = 0.1$ and continue. We would then have a well resolved answer to the problem throughout the entire relevant time interval. However, this is not possible because of the coupled presence of $(1-100h)^n$, which in just 10 steps at $h = 0.1$ amplifies those terms by $\approx 10^9$, far outweighing the initial decrease obtained with the smaller time step. In fact, with $h = 0.02$, the maximum step size that can be taken in order to maintain stability, about 339 time steps have to be computed in order to drive e^{-t} to below 0.001. Thus the total simulation requires 405 time steps.

8.5.2 Implicit Methods

Now let us re-examine the problem that produced Eq. 8.6 but this time using an unconditionally stable implicit method for the time march. We choose the trapezoidal method. Its behavior in the λh plane is shown in Fig. 7.4b. Since this is also a one-root method, we simply replace the Euler σ with the trapezoidal one and analyze the result. It follows that the final numerical solution to the ODE is now represented in real space by

$$\begin{aligned}
u_1(n) &= c_1\left(\frac{1-50h}{1+50h}\right)^n x_{11} + c_2\left(\frac{1-0.5h}{1+0.5h}\right)^n x_{12} + (PS)_1 \\
u_2(n) &= c_1\left(\frac{1-50h}{1+50h}\right)^n x_{21} + c_2\left(\frac{1-0.5h}{1+0.5h}\right)^n x_{22} + (PS)_2
\end{aligned} \tag{8.7}$$

In order to resolve the initial transient of the term e^{-100t}, we need to use a step size of about $h = 0.001$. This is the same step size used in applying the explicit Euler method because here accuracy is the only consideration and a very small step size must be chosen to get the desired resolution. (It is true that for the same accuracy we could in this case use a larger step size because this is a second-order method, but that is not the point of this exercise). After 70 time steps the λ_1 term has amplitude less than 0.001 and can be neglected. Now with the implicit method we can proceed to calculate the remaining part of the event using our desired step size $h = 0.1$ without any problem of instability, with 69 steps required to reduce the amplitude of the second term to below 0.001. In both intervals the desired solution is second-order accurate and well resolved. It is true that in the final 69 steps one σ-root is $[1-50(0.1)]/[1+50(0.1)] = 0.666\cdots$, and this has no physical meaning whatsoever. However, its influence on the coupled solution is negligible at the end of the first 70 steps, and, since $(0.666\cdots)^n < 1$, its influence in the remaining 70 steps is even less. Actually, although this root is one of the principal roots in the system, its behavior for $t > 0.07$ is identical to that of a stable spurious root. The total simulation requires 139 time steps.

8.5.3 A Perspective

It is important to retain a proper perspective on a problem represented by the above example. It is clear that an unconditionally stable method can always be called upon to solve stiff problems with a minimum number of time steps. In the example, the conditionally stable Euler method required 405 time steps, as compared to about 139 for the trapezoidal method, about three times as many. However, the Euler method is extremely easy to program and requires very little arithmetic per step. For preliminary investigations it is often the best method to use for mildly-stiff diffusion dominated problems. For refined investigations of such problems an explicit method of second order or higher, such as Adams-Bashforth or Runge-Kutta methods, is recommended. These explicit methods can be considered as effective mildly stiff-stable methods. However, it should be clear that as the degree of stiffness of the problem increases, the advantage begins to tilt towards implicit methods, as the reduced number of time steps begins to outweigh the increased cost per time step. The reader can repeat the above example with $\lambda_1 = -10,000$, $\lambda_2 = -1$, which is in the strongly-stiff category.

There is yet another technique for coping with certain stiff systems in fluid dynamic applications. This is known as the *multigrid* method. It has enjoyed remarkable success in many practical problems; however, we need an introduction to the theory of relaxation before it can be presented.

8.6 Steady Problems

In Chapter 6 we wrote the OΔE solution in terms of the principal and spurious roots as follows:

$$
\begin{aligned}
u_n = \; & c_{11}(\sigma_1)_1^n \, \vec{x}_1 + \cdots + c_{m1}(\sigma_m)_1^n \, \vec{x}_m + \cdots + c_{M1}(\sigma_M)_1^n \, \vec{x}_M + P.S. \\
& + c_{12}(\sigma_1)_2^n \, \vec{x}_1 + \cdots + c_{m2}(\sigma_m)_2^n \, \vec{x}_m + \cdots + c_{M2}(\sigma_M)_2^n \, \vec{x}_M \\
& + c_{13}(\sigma_1)_3^n \, \vec{x}_1 + \cdots + c_{m3}(\sigma_m)_3^n \, \vec{x}_m + \cdots + c_{M3}(\sigma_M)_3^n \, \vec{x}_M \\
& + \text{etc., if there are more spurious roots}
\end{aligned}
\tag{8.8}
$$

When solving a *steady* problem, we have no interest whatsoever in the transient portion of the solution. Our sole goal is to eliminate it as quickly as possible. Therefore, the choice of a time-marching method for a steady problem is similar to that for a stiff problem, the difference being that the order of accuracy is irrelevant. Hence the explicit Euler method is a candidate for steady diffusion dominated problems, and the fourth-order Runge-Kutta method is a candidate for steady convection dominated problems, because of their stability properties. Among implicit methods, the implicit Euler method is the obvious choice for steady problems.

When we seek only the steady solution, all of the eigenvalues can be considered to be parasitic. Referring to Fig. 8.1, none of the eigenvalues are required to fall in the accurate region of the time-marching method. Therefore the time step can be chosen to eliminate the transient as quickly as possible with no regard for time accuracy. For example, when using the implicit Euler method with local time linearization, Eq. 6.96, one would like to take the limit $h \to \infty$, which leads to Newton's method, Eq. 6.98. However, a finite time step may be required until the solution is somewhat close to the steady solution.

8.7 Problems

1. Repeat the time-march comparisons for diffusion (Section 8.4.2) and periodic convection (Section 8.4.3) using 2nd- and 3rd-order Runge-Kutta methods.

2. Repeat the time-march comparisons for diffusion (Section 8.4.2) and periodic convection (Section 8.4.3) using the 3rd- and 4th-order Adams-Bashforth methods. Considering the stability bounds for these methods (see problem 4 in Chapter 7) as well as their memory requirements, compare and contrast them with the 3rd- and 4th-order Runge-Kutta methods.

3. Consider the diffusion equation (with $\nu = 1$) discretized using 2nd-order central differences on a grid with 10 (interior) points. Find and plot the eigenvalues and the corresponding modified wavenumbers. If we use the explicit Euler time-marching method what is the maximum allowable time step if all but the first two eigenvectors are considered parasitic? Assume that sufficient accuracy is obtained as long as $|\lambda h| \leq 0.1$. What is the maximum allowable time step if all but the first eigenvector are considered parasitic?

Chapter 9

RELAXATION METHODS

In the past three chapters, we developed a methodology for designing, analyzing, and choosing time-marching methods. These methods can be used to compute the time-accurate solution to linear and nonlinear systems of ODE's in the general form

$$\frac{d\vec{u}}{dt} = \vec{F}(\vec{u}, t) \tag{9.1}$$

which arise after spatial discretization of a PDE. Alternatively, they can be used to solve for the steady solution of Eq. 9.1, which satisfies the following coupled system of nonlinear algebraic equations:

$$\vec{F}(\vec{u}) = 0 \tag{9.2}$$

In the latter case, the unsteady equations are integrated until the solution converges to a steady solution. The same approach permits a time-marching method to be used to solve a linear system of algebraic equations in the form

$$A\vec{x} = \vec{b} \tag{9.3}$$

To solve this system using a time-marching method, a time derivative is introduced as follows

$$\frac{d\vec{x}}{dt} = A\vec{x} - \vec{b} \tag{9.4}$$

and the system is integrated in time until the transient has decayed to a sufficiently low level. Following a time-dependent path to steady state is possible only if all of the eigenvalues of the matrix A (or $-A$) have real parts lying in the left half-plane. Although the solution $\vec{x} = A^{-1}\vec{b}$ exists as long as A is nonsingular, the ODE given by Eq. 9.4 has a stable steady solution only if A meets the above condition.

The common feature of all time-marching methods is that they are at least first-order accurate. In this chapter, we consider iterative methods which are not time accurate at all. Such methods are known as *relaxation methods*. While they are applicable to coupled systems of nonlinear algebraic equations in the form of Eq. 9.2, our analysis will focus on their application to large sparse linear systems of equations in the form

$$A_b\vec{u} - \vec{f}_b = 0 \tag{9.5}$$

where A_b is nonsingular, and the use of the subscript b will become clear shortly. Such systems of equations arise, for example, at each time step of an implicit time-marching method or at each iteration of Newton's method. Using an iterative method, we seek to obtain rapidly a solution which is arbitrarily close to the exact solution of Eq. 9.5, which is given by

$$\vec{u}_\infty = A_b^{-1}\vec{f}_b \tag{9.6}$$

9.1 Formulation of the Model Problem

9.1.1 Preconditioning the Basic Matrix

It is standard practice in applying relaxation procedures to *precondition* the basic equation. This preconditioning has the effect of multiplying Eq. 9.5 from the left by some nonsingular matrix. In the simplest possible case the conditioning matrix is a diagonal matrix composed of a constant $D(b)$. If we designate the conditioning matrix by C, the problem becomes one of solving for $\vec{u}$ in

$$CA_b\vec{u} - C\vec{f}_b = 0 \tag{9.7}$$

Notice that the solution of Eq. 9.7 is

$$\vec{u} = [CA_b]^{-1}C\vec{f}_b = A_b^{-1}C^{-1}C\vec{f}_b = A_b^{-1}\vec{f}_b \tag{9.8}$$

which is identical to the solution of Eq. 9.5, provided C^{-1} exists.

In the following we will see that our approach to the iterative solution of Eq. 9.7 depends crucially on the eigenvalue and eigenvector structure of the matrix CA_b, and, equally important, does not depend at all on the eigensystem of the basic matrix A_b. For example, there are well-known techniques for accelerating relaxation schemes if the eigenvalues of CA_b are all real and of the same sign. To use these schemes, the conditioning matrix C must be chosen such that this requirement is satisfied. A choice of C which ensures this condition is the negative transpose of A_b.

For example, consider a spatial discretization of the linear convection equation using centered differences with a Dirichlet condition on the left side and no constraint on the right side. Using a first-order backward difference on the right side (as in Section 3.6), this leads to the approximation

$$\delta_x \vec{u} = \frac{1}{2\Delta x}\left\{\begin{bmatrix} 0 & 1 & & & \\ -1 & 0 & 1 & & \\ & -1 & 0 & 1 & \\ & & -1 & 0 & 1 \\ & & & -2 & 2 \end{bmatrix}\vec{u} + \begin{bmatrix} -u_a \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}\right\} \tag{9.9}$$

The matrix in Eq. 9.9 has eigenvalues whose imaginary parts are much larger than their real parts. It can first be conditioned so that the modulus of each element is 1. This is accomplished using a diagonal preconditioning matrix

$$D = 2\Delta x\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} \end{bmatrix} \tag{9.10}$$

which scales each row. We then further condition with multiplication by the negative transpose. The result is

$$A_2 = -A_1^T A_1 =$$

$$\begin{bmatrix} 0 & 1 & & & \\ -1 & 0 & 1 & & \\ & -1 & 0 & 1 & \\ & & -1 & 0 & 1 \\ & & & -1 & -1 \end{bmatrix}\begin{bmatrix} 0 & 1 & & & \\ -1 & 0 & 1 & & \\ & -1 & 0 & 1 & \\ & & -1 & 0 & 1 \\ & & & -1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -1 & 0 & 1 & & \\ 0 & -2 & 0 & 1 & \\ 1 & 0 & -2 & 0 & 1 \\ & 1 & 0 & -2 & 1 \\ & & 1 & 1 & -2 \end{bmatrix} \tag{9.11}$$

If we define a permutation matrix P [1] and carry out the process $P^T[-A_1^TA_1]P$ (which just reorders the elements of A_1 and doesn't change the eigenvalues) we find

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 & & \\ 0 & -2 & 0 & 1 & \\ 1 & 0 & -2 & 0 & 1 \\ & 1 & 0 & -2 & 1 \\ & & 1 & 1 & -2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & 1 \\ & & & 1 & -1 \end{bmatrix} \tag{9.12}$$

which has all negative real eigenvalues, as given in Appendix B. Thus even when the basic matrix A_b has nearly imaginary eigenvalues, the conditioned matrix $-A_b^TA_b$ is nevertheless symmetric negative definite (i.e., symmetric with negative real eigenvalues), and the classical relaxation methods can be applied. We do not necessarily recommend the use of $-A_b^T$ as a preconditioner; we simply wish to show that a broad range of matrices can be preconditioned into a form suitable for our analysis.

9.1.2 The Model Equations

Preconditioning processes such as those described in the last section allow us to prepare our algebraic equations in advance so that certain eigenstructures are guaranteed. In the remainder of this chapter, we will thoroughly investigate some simple equations which model these structures. We will consider the preconditioned system of equations having the form

$$A\vec{\phi} - \vec{f} = 0 \tag{9.13}$$

where A is symmetric negative definite.[2] The symbol for the dependent variable has been changed to ϕ as a reminder that the physics being modeled is no longer time

[1] A permutation matrix (defined as a matrix with exactly one 1 in each row and column and has the property that $P^T = P^{-1}$) just rearranges the rows and columns of a matrix.

[2] We use a symmetric negative definite matrix to simplify certain aspects of our analysis. Relaxation methods are applicable to more general matrices. The classical methods will usually converge if A_b is *diagonally dominant*, as defined in Appendix A.

accurate when we later deal with ODE formulations. Note that the solution of Eq. 9.13, $\vec{\phi} = A^{-1}\vec{f}$, is guaranteed to exist because A is nonsingular. In the notation of Eqs. 9.5 and 9.7,

$$A = CA_b \qquad \text{and} \qquad \vec{f} = C\vec{f}_b \tag{9.14}$$

The above was written to treat the general case. It is instructive in formulating the concepts to consider the special case given by the diffusion equation in one dimension with unit diffusion coefficient ν:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - g(x) \tag{9.15}$$

The steady-state solution satisfies

$$\frac{\partial^2 u}{\partial x^2} = g(x) \tag{9.16}$$

Introducing the three-point central differencing scheme for the second derivative with Dirichlet boundary conditions, we find

$$\frac{d\vec{u}}{dt} = \frac{1}{\Delta x^2} B(1, -2, 1)\vec{u} + (\vec{bc}) - \vec{g} \tag{9.17}$$

where $(\vec{bc})$ contains the boundary conditions and $\vec{g}$ contains the values of the source term at the grid nodes. In this case

$$\begin{aligned} A_b &= \frac{1}{\Delta x^2} B(1, -2, 1) \\ \vec{f}_b &= \vec{g} - (\vec{bc}) \end{aligned} \tag{9.18}$$

Choosing $C = \Delta x^2 I$, we obtain

$$B(1, -2, 1)\vec{\phi} = \vec{f} \tag{9.19}$$

where $\vec{f} = \Delta x^2 \vec{f}_b$. If we consider a Dirichlet boundary condition on the left side and either a Dirichlet or a Neumann condition on the right side, then A has the form

$$A = B\Big(1, \vec{b}, 1\Big)$$

$$\vec{b} = [-2, -2, \cdots, -2, s]^T$$

$$s = -2 \qquad \text{or} \qquad -1 \tag{9.20}$$

Note that $s = -1$ is easily obtained from the matrix resulting from the Neumann boundary condition given in Eq. 3.24 using a diagonal conditioning matrix. A tremendous amount of insight to the basic features of relaxation is gained by an appropriate study of the one-dimensional case, and much of the remaining material is devoted to this case. We attempt to do this in such a way, however, that it is directly applicable to two- and three-dimensional problems.

9.2 Classical Relaxation

9.2.1 The Delta Form of an Iterative Scheme

We will consider relaxation methods which can be expressed in the following delta form:

$$H\left[\vec{\phi}_{n+1} - \vec{\phi}_n\right] = A\vec{\phi}_n - \vec{f} \tag{9.21}$$

where H is some nonsingular matrix which depends upon the iterative method. The matrix H is independent of n for *stationary* methods and is a function of n for *nonstationary* ones. The iteration count is designated by the subscript n or the superscript (n). The converged solution is designated $\vec{\phi}_\infty$ so that

$$\vec{\phi}_\infty = A^{-1}\vec{f} \tag{9.22}$$

9.2.2 The Converged Solution, the Residual, and the Error

Solving Eq. 9.21 for $\vec{\phi}_{n+1}$ gives

$$\vec{\phi}_{n+1} = [I + H^{-1}A]\vec{\phi}_n - H^{-1}\vec{f} = G\vec{\phi}_n - H^{-1}\vec{f} \tag{9.23}$$

where

$$G \equiv I + H^{-1}A \tag{9.24}$$

Hence it is clear that H should lead to a system of equations which is easy to solve, or at least easier to solve than the original system. The *error* at the nth iteration is defined as

$$\vec{e}_n \equiv \vec{\phi}_n - \vec{\phi}_\infty = \vec{\phi}_n - A^{-1}\vec{f} \tag{9.25}$$

where $\vec{\phi}_\infty$ was defined in Eq. 9.22. The *residual* at the nth iteration is defined as

$$\vec{r}_n \equiv A\vec{\phi}_n - \vec{f} \tag{9.26}$$

Multiply Eq. 9.25 by A from the left, and use the definition in Eq. 9.26. There results the relation between the error and the residual

$$A\vec{e}_n - \vec{r}_n = 0 \tag{9.27}$$

Finally, it is not difficult to show that

$$\vec{e}_{n+1} = G\vec{e}_n \tag{9.28}$$

Consequently, G is referred to as the basic iteration matrix, and its eigenvalues, which we designate as σ_m, determine the convergence rate of a method.

In all of the above, we have considered only what are usually referred to as stationary processes in which H is constant throughout the iterations. Nonstationary processes in which H (and possibly C) is varied at each iteration are discussed in Section 9.5.

9.2.3 The Classical Methods

Point Operator Schemes in One Dimension

Let us consider three classical relaxation procedures for our model equation

$$B(1,-2,1)\vec{\phi} = \vec{f} \tag{9.29}$$

as given in Section 9.1.2. The Point-Jacobi method is expressed in point operator form for the one-dimensional case as

$$\phi_j^{(n+1)} = \frac{1}{2}\left[\phi_{j-1}^{(n)} + \phi_{j+1}^{(n)} - f_j\right] \tag{9.30}$$

This operator comes about by choosing the value of $\phi_j^{(n+1)}$ such that together with the *old* values of ϕ_{j-1} and ϕ_{j+1}, the jth row of Eq. 9.29 is satisfied. The Gauss-Seidel method is

$$\phi_j^{(n+1)} = \frac{1}{2}\left[\phi_{j-1}^{(n+1)} + \phi_{j+1}^{(n)} - f_j\right] \tag{9.31}$$

This operator is a simple extension of the point-Jacobi method which uses the most recent update of ϕ_{j-1}. Hence the jth row of Eq. 9.29 is satisfied using the *new* values of ϕ_j and ϕ_{j-1} and the *old* value of ϕ_{j+1}. The method of successive overrelaxation (SOR) is based on the idea that if the correction produced by the Gauss-Seidel method tends to move the solution toward $\vec{\phi}_\infty$, then perhaps it would be better to move further in this direction. It is usually expressed in two steps as

$$\begin{aligned} \tilde{\phi}_j &= \frac{1}{2}\Big[\phi_{j-1}^{(n+1)} + \phi_{j+1}^{(n)} - f_j\Big] \\ \phi_j^{(n+1)} &= \phi_j^{(n)} + \omega\Big[\tilde{\phi}_j - \phi_j^{(n)}\Big] \end{aligned} \tag{9.32}$$

where ω generally lies between 1 and 2, but it can also be written in the single line

$$\phi_j^{(n+1)} = \frac{\omega}{2}\phi_{j-1}^{(n+1)} + (1-\omega)\phi_j^{(n)} + \frac{\omega}{2}\phi_{j+1}^{(n)} - \frac{\omega}{2}f_j \tag{9.33}$$

The General Form

The general form of the classical methods is obtained by splitting the matrix A in Eq. 9.13 into its diagonal, D, the portion of the matrix below the diagonal, L, and the portion above the diagonal, U, such that

$$A = L + D + U \tag{9.34}$$

Then the point-Jacobi method is obtained with $H = -D$, which certainly meets the criterion that it is easy to solve. The Gauss-Seidel method is obtained with $H = -(L + D)$, which is also easy to solve, being lower triangular.

9.3 The ODE Approach to Classical Relaxation

9.3.1 The Ordinary Differential Equation Formulation

The particular type of delta form given by Eq. 9.21 leads to an interpretation of relaxation methods in terms of solution techniques for coupled first-order ODE's, about which we have already learned a great deal. One can easily see that Eq. 9.21 results from the application of the explicit Euler time-marching method (with $h = 1$) to the following system of ODE's:

$$H\frac{d\vec{\phi}}{dt} = A\vec{\phi} - \vec{f} \tag{9.35}$$

This is equivalent to

$$\frac{d\vec{\phi}}{dt} = H^{-1}C\Big[A_b\vec{\phi} - \vec{f}_b\Big] = H^{-1}[A\vec{\phi} - \vec{f}] \tag{9.36}$$

In the special case where $H^{-1}A$ depends on neither $\vec{u}$ nor t, $H^{-1}\vec{f}$ is also independent of t, *and* the eigenvectors of $H^{-1}A$ are linearly independent, the solution can be written as

$$\vec{\phi} = \underbrace{c_1 e^{\lambda_1 t}\vec{x}_1 + \cdots + c_M e^{\lambda_M t}\vec{x}_M}_{\text{error}} + \vec{\phi}_\infty \tag{9.37}$$

where what is referred to in time-accurate analysis as the transient solution, is now referred to in relaxation analysis as the error. It is clear that, if all of the eigenvalues of $H^{-1}A$ have negative real parts (which implies that $H^{-1}A$ is nonsingular), then the system of ODE's has a *steady-state* solution which is approached as $t \to \infty$, given by

$$\vec{\phi}_\infty = A^{-1}\vec{f} \tag{9.38}$$

which is the solution of Eq. 9.13. We see that the goal of a relaxation method is to remove the transient solution from the general solution in the most efficient way possible. The λ eigenvalues are fixed by the basic matrix in Eq. 9.36, the preconditioning matrix in 9.7, and the secondary conditioning matrix in 9.35. The σ eigenvalues are fixed for a given λh by the choice of time-marching method. Throughout the remaining discussion we will refer to the independent variable t as "time", even though no true time accuracy is involved.

In a stationary method, H and C in Eq. 9.36 are independent of t, that is, they are not changed throughout the iteration process. The generalization of this in our approach is to make h, the "time" step, a constant for the entire iteration.

Suppose the explicit Euler method is used for the time integration. For this method $\sigma_m = 1 + \lambda_m h$. Hence the numerical solution after n steps of a stationary relaxation method can be expressed as (see Eq. 6.28)

$$\vec{\phi}_n = \underbrace{c_1\vec{x}_1(1+\lambda_1 h)^n + \cdots + c_m\vec{x}_m(1+\lambda_m h)^n + \cdots + c_M\vec{x}_M(1+\lambda_M h)^n}_{\text{error}} + \vec{\phi}_\infty \tag{9.39}$$

The initial amplitudes of the eigenvectors are given by the magnitudes of the c_m. These are fixed by the initial guess. In general it is assumed that any or all of the eigenvectors could have been given an equally "bad" excitation by the initial guess, so that we must devise a way to remove them all from the general solution on an equal basis. Assuming that $H^{-1}A$ has been chosen (that is, an iteration process has been decided upon), the only free choice remaining to accelerate the removal of the error terms is the choice of h. As we shall see, the three classical methods have all been conditioned by the choice of H to have an optimum h equal to 1 for a stationary iteration process.

9.3.2 ODE Form of the Classical Methods

The three iterative procedures defined by Eqs. 9.30, 9.31 and 9.32 obey no apparent pattern except that they are easy to implement in a computer code since all of the data required to update the value of one point are explicitly available at the time of the update. Now let us study these methods as subsets of ODE as formulated in Section 9.3.1. Insert the model equation 9.29 into the ODE form 9.35. Then

$$H\frac{d\vec{\phi}}{dt} = B(1,-2,1)\vec{\phi} - \vec{f} \tag{9.40}$$

As a start, let us use for the numerical integration the explicit Euler method

$$\phi_{n+1} = \phi_n + h\phi'_n \tag{9.41}$$

with a step size, h, equal to 1. We arrive at

$$H(\vec{\phi}_{n+1} - \vec{\phi}_n) = B(1,-2,1)\vec{\phi}_n - \vec{f} \tag{9.42}$$

It is clear that the best choice of H from the point of view of matrix algebra is $-B(1,-2,1)$ since then multiplication from the left by $-B^{-1}(1,-2,1)$ gives the correct answer in one step. However, this is not in the spirit of our study, since multiplication by the inverse amounts to solving the problem by a direct method without iteration. The constraint on H that is in keeping with the formulation of the three methods described in Section 9.2.3 is that all the elements above the diagonal (or below the diagonal if the sweeps are from right to left) are zero. If we impose this constraint and further restrict ourselves to banded tridiagonals with a single constant in each band, we are led to

$$B(-\beta, \frac{2}{\omega}, 0)(\vec{\phi}_{n+1} - \vec{\phi}_n) = B(1,-2,1)\vec{\phi}_n - \vec{f} \tag{9.43}$$

where β and ω are arbitrary. With this choice of notation the three methods presented in Section 9.2.3 can be identified using the entries in Table 9.1.

TABLE 9.1: VALUES OF β and ω IN EQ. 9.43 THAT LEAD TO CLASSICAL RELAXATION METHODS

β	ω	Method	Equation
0	1	Point-Jacobi	9.30
1	1	Gauss-Seidel	9.31
1	$2/\left[1+\sin\left(\frac{\pi}{M+1}\right)\right]$	Optimum SOR	9.32

The fact that the values in the tables lead to the methods indicated can be verified by simple algebraic manipulation. However, our purpose is to examine the whole procedure as a special subset of the theory of ordinary differential equations. In this light, the three methods are all contained in the following set of ODE's

$$\frac{d\vec{\phi}}{dt} = B^{-1}(-\beta, \frac{2}{\omega}, 0)\left[B(1,-2,1)\vec{\phi} - \vec{f}\right] \tag{9.44}$$

and appear from it in the special case when the explicit Euler method is used for its numerical integration. The point operator that results from the use of the explicit Euler scheme is

$$\phi_j^{(n+1)} = \left(\frac{\omega\beta}{2}\phi_{j-1}^{(n+1)} + \frac{\omega}{2}(h-\beta)\phi_{j-1}^{(n)}\right) - \left((\omega h - 1)\phi_j^{(n)}\right) + \left(\frac{\omega h}{2}\phi_{j+1}^{(n)}\right) - \frac{\omega h}{2}f_j \tag{9.45}$$

This represents a generalization of the classical relaxation techniques.

9.4 Eigensystems of the Classical Methods

The ODE approach to relaxation can be summarized as follows. The basic equation to be solved came from some time-accurate derivation

$$A_b\vec{u} - \vec{f}_b = 0 \tag{9.46}$$

This equation is preconditioned in some manner which has the effect of multiplication by a conditioning matrix C giving

$$A\vec{\phi} - \vec{f} = 0 \tag{9.47}$$

An iterative scheme is developed to find the converged, or steady-state, solution of the set of ODE's

$$H\frac{d\vec{\phi}}{dt} = A\vec{\phi} - \vec{f} \tag{9.48}$$

This solution has the analytical form

$$\vec{\phi}_n = \vec{e}_n + \vec{\phi}_\infty \tag{9.49}$$

where $\vec{e}_n$ is the transient, or error, and $\vec{\phi}_\infty \equiv A^{-1}\vec{f}$ is the steady-state solution. The three classical methods, Point-Jacobi, Gauss-Seidel, and SOR, are identified for the one-dimensional case by Eq. 9.44 (solved using the explicit Euler method with $h = 1$) and Table 9.1.

Given our assumption that the component of the error associated with each eigenvector is equally likely to be excited, the asymptotic convergence rate is determined by the eigenvalue σ_m of G ($\equiv I + H^{-1}A$) having maximum absolute value. Thus

$$\text{Convergence rate} \quad \sim \quad |\sigma_m|_{max} \quad , \quad m = 1, 2, \cdots, M \tag{9.50}$$

In this section, we use the ODE analysis to find the convergence rates of the three classical methods represented by Eqs. 9.30, 9.31, and 9.32. It is also instructive to inspect the eigenvectors and eigenvalues in the $H^{-1}A$ matrix for the three methods. This amounts to solving the generalized eigenvalue problem

$$A\vec{x}_m = \lambda_m H\vec{x}_m \tag{9.51}$$

for the special case

$$B(1, -2, 1)\vec{x}_m = \lambda_m B(-\beta, \frac{2}{\omega}, 0)\vec{x}_m \tag{9.52}$$

The generalized eigensystem for simple tridiagonals is given in Appendix B.2. The three special cases considered below are obtained with $a = 1$, $b = -2$, $c = 1$, $d = -\beta$, $e = 2/\omega$, and $f = 0$. To illustrate the behavior, we take $M = 5$ for the matrix order. This special case makes the general result quite clear.

9.4.1 The Point-Jacobi System

If $\beta = 0$ and $\omega = 1$ in Eq. 9.44, the ODE matrix $H^{-1}A$ reduces to simply $B(\frac{1}{2}, -1, \frac{1}{2})$. The eigensystem can be determined from Appendix B.1 since both d and f are zero. The eigenvalues are given by the equation

$$\lambda_m = -1 + \cos\left(\frac{m\pi}{M+1}\right), \quad m = 1, 2, \ldots, M \tag{9.53}$$

The λ-σ relation for the explicit Euler method is $\sigma_m = 1 + \lambda_m h$. This relation can be plotted for any h. The plot for $h = 1$, the optimum stationary case, is shown in Fig. 9.1. For $h < 1$, the maximum $|\sigma_m|$ is obtained with $m = 1$, Fig. 9.2 and for $h > 1$, the maximum $|\sigma_m|$ is obtained with $m = M$, Fig. 9.3. Note that for $h > 1.0$ (depending on M) there is the possibility of instability, i.e. $|\sigma_m| > 1.0$. To obtain the optimal scheme we wish to minimize the maximum $|\sigma_m|$ which occurs when $h = 1$, $|\sigma_1| = |\sigma_M|$ and the best possible convergence rate is achieved:

$$|\sigma_m|_{max} = \cos\left(\frac{\pi}{M+1}\right) \tag{9.54}$$

For $M = 40$, we obtain $|\sigma_m|_{max} = 0.9971$. Thus after 500 iterations the error content associated with each eigenvector is reduced to no more than 0.23 times its initial level.

Again from Appendix B.1, the eigenvectors of $H^{-1}A$ are given by

$$\vec{x}_j = (x_j)_m = \sin\left[j\left(\frac{m\pi}{M+1}\right)\right], \quad j = 1, 2, \ldots, M \tag{9.55}$$

This is a very "well-behaved" eigensystem with linearly independent eigenvectors and distinct eigenvalues. The first 5 eigenvectors are simple sine waves. For $M = 5$, the eigenvectors can be written as

$$\vec{x}_1 = \begin{bmatrix} 1/2 \\ \sqrt{3}/2 \\ 1 \\ \sqrt{3}/2 \\ 1/2 \end{bmatrix}, \vec{x}_2 = \begin{bmatrix} \sqrt{3}/2 \\ \sqrt{3}/2 \\ 0 \\ -\sqrt{3}/2 \\ -\sqrt{3}/2 \end{bmatrix}, \vec{x}_3 = \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 1 \end{bmatrix},$$
$$\vec{x}_4 = \begin{bmatrix} \sqrt{3}/2 \\ -\sqrt{3}/2 \\ 0 \\ \sqrt{3}/2 \\ -\sqrt{3}/2 \end{bmatrix}, \vec{x}_5 = \begin{bmatrix} 1/2 \\ -\sqrt{3}/2 \\ 1 \\ -\sqrt{3}/2 \\ 1/2 \end{bmatrix} \tag{9.56}$$

The corresponding eigenvalues are, from Eq. 9.53

$$\begin{aligned} \lambda_1 = &\quad -1 + \frac{\sqrt{3}}{2} &= -0.134\cdots \\ \lambda_2 = &\quad -1 + \frac{1}{2} &= -0.5 \\ \lambda_3 = &\quad -1 &= -1.0 \\ \lambda_4 = &\quad -1 - \frac{1}{2} &= -1.5 \\ \lambda_5 = &\quad -1 - \frac{\sqrt{3}}{2} &= -1.866\cdots \end{aligned} \tag{9.57}$$

From Eq. 9.39, the numerical solution written in full is

$$\begin{aligned} \vec{\phi}_n - \vec{\phi}_\infty &= c_1[1 - (1 - \frac{\sqrt{3}}{2})h]^n \vec{x}_1 \\ &+ c_2[1 - (1 - \frac{1}{2})h]^n \vec{x}_2 \end{aligned}$$

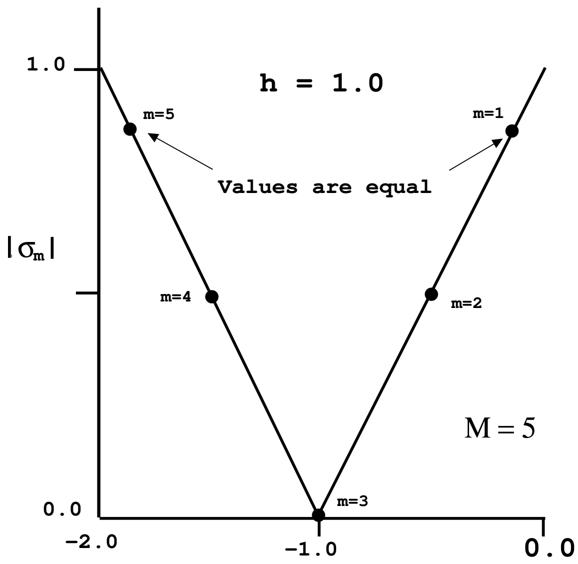

Figure 9.1: The σ, λ relation for Point-Jacobi, $h = 1, M = 5$.

$$
\begin{aligned}
&+ \quad c_3[1-(1 \qquad\quad)h]^n \vec{x}_3 \\
&+ \quad c_4[1-(1+\frac{1}{2})h]^n \vec{x}_4 \\
&+ \quad c_5[1-(1+\frac{\sqrt{3}}{2})h]^n \vec{x}_5
\end{aligned} \tag{9.58}
$$

9.4.2 The Gauss-Seidel System

If β and ω are equal to 1 in Eq. 9.44, the matrix eigensystem evolves from the relation

$$B(1,-2,1)\vec{x}_m = \lambda_m B(-1,2,0)\vec{x}_m \tag{9.59}$$

which can be studied using the results in Appendix B.2. One can show that the $H^{-1}A$ matrix for the Gauss-Seidel method, A_{GS}, is

$$A_{GS} \equiv B^{-1}(-1,2,0)B(1,-2,1) =$$

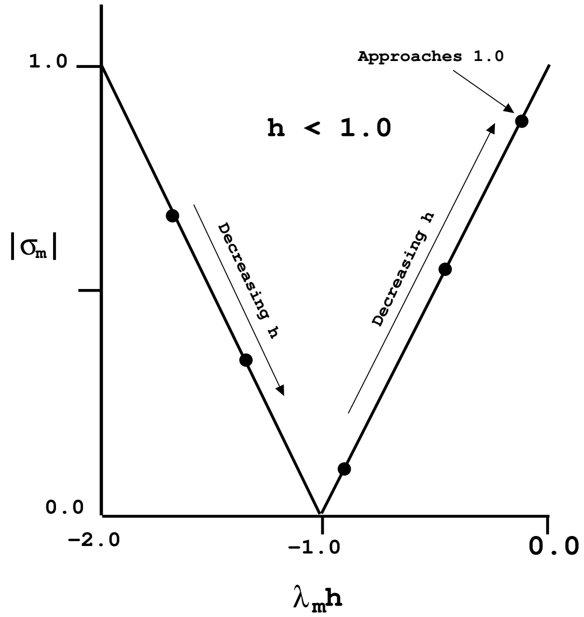

Figure 9.2: The σ, λ relation for Point-Jacobi, $h = 0.9, M = 5$.

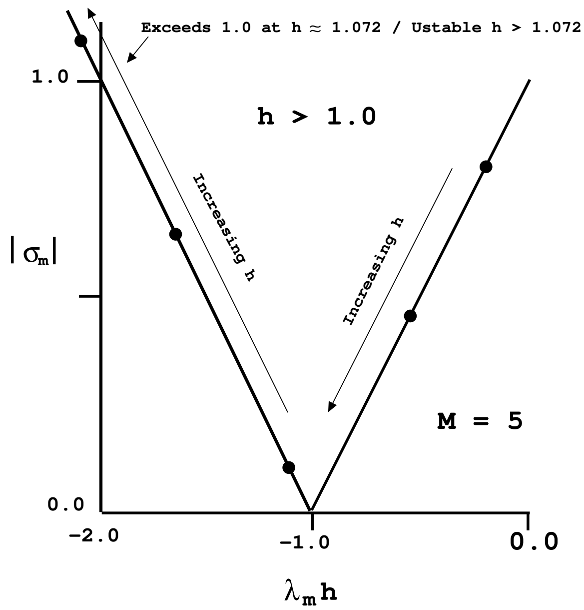

Figure 9.3: The σ, λ relation for Point-Jacobi, $h = 1.1, M = 5$.

$$
\begin{bmatrix}
-1 & 1/2 & & & & & \\
0 & -3/4 & 1/2 & & & & \\
0 & 1/8 & -3/4 & 1/2 & & & \\
0 & 1/16 & 1/8 & -3/4 & 1/2 & & \\
0 & 1/32 & 1/16 & 1/8 & -3/4 & 1/2 & \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & 1/2^M & \cdots & & & &
\end{bmatrix}
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ \vdots \\ M \end{array}
\tag{9.60}
$$

The eigenvector structure of the Gauss-Seidel ODE matrix is quite interesting. If M is odd there are $(M+1)/2$ distinct eigenvalues with corresponding linearly independent eigenvectors, and there are $(M-1)/2$ defective eigenvalues with corresponding principal vectors. The equation for the nondefective eigenvalues in the ODE matrix is (for odd M)

$$
\lambda_m = -1 + \cos^2\!\left(\frac{m\pi}{M+1}\right) \quad , \quad m = 1, 2, \ldots, \frac{M+1}{2}
\tag{9.61}
$$

and the corresponding eigenvectors are given by

$$
\vec{x}_m = \left[\cos\left(\frac{m\pi}{M+1}\right)\right]^{j-1} \sin\left[j\left(\frac{m\pi}{M+1}\right)\right] \quad , \quad m = 1, 2, \ldots, \frac{M+1}{2}
\tag{9.62}
$$

The λ-σ relation for $h = 1$, the optimum stationary case, is shown in Fig. 9.4. The σ_m with the largest amplitude is obtained with $m = 1$. Hence the convergence rate is

$$
|\sigma_m|_{max} = \left[\cos\left(\frac{\pi}{M+1}\right)\right]^2
\tag{9.63}
$$

Since this is the square of that obtained for the Point-Jacobi method, the error associated with the "worst" eigenvector is removed in half as many iterations. For $M = 40$, $|\sigma_m|_{max} = 0.9942$. 250 iterations are required to reduce the error component of the worst eigenvector by a factor of roughly 0.23.

The eigenvectors are quite unlike the Point-Jacobi set. They are no longer symmetrical, producing waves that are higher in amplitude on one side (the updated side) than they are on the other. Furthermore, they do not represent a common family for different values of M.

The Jordan canonical form for $M = 5$ is

$$
X^{-1} A_{GS} X = J_{GS} =
\begin{bmatrix}
\left[\vec{\phi}_1\right] & & \\
& \left[\vec{\phi}_2\right] & \\
& & \begin{bmatrix} \vec{\phi}_3 & 1 & \\ & \vec{\phi}_3 & 1 \\ & & \vec{\phi}_3 \end{bmatrix}
\end{bmatrix}
\tag{9.64}
$$

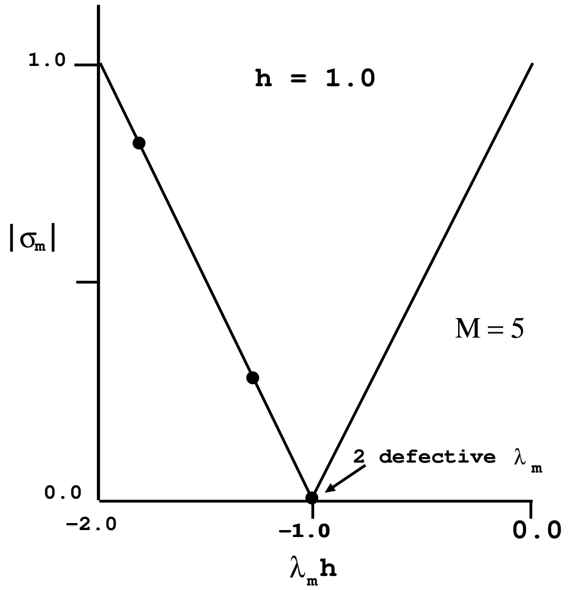

Figure 9.4: The σ, λ relation for Gauss-Seidel, $h = 1.0, M = 5$.

The eigenvectors and principal vectors are all real. For $M = 5$ they can be written

$$\vec{x}_1 = \begin{bmatrix} 1/2 \\ 3/4 \\ 3/4 \\ 9/16 \\ 9/32 \end{bmatrix}, \vec{x}_2 = \begin{bmatrix} \sqrt{3}/2 \\ \sqrt{3}/4 \\ 0 \\ -\sqrt{3}/16 \\ -\sqrt{3}/32 \end{bmatrix}, \vec{x}_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \vec{x}_4 = \begin{bmatrix} 0 \\ 2 \\ -1 \\ 0 \\ 0 \end{bmatrix}, \vec{x}_5 = \begin{bmatrix} 0 \\ 0 \\ 4 \\ -4 \\ 1 \end{bmatrix} \tag{9.65}$$

The corresponding eigenvalues are

$$\begin{aligned} \lambda_1 &= -1/4 \\ \lambda_2 &= -3/4 \\ \lambda_3 &= -1 \\ \left.\begin{matrix} (4) \\ (5) \end{matrix}\right\} & \quad \begin{matrix} \text{Defective, linked to} \\ \lambda_3 \text{ Jordan block} \end{matrix} \end{aligned} \tag{9.66}$$

The numerical solution written in full is thus

$$\begin{aligned} \vec{\phi}_n - \vec{\phi}_\infty &= c_1(1 - \frac{h}{4})^n \vec{x}_1 \\ &+ c_2(1 - \frac{3h}{4})^n \vec{x}_2 \end{aligned}$$

$$\begin{aligned}
&+ \left[c_3(1-h)^n + c_4 h\frac{n}{1!}(1-h)^{n-1} + c_5 h^2 \frac{n(n-1)}{2!}(1-h)^{n-2}\right]\vec{x}_3 \\
&+ \left[c_4(1-h)^n + c_5 h\frac{n}{1!}(1-h)^{n-1}\right]\vec{x}_4 \\
&+ \ c_5(1-h)^n \vec{x}_5
\end{aligned} \tag{9.67}$$

9.4.3 The SOR System

If $\beta = 1$ and $2/\omega = x$ in Eq. 9.44, the ODE matrix is $B^{-1}(-1, x, 0)B(1, -2, 1)$. One can show that this can be written in the form given below for $M = 5$. The generalization to any M is fairly clear. The $H^{-1}A$ matrix for the SOR method, $A_{SOR} \equiv B^{-1}(-1, x, 0)B(1, -2, 1)$, is

$$\frac{1}{x^5}\begin{bmatrix}
-2x^4 & x^4 & 0 & 0 & 0 \\
-2x^3 + x^4 & x^3 - 2x^4 & x^4 & 0 & 0 \\
-2x^2 + x^3 & x^2 - 2x^3 + x^4 & x^3 - 2x^4 & x^4 & 0 \\
-2x + x^2 & x - 2x^2 + x^3 & x^2 - 2x^3 + x^4 & x^3 - 2x^4 & x^4 \\
-2 + x & 1 - 2x + x^2 & x - 2x^2 + x^3 & x^2 - 2x^3 + x^4 & x^3 - 2x^4
\end{bmatrix} \tag{9.68}$$

Eigenvalues of the system are given by

$$\lambda_m = -1 + \left(\frac{\omega p_m + z_m}{2}\right)^2 \quad , \quad m = 1, 2, \ldots M \tag{9.69}$$

where

$$\begin{aligned}
z_m &= [4(1-\omega) + \omega^2 p_m^2]^{1/2} \\
p_m &= \cos[m\pi/(M+1)]
\end{aligned}$$

If $\omega = 1$, the system is Gauss-Seidel. If $4(1-\omega) + \omega^2 p_m^2 < 0$, z_m and λ_m are complex. If ω is chosen such that $4(1-\omega) + \omega^2 p_1^2 = 0$, ω is optimum for the stationary case, and the following conditions hold:

1. Two eigenvalues are real, equal and defective.

2. If M is even, the remaining eigenvalues are complex and occur in conjugate pairs.

3. If M is odd, one of the remaining eigenvalues is real and the others are complex occurring in conjugate pairs.

One can easily show that the optimum ω for the stationary case is

$$\omega_{opt} = 2\Big/\left[1 + \sin\left(\frac{\pi}{M+1}\right)\right] \tag{9.70}$$

and for $\omega = \omega_{opt}$

$$\begin{aligned} \lambda_m &= \zeta_m^2 - 1 \\ \vec{x}_m &= \zeta_m^{j-1} \sin\left[j\left(\frac{m\pi}{M+1}\right)\right] \end{aligned} \tag{9.71}$$

where

$$\zeta_m = \frac{\omega_{opt}}{2}\left[p_m + i\sqrt{p_1^2 - p_m^2}\right]$$

Using the explicit Euler method to integrate the ODE's, $\sigma_m = 1 - h + h\zeta_m^2$, and if $h = 1$, the optimum value for the stationary case, the λ-σ relation reduces to that shown in Fig. 9.5. This illustrates the fact that for optimum stationary SOR all the $|\sigma_m|$ are identical and equal to $\omega_{opt} - 1$. Hence the convergence rate is

$$\begin{aligned} |\sigma_m|_{max} &= \omega_{opt} - 1 \\ \omega_{opt} &= 2\Big/\left[1 + \sin\left(\frac{\pi}{M+1}\right)\right] \end{aligned} \tag{9.72}$$

For $M = 40$, $|\sigma_m|_{max} = 0.8578$. Hence the worst error component is reduced to less than 0.23 times its initial value in only 10 iterations, much faster than both Gauss-Seidel and Point-Jacobi. In practical applications, the optimum value of ω may have to be determined by trial and error, and the benefit may not be as great.

For odd M, there are two real eigenvectors and one real principal vector. The remaining linearly independent eigenvectors are all complex. For $M = 5$ they can be written

$$\vec{x}_1 = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/3 \\ 1/6 \\ 1/18 \end{bmatrix}, \vec{x}_2 = \begin{bmatrix} -6 \\ 9 \\ 16 \\ 13 \\ 6 \end{bmatrix}, \vec{x}_{3,4} = \begin{bmatrix} \sqrt{3}(1 \qquad)/2 \\ \sqrt{3}(1 \pm \ i\sqrt{2})/6 \\ 0 \\ \sqrt{3}(5 \pm \ i\sqrt{2})/54 \\ \sqrt{3}(7 \pm 4i\sqrt{2})/162 \end{bmatrix}, \vec{x}_5 = \begin{bmatrix} 1 \\ 0 \\ 1/3 \\ 0 \\ 1/9 \end{bmatrix} \tag{9.73}$$

The corresponding eigenvalues are

$$\begin{aligned} \lambda_1 &= -2/3 \\ (2) & \quad \text{Defective linked to} \quad \lambda_1 \\ \lambda_3 &= -(10 - 2\sqrt{2}i)/9 \\ \lambda_4 &= -(10 + 2\sqrt{2}i)/9 \\ \lambda_5 &= -4/3 \end{aligned} \tag{9.74}$$

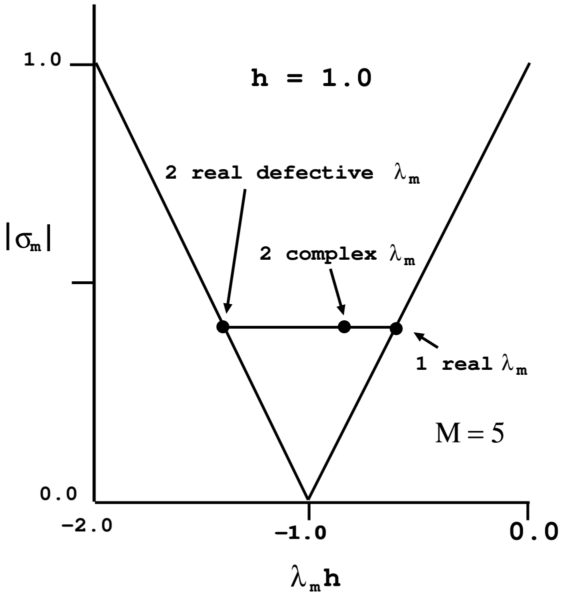

Figure 9.5: The σ, λ relation for optimum stationary SOR, $M = 5$, $h = 1$.

The numerical solution written in full is

$$\begin{aligned}
\vec{\phi}_n - \vec{\phi}_\infty &= [c_1(1-2h/3)^n + c_2 nh(1-2h/3)^{n-1}]\vec{x}_1 \\
&+ c_2(1-2h/3)^n\vec{x}_2 \\
&+ c_3[1-(10-2\sqrt{2}i)h/9]^n\vec{x}_3 \\
&+ c_4[1-(10+2\sqrt{2}i)h/9]^n\vec{x}_4 \\
&+ c_5(1-4h/3)^n\vec{x}_5
\end{aligned} \tag{9.75}$$

9.5 Nonstationary Processes

In classical terminology a method is said to be nonstationary if the conditioning matrices, H and C, are varied at each time step. This does not change the steady-state solution $A_b^{-1}\vec{f}_b$, but it can greatly affect the convergence rate. In our ODE approach this could also be considered and would lead to a study of equations with nonconstant coefficients. It is much simpler, however, to study the case of fixed H and C but variable step size, h. This process changes the Point-Jacobi method to Richardson's method in standard terminology. For the Gauss-Seidel and SOR methods it leads to processes that can be superior to the stationary methods.

The nonstationary form of Eq. 9.39 is

$$
\begin{aligned}
\vec{\phi}_N &= c_1\vec{x}_1\prod_{n=1}^{N}(1+\lambda_1 h_n)+\cdots+c_m\vec{x}_m\prod_{n=1}^{N}(1+\lambda_m h_n) \\
&+ \cdots+c_M\vec{x}_M\prod_{n=1}^{N}(1+\lambda_M h_n)+\vec{\phi}_\infty
\end{aligned} \tag{9.76}
$$

where the symbol Π stands for product. Since h_n can now be changed at each step, the error term can theoretically be completely eliminated in M steps by taking $h_m = -1/\lambda_m$, for $m = 1, 2, \cdots, M$. However, the eigenvalues λ_m are generally unknown and costly to compute. It is therefore unnecessary and impractical to set $h_m = -1/\lambda_m$ for $m = 1, 2, \ldots, M$. We will see that a few well chosen h's can reduce whole clusters of eigenvectors associated with nearby λ's in the λ_m spectrum. This leads to the concept of selectively annihilating clusters of eigenvectors from the error terms as part of a total iteration process. This is the basis for the multigrid methods discussed in Chapter 10.

Let us consider the very important case when all of the λ_m are *real* and *negative* (remember that they arise from a conditioned matrix so this constraint is not unrealistic for quite practical cases). Consider one of the error terms taken from

$$
\vec{e}_N \equiv \vec{\phi}_N - \vec{\phi}_\infty = \sum_{m=1}^{M} c_m\vec{x}_m\prod_{n=1}^{N}(1+\lambda_m h_n) \tag{9.77}
$$

and write it in the form

$$
c_m\vec{x}_m P_e(\lambda_m) \equiv c_m\vec{x}_m\prod_{n=1}^{N}(1+\lambda_m h_n) \tag{9.78}
$$

where P_e signifies an "Euler" polynomial. Now focus attention on the polynomial

$$
(P_e)_N(\lambda) = (1+h_1\lambda)(1+h_2\lambda)\cdots(1+h_N\lambda) \tag{9.79}
$$

treating it as a continuous function of the independent variable λ. In the annihilation process mentioned after Eq. 9.76, we considered making the error exactly zero by taking advantage of some knowledge about the discrete values of λ_m for a particular case. Now we pose a less demanding problem. Let us choose the h_n so that the maximum value of $(P_e)_N(\lambda)$ is as small as possible for all λ lying between λ_a and λ_b such that $\lambda_b \le \lambda \le \lambda_a \le 0$. Mathematically stated, we seek

$$
\max_{\lambda_b \le \lambda \le \lambda_a} |(P_e)_N(\lambda)| = \text{minimum} \quad , \quad \text{with} (P_e)_N(0) = 1 \tag{9.80}
$$

This problem has a well known solution due to Markov. It is

$$(P_e)_N(\lambda) = \frac{T_N\left(\dfrac{2\lambda - \lambda_a - \lambda_b}{\lambda_a - \lambda_b}\right)}{T_N\left(\dfrac{-\lambda_a - \lambda_b}{\lambda_a - \lambda_b}\right)} \tag{9.81}$$

where

$$T_N(y) = \cos(N \arccos y) \tag{9.82}$$

are the Chebyshev polynomials along the interval $-1 \le y \le 1$ and

$$T_N(y) = \frac{1}{2}\left[y + \sqrt{y^2 - 1}\right]^N + \frac{1}{2}\left[y - \sqrt{y^2 - 1}\right]^N \tag{9.83}$$

are the Chebyshev polynomials for $|y| > 1$. In relaxation terminology this is generally referred to as Richardson's method, and it leads to the nonstationary step size choice given by

$$\frac{1}{h_n} = \frac{1}{2}\left\{-\lambda_b - \lambda_a + (\lambda_b - \lambda_a)\cos\left[\frac{(2n-1)\pi}{2N}\right]\right\}, \quad n = 1, 2, \ldots N \tag{9.84}$$

Remember that all λ are negative real numbers representing the magnitudes of λ_m in an eigenvalue spectrum.

The error in the relaxation process represented by Eq. 9.76 is expressed in terms of a set of eigenvectors, $\vec{x}_m$, amplified by the coefficients $c_m \prod(1 + \lambda_m h_n)$. With each eigenvector there is a corresponding eigenvalue. Eq. 9.84 gives us the best choice of a series of h_n that will minimize the amplitude of the error carried in the eigenvectors associated with the eigenvalues between λ_b and λ_a.

As an example for the use of Eq. 9.84, let us consider the following problem:

Minimize the maximum error associated with the λ eigenvalues in the interval $-2 \le \lambda \le -1$ using only 3 iterations.	(9.85)

The three values of h which satisfy this problem are

$$h_n = 2\Big/\left(3 - \cos\left[\frac{(2n-1)\pi}{6}\right]\right) \tag{9.86}$$

and the amplitude of the eigenvector is reduced to

$$(P_e)_3(\lambda) = T_3(2\lambda + 3)/T_3(3) \tag{9.87}$$

where

$$T_3(3) = \left\{[3+\sqrt{8}]^3 + [3-\sqrt{8}]^3\right\}/2 \approx 99 \tag{9.88}$$

A plot of Eq. 9.87 is given in Fig. 9.6 and we see that the amplitudes of all the eigenvectors associated with the eigenvalues in the range $-2 \le \lambda \le -1$ have been reduced to less than about 1% of their initial values. The values of h used in Fig. 9.6 are

$$\begin{aligned} h_1 &= 4/(6-\sqrt{3}) \\ h_2 &= 4/(6-0) \\ h_3 &= 4/(6+\sqrt{3}) \end{aligned}$$

Return now to Eq. 9.76. This was derived from Eq. 9.37 on the condition that the explicit Euler method, Eq. 9.41, was used to integrate the basic ODE's. If instead the implicit trapezoidal rule

$$\phi_{n+1} = \phi_n + \frac{1}{2}h(\phi'_{n+1} + \phi'_n) \tag{9.89}$$

is used, the nonstationary formula

$$\vec{\phi}_N = \sum_{m=1}^{M} c_m \vec{x}_m \prod_{n=1}^{N} \left(\frac{1 + \frac{1}{2}h_n\lambda_m}{1 - \frac{1}{2}h_n\lambda_m} \right) + \vec{\phi}_\infty \tag{9.90}$$

would result. This calls for a study of the rational "trapezoidal" polynomial, P_t:

$$(P_t)_N(\lambda) = \prod_{n=1}^{N} \left(\frac{1 + \frac{1}{2}h_n\lambda}{1 - \frac{1}{2}h_n\lambda} \right) \tag{9.91}$$

under the same constraints as before, namely that

$$\begin{aligned} \max_{\lambda_b \le \lambda \le \lambda_a} |(P_t)_N(\lambda)| &= \text{minimum}\ , \\ \text{with } (P_t)_N(0) &= 1 \end{aligned} \tag{9.92}$$

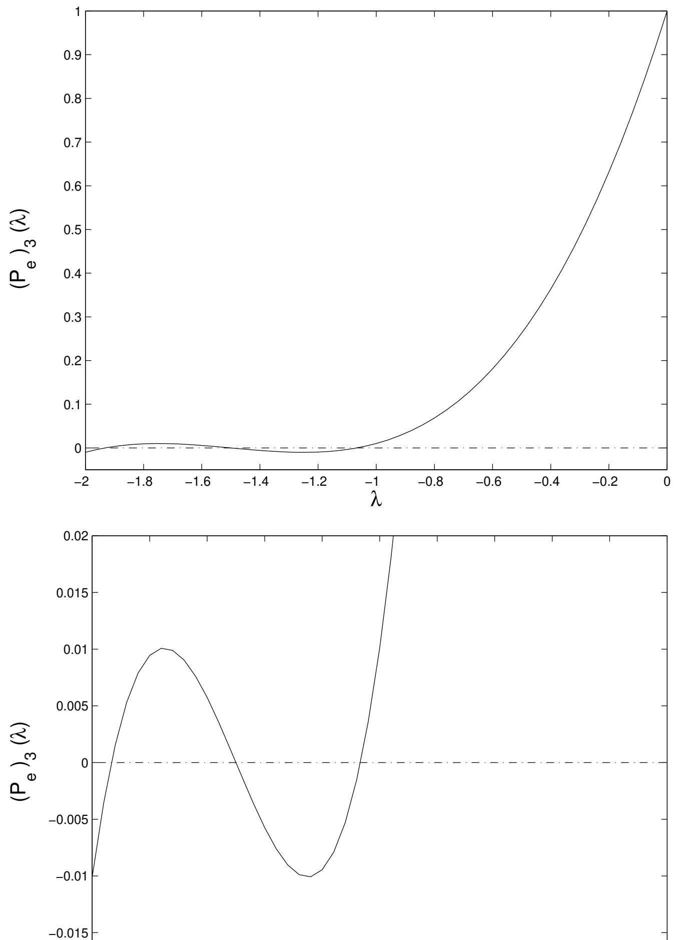

Figure 9.6: Richardson's method for 3 steps, minimization over $-2 \le \lambda \le -1$.

The optimum values of h can also be found for this problem, but we settle here for the approximation suggested by Wachspress

$$\frac{2}{h_n} = -\lambda_b \left(\frac{\lambda_a}{\lambda_b}\right)^{(n-1)/(N-1)} \quad , \quad n = 1, 2, \cdots, N \tag{9.93}$$

This process is also applied to problem 9.85. The results for $(P_t)_3(\lambda)$ are shown in Fig. 9.7. The error amplitude is about 1/5 of that found for $(P_e)_3(\lambda)$ in the same interval of λ. The values of h used in Fig. 9.7 are

$$\begin{aligned} h_1 &= 1 \\ h_2 &= \sqrt{2} \\ h_3 &= 2 \end{aligned}$$

9.6 Problems

1. Given a relaxation method in the form

$$H \Delta \vec{\phi}_n = A \vec{\phi}_n - \vec{f}$$

show that

$$\vec{\phi}_n = G^n \vec{\phi}_0 + (I - G^n) A^{-1} \vec{f}$$

where $G = I + H^{-1}A$.

2. For a linear system of the form $(A_1 + A_2)x = b$, consider the iterative method

$$\begin{aligned} (I + \mu A_1)\tilde{x} &= (I - \mu A_2)x_n + \mu b \\ (I + \mu A_2)x_{n+1} &= (I - \mu A_1)\tilde{x} + \mu b \end{aligned}$$

where μ is a parameter. Show that this iterative method can be written in the form

$$H(x_{k+1} - x_k) = (A_1 + A_2)x_k - b$$

Determine the iteration matrix G if $\mu = -1/2$.

3. Using Appendix B.2, find the eigenvalues of $H^{-1}A$ for the SOR method with $A = B(4 : 1, -2, 1)$ and $\omega = \omega_{opt}$. (You do not have to find H^{-1}. Recall that the eigenvalues of $H^{-1}A$ satisfy $A\vec{x}_m = \lambda_m H \vec{x}_m$.) Find the numerical values, not just the expressions. Then find the corresponding $|\sigma_m|$ values.

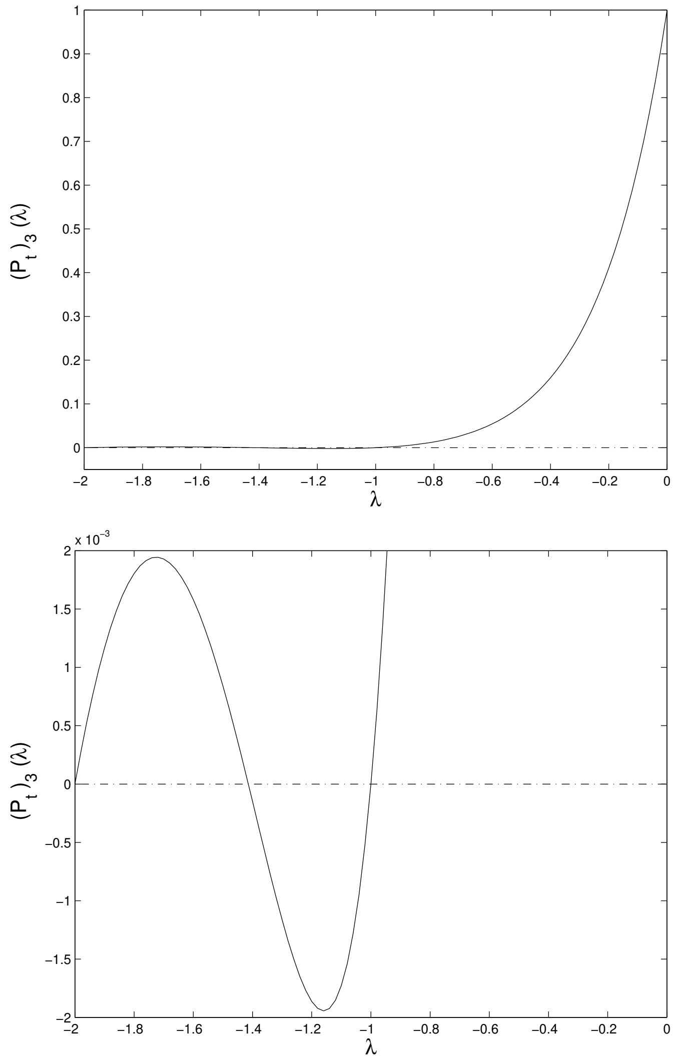

Figure 9.7: Wachspress method for 3 steps, minimization over $-2 \leq \lambda \leq -1$.

4. Solve the following equation on the domain $0 \le x \le 1$ with boundary conditions $u(0) = 0$, $u(1) = 1$:

$$\frac{\partial^2 u}{\partial x^2} - 6x = 0$$

For the initial condition, use $u(x) = 0$. Use second-order centered differences on a grid with 40 cells ($M = 39$). Iterate to steady state using

(a) the point-Jacobi method,

(b) the Gauss-Seidel method,

(c) the SOR method with the optimum value of ω, and

(d) the 3-step Richardson method derived in Section 9.5.

Plot the solution after the residual is reduced by 2, 3, and 4 orders of magnitude. Plot the logarithm of the L_2-norm of the residual vs. the number of iterations. Determine the asymptotic convergence rate. Compare with the theoretical asymptotic convergence rate.

Bibliography

[1] D. Anderson, J. Tannehill, and R. Pletcher. *Computational Fluid Mechanics and Heat Transfer*. McGraw-Hill Book Company, New York, 1984.

[2] C. Hirsch. *Numerical Computation of Internal and External Flows*, volume 1,2. John Wiley & Sons, New York, 1988.