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 O∆E'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 Kramer'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

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

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

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

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

The σ-root is found from

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

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

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

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

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

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

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

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.

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

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
>
> $$\sigma = 1 + \lambda h + \frac{1}{2}\lambda^2 h^2 + \cdots + \frac{1}{k!}\lambda^k h^k + O\left(h^{k+1}\right) \tag{6.33}$$
>
> where k is the order of the time-marching method.

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

$$\begin{aligned} u_n \;=\;& c_{11}(\sigma_1)_1^n\,\vec{x}_1 + \cdots + c_{m1}(\sigma_m)_1^n\,\vec{x}_m + \cdots + c_{M1}(\sigma_M)_1^n\,\vec{x}_M + P.S. \\ & + c_{12}(\sigma_1)_2^n\,\vec{x}_1 + \cdots + c_{m2}(\sigma_m)_2^n\,\vec{x}_m + \cdots + c_{M2}(\sigma_M)_2^n\,\vec{x}_M \\ & + c_{13}(\sigma_1)_3^n\,\vec{x}_1 + \cdots + c_{m3}(\sigma_m)_3^n\,\vec{x}_m + \cdots + c_{M3}(\sigma_M)_3^n\,\vec{x}_M \\ & + \text{etc., if there are more spurious roots} \end{aligned} \tag{6.36}$$

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

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

where

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

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

$$er_\lambda = c_1 \cdot (\lambda h)^{k_1+1} \tag{6.45}$$

$$er_\mu = c_2 \cdot (\lambda h)^{k_2+1} \tag{6.46}$$

$$\text{where } k = \text{smallest of}(k_1, k_2) \tag{6.47}$$

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

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

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

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

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

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

Implicit Methods

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

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

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

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

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 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

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

is a second-order accurate method for any α.

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

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

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

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 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 $er_\mu = O(k^5)$, 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

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

[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.

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 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

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

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

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

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 $h \to \infty$ 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 $h \to \infty$ 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

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

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]

[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.

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

where $t - t_n$ and 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

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

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.