CHAPTER 10

Numerical Calculus

10.0 Introduction

This chapter concerns numerical differentiation, numerical integration, and the numerical solution of ordinary differential equations using Runge–Kutta methods. More specifically, the first two sections consider numerical approximations to derivatives and integrals based on samples $f(x_i)$ where $x_0 \leq x_1 \leq \cdots \leq x_{N-1} \leq x_N$. The differentiation and integration formulae found in this chapter are the foundation of the finite-difference and finite-volume methods used in computational gasdynamics, seen starting in Chapter 11. The last section of this chapter concerns Runge–Kutta methods for the numerical solution of ordinary differential equations. Of course, the equations seen in computational gasdynamics are partial differential equations rather than ordinary differential equations. In other words, the gasdynamic equations such as the differential forms of the Euler equations involve two or more independent variables rather than a single independent variable. However, after discretizing all but one independent variable, usually time, partial differential equations may be approximated by ordinary differential equations as seen in Subsection 11.2.1.

10.1 Numerical Differentiation

If a function is represented by samples, the samples must first be converted to a functional representation using interpolation or some other reconstruction procedure. The functional representation can then be differentiated using the usual rules of calculus. This section concerns differentiation formulae based on polynomial interpolation.

10.1.1 *Linear Approximations*

The line passing through any two samples $(x_i, f(x_i))$ and $(x_{i+1}, f(x_{i+1}))$ is as follows:

$$f(x) \approx p_1(x) = f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i). \tag{10.1}$$

Then

$$\frac{df}{dx}(x) \approx \frac{dp_1}{dx}(x) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}. \tag{10.2}$$

Notice that this approximation is constant. It is called a *backward difference* if $x = x_{i+1}$, a *forward difference* if $x = x_i$, and a *central difference* if $x = (x_{i+1} + x_i)/2$. However, only the name of the approximation changes with x – the approximation itself is constant and does not depend on x.

Assuming that $f(x)$ has a continuous second derivative, Equation (8.13) gives the following expression for the error of Equation (10.1):

$$e_1(x) = f(x) - p_1(x) = (x - x_i)(x - x_{i+1})\frac{1}{2}\frac{d^2 f(\xi)}{dx^2}$$

for all $x_i \le x \le x_{i+1}$, where $x_i \le \xi(x) \le x_{i+1}$. Then the error of Equation (10.2) is

$$\frac{de_1}{dx}(x) = \left(x - \frac{x_{i+1} + x_i}{2}\right)\frac{d^2f(\xi)}{dx^2} + \frac{1}{2}(x - x_i)(x - x_{i+1})\frac{d^3f(\xi)}{dx_3}\frac{d\xi}{dx}. \tag{10.3}$$

In particular, for backward differences

$$\frac{de_1}{dx}(x_{i+1}) = \frac{x_{i+1} - x_i}{2}\frac{d^2f(\xi)}{dx^2} = O(x_{i+1} - x_i), \tag{10.4}$$

for forward differences

$$\frac{de_1}{dx}(x_i) = -\frac{x_{i+1} - x_i}{2}\frac{d^2f(\xi)}{dx^2} = O(x_{i+1} - x_i), \tag{10.5}$$

and for central differences

$$\begin{aligned}\frac{de_1}{dx}\left(\frac{x_{i+1} + x_i}{2}\right) &= \frac{1}{2}\left(\frac{x_{i+1} + x_i}{2} - x_i\right)\left(\frac{x_{i+1} + x_i}{2} - x_{i+1}\right)\frac{d^3f(\xi)}{dx_3}\frac{d\xi}{dx} \\ &= O((x_{i+1} - x_i)^2).\end{aligned} \tag{10.6}$$

Then for constant Δx we have

$$\frac{df}{dx}(x_{i+1}) = \frac{f(x_{i+1}) - f(x_i)}{\Delta x} + O(\Delta x), \tag{10.7}$$

$$\frac{df}{dx}(x_i) = \frac{f(x_{i+1}) - f(x_i)}{\Delta x} + O(\Delta x), \tag{10.8}$$

$$\frac{df}{dx}\left(\frac{x_{i+1} + x_i}{2}\right) = \frac{f(x_{i+1}) - f(x_i)}{\Delta x} + O(\Delta x^2). \tag{10.9}$$

Notice that the central-difference approximation has a higher order of accuracy than the forward- and backward-difference approximations. This will be discussed further in the next subsection.

10.1.2 *Quadratic Approximations*

Using the results of Example 8.5, the quadratic passing through any three samples $(x_{i-1}, f(x_{i-1}))$, $(x_i, f(x_i))$, $(x_{i+1}, f(x_{i+1}))$ can be written as

$$p_2(x) = a_2(x - x_{i-1})^2 + a_1(x - x_{i-1}) + f(x_{i-1}), \tag{10.10}$$

where

$$a_2 = \frac{1}{x_{i+1} - x_{i-1}}\left(\frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} - \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}\right), \tag{10.11}$$

$$a_1 = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} - (x_i - x_{i-1})a_2. \tag{10.12}$$

Then

$$\frac{df}{dx}(x) \approx \frac{dp_2}{dx}(x) = 2a_2(x - x_{i-1}) + a_1. \tag{10.13}$$

For example, if $x = x_i$ then

$$\begin{aligned}\frac{df}{dx}(x_i) &\approx \frac{dp_2}{dx}(x_i) = 2a_2(x_i - x_{i-1}) + a_1 \\ &= 2a_2(x_i - x_{i-1}) + \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} - (x_i - x_{i-1})a_2 \\ &= a_2(x_i - x_{i-1}) + \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}\end{aligned}$$

or

$$\begin{aligned}\frac{df}{dx}(x_i) \approx \frac{x_i - x_{i-1}}{x_{i+1} - x_{i-1}} &\left(\frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} - \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} \right) \\ &+ \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}\end{aligned}$$

or

$$\frac{df}{dx}(x_i) \approx \theta_i \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} + (1 - \theta_i)\frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}, \tag{10.14}$$

where

$$\theta_i = \frac{x_i - x_{i-1}}{x_{i+1} - x_{i-1}}$$

and where $0 \leq \theta_i \leq 1$. The attentive reader may recall that a similar expression appeared in Equation (5.42). As with Equation (5.42), the expression in Equation (10.14) is called a *linear average*, *linear interpolation*, or a *convex linear combination* of first divided differences.

For constant Δx, Equation (10.14) becomes

$$\frac{df}{dx}(x_i) \approx \frac{f(x_{i+1}) - f(x_{i-1})}{2\Delta x}, \tag{10.15}$$

which is almost exactly the same as Equation (10.9). The main difference is the factor of two in the denominator, which reflects the fact that the samples x_{i-1} and x_{i+1} are spaced apart by $2\Delta x$ rather than by Δx. Unfortunately, approximation (10.15) wastefully ignores the middle sample $f(x_i)$. In particular, if the index i is even, then Equation (10.15) uses only the neighboring odd-indexed samples; similarly, if the index i is odd, then Equation (10.15) uses only the neighboring even-indexed samples. Then the even- and odd-indexed approximations draw from two completely different sets of samples, which can lead to a separation between the even- and odd-indexed approximations. Typical symptoms include large odd–even $2\Delta x$-wave oscillations in the approximate derivative. This phenomenon is sometimes called *odd–even decoupling*. Although the more general expression given by Equation (10.14) does not ignore the middle sample, and thus technically does not suffer from odd–even decoupling, it may behave similarly in many cases.

Much as in the last section, one can perform an order-of-error analysis using Equation (8.13); the details are omitted. For constant Δx,

♦ $$\frac{df}{dx}(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2\Delta x} + O(\Delta x^2). \tag{10.16}$$

Similarly,

$$\frac{df}{dx}(x_{i-1}) = \frac{-f(x_{i+1}) + 4f(x_i) - 3f(x_{i-1})}{2\Delta x} + O(\Delta x^2) \tag{10.17}$$

and

$$\frac{df}{dx}(x_{i+1}) = \frac{3f(x_{i+1}) - 4f(x_i) + f(x_{i-1})}{2\Delta x} + O(\Delta x^2). \tag{10.18}$$

These approximations are sometimes called *second-order forward-difference* and *second-order backward-difference*, respectively.

Quadratic interpolation can also be used to approximate second derivatives. In particular,

$$\frac{d^2f}{dx^2}(x) \approx \frac{d^2p_2}{dx^2}(x) = 2a_2. \tag{10.19}$$

For constant Δx,

$$a_2 = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{2\Delta x^2},$$

and then one can show that

$$\frac{d^2f}{dx^2}(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{\Delta x^2} + O(\Delta x^2). \tag{10.20}$$

By differentiating cubics, quartics, and higher-order interpolation polynomials, one can obtain higher-order differentiation formulae; the details are omitted. For a complete listing of differentiation formulae for the first through the sixth derivative using three to six samples, see Table 25.2 of Abramowitz and Stegun (1964).

10.2 Numerical Integration

If a function is represented by samples, the samples must first be converted to a functional representation using interpolation or some other reconstruction procedure. The functional representation can then be integrated using the usual rules of calculus. Before proceeding, some standard terminology is introduced. Suppose an integration formula can be written in the following form:

$$\int_a^b f(x)\,dx = \sum_{i=0}^{N} w_i f(x_i) + E_N, \tag{10.21}$$

where E_N is the error, w_i are constant *weightings*, and x_i are the sample points, also known as *nodes* in this context. Any such integration formula is called a *quadrature*. *Quadratures* based on polynomial interpolation are called *Newton–Cotes quadratures*. A Newton–Cotes quadrature is *closed* if the first and last samples equal the limits of integration, that is, if $a = x_0$ and $b = x_N$.

The error analysis in this section requires a result sometimes referred to as the second integral mean value theorem. Most readers will be familiar with the ordinary or *first integral mean value theorem*:

$$\int_a^b f(x)\,dx = f(c)\int_a^b dx = f(c)(b-a) \quad \text{for some} \quad a \le c \le b. \tag{10.22}$$

Now suppose $g(x)$ is a function that does not change sign on $[a, b]$. Then the *second integral mean value theorem* is

$$\int_a^b f(x)g(x)\,dx = f(c)\int_a^b g(x)\,dx \quad \text{for some} \quad a \le c \le b. \tag{10.23}$$

Notice that the second integral mean value theorem implies the first integral mean value theorem.

10.2.1 *Constant Approximations*

The constant passing through a single sample $(x_0, f(x_0))$ is, of course, $p_0(x) = f(x_0)$. Then

$$\int_a^b f(x)\,dx \approx \int_a^b p_0(x)\,dx = \int_a^b f(x_0)\,dx = (b-a)f(x_0). \tag{10.24}$$

Assuming that $f(x)$ has a continuous first derivative, we have

$$e_0(x) = f(x) - p_0(x) = f(x) - f(x_0) = (x - x_0)\frac{df(\xi)}{dx},$$

where $\xi(x)$ is somewhere between x and x_0. Then the error of Equation (10.24) is

$$E_0 = \int_a^b e_0(x)\,dx = \int_a^b (f(x) - p_0(x))\,dx = \int_a^b (x - x_0)\frac{df(\xi)}{dx}dx.$$

Divide the integration up into two regions such that $x - x_0$ does not change sign in either region to find

$$E_0 = \int_a^{x_0} (x - x_0)\frac{df(\xi)}{dx}dx + \int_{x_0}^b (x - x_0)\frac{df(\xi)}{dx}dx.$$

Apply the second integral mean value theorem, Equation (10.23), to find

$$E_0 = \frac{df\left(\xi\left(c_1'\right)\right)}{dx}\int_a^{x_0} (x - x_0)\,dx + \frac{df\left(\xi\left(c_2'\right)\right)}{dx}\int_{x_0}^b (x - x_0)\,dx.$$

Let $c_1 = \xi(c_1')$ and $c_2 = \xi(c_2')$. Then the final result is

$$E_0 = -\frac{1}{2}(x_0 - a)^2\frac{df(c_1)}{dx} + \frac{1}{2}(x_0 - b)^2\frac{df(c_2)}{dx}, \tag{10.25}$$

where c_1 and c_2 are both between a and b.

Suppose $b - a = \Delta x$. Notice that $E_0 = O(\Delta x^2)$ for any $a \le x_0 \le b$. Then

♦ $$\int_a^b f(x)\,dx = \Delta x f(x_0) + O(\Delta x^2) \tag{10.26}$$

for any $a \le x_0 \le b$. As seen in the following example, there is one exception where the order of accuracy is three rather than two.

Example 10.1 As with numerical differentiation, centering can improve the order of accuracy of numerical integration. In particular, suppose the limits of integration are $a = x_0 - \Delta x/2$ and $b = x_0 + \Delta x/2$. Show that Equation (10.26) is third-order accurate.

Solution By Equation (10.25), the error in Equation (10.26) is

$$E_0 = -\frac{1}{2}\left(\frac{\Delta x}{2}\right)^2 \frac{df(c_1)}{dx} + \frac{1}{2}\left(\frac{\Delta x}{2}\right)^2 \frac{df(c_2)}{dx} = \frac{1}{8}\Delta x^2\left(\frac{df(c_2)}{dx} - \frac{df(c_1)}{dx}\right).$$

By the mean value theorem,

$$E_0 = \frac{1}{8}\Delta x^2(c_2 - c_1)\frac{d^2f(c_3)}{dx^2} \leq \frac{1}{8}\Delta x^3 \frac{d^2f(c_3)}{dx^2} = O(\Delta x^3),$$

where c_1, c_2, and c_3 are between a and b and $b - a = \Delta x$. Then Equation (10.26) becomes

$$\int_{x_0-\Delta x/2}^{x_0+\Delta x/2} f(x)\,dx = \Delta x f(x_0) + O(\Delta x^3). \tag{10.27}$$

Note that after division by Δx, this result proves Equation (9.3).

As seen in any elementary calculus book, a large integration can be decomposed into a number of smaller integrations as follows:

$$\int_a^b f(x)\,dx = \int_a^{y_1} f(x)\,dx + \int_{y_1}^{y_2} f(x)\,dx + \cdots + \int_{y_{N-2}}^{y_{N-1}} f(x)\,dx + \int_{y_{N-1}}^{b} f(x)\,dx, \tag{10.28}$$

where the limits of integration y_i may or may not equal the sample points x_i. Then each of the small integrations can be approximated by Equation (10.26). This is known as a *Riemann sum.*

10.2.2 *Linear Approximations*

The line passing through any two samples $(x_0, f(x_0))$ and $(x_1, f(x_1))$ is as follows:

$$f(x) \approx p_1(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0). \tag{10.29}$$

Then

$$\int_a^b f(x)\,dx \approx \int_a^b p_1(x)\,dx = \int_a^b \left(f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0)\right)dx$$

or

$$\int_a^b f(x)\,dx \approx f(x_0)(b - a) + \frac{1}{2}\frac{f(x_1) - f(x_0)}{x_1 - x_0}[(b - x_0)^2 - (a - x_0)^2]. \tag{10.30}$$

One can analyze the order of accuracy of this expression much as in the previous subsection; the details are omitted.

Suppose $a = x_0$, $b = x_1$, and $b - a = x_1 - x_0 = \Delta x$. Then

♦ $$\int_{x_0}^{x_1} f(x)\,dx = \frac{\Delta x}{2}(f(x_1) + f(x_0)) + O(\Delta x^3), \tag{10.31}$$

which is called the *trapezoid rule*, since the area under the curve $f(x)$ is approximated by a trapezoid. A large integration can be decomposed into small integrations as in Equation (10.28). Then each of the small integrations can be approximated using the trapezoid rule, Equation (10.31), which is known as the *composite trapezoid rule*. By integrating quadratics, cubics, quartics, and so on, one obtains higher-order integration formulae such as Simpson's rule, Simpson's 3/8 rule, and Boole's rule; the details are omitted.

The properties of the integration formulae seen in this section are similar to the properties of the polynomial interpolations that birthed them. For example, as seen in Section 8.1.5, interpolation error is minimized by choosing the interpolation points to be the roots of Legendre or Chebyshev polynomials. Then integration error can also be minimized by choosing the samples to be the roots of Legendre or Chebyshev polynomials. In particular, if the samples are the roots of a Legendre polynomial, the integration is called a *Gauss–Legendre quadrature*. Similarly, if the samples are the roots of a Chebyshev polynomial, the integration is called a *Gauss–Chebyshev quadrature*. Tables 25.4–25.8 in Abramowitz and Stegun (1964) list weightings and nodes for Gauss–Legendre and Gauss–Chebyshev quadratures.

10.3 Runge–Kutta Methods for Solving Ordinary Differential Equations

Consider the following system of ordinary differential equations:

$$\frac{d\mathbf{u}}{dt} = \mathbf{R}(\mathbf{u}, t), \tag{10.32}$$

where, for future reference, the independent variable is now t instead of x, and the dependent variables are now $\mathbf{u}$ instead of $\mathbf{f}$. Using forward differences, as defined by Equation (10.8), we can write a very simple first-order accurate numerical approximation as

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = \mathbf{R}(\mathbf{u}^n, t^n)$$

or

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \mathbf{R}(\mathbf{u}^n, t^n), \tag{10.33}$$

where $t^{n+1} = t^n + \Delta t$, $\mathbf{u}^{n+1} \approx \mathbf{u}(t^{n+1})$, and $\mathbf{u}^n \approx \mathbf{u}(t^n)$. This is called the *forward-Euler approximation*.

The forward-Euler approximation can also be derived using numerical integration rather than numerical differentiation. To see this, integrate both sides of the ordinary differential equation to obtain the following integral equation:

$$u(t^{n+1}) - u(t^n) = \int_{t^n}^{t^{n+1}} \mathbf{R}(\mathbf{u}, t)\, dt. \tag{10.34}$$

Then apply Equation (10.26) to the right-hand side to obtain Equation (10.33). To save space, the derivations of the rest of the formulae in this section are omitted. However, rest assured that they can be obtained using numerical integration, numerical differentiation, or Taylor series.

Now consider the following second-order two-stage method:

$$\mathbf{u}^{(1)} = \mathbf{u}^n + \Delta t \mathbf{R}(\mathbf{u}^n, t^n), \tag{10.35a}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \frac{1}{2}\Delta t \mathbf{R}(\mathbf{u}^n, t^n) + \frac{1}{2}\Delta t \mathbf{R}\left(\mathbf{u}^{(1)}, t^{n+1}\right), \tag{10.35b}$$

which is called the *improved Euler approximation.* Notice that the initial guess found in the first stage is the forward-Euler approximation; the first stage is sometimes called a *predictor.* The second stage refines the initial guess, improving the order of accuracy by one; the second stage is sometimes called a *corrector.* As an alternative to the improved Euler method, consider the following second-order accurate two-step procedure:

$$\mathbf{u}^{(1)} = \mathbf{u}^n + \frac{1}{2}\Delta t \mathbf{R}(\mathbf{u}^n, t^n), \tag{10.36a}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \mathbf{R}\left(\mathbf{u}^{(1)}, t^n + \frac{1}{2}\Delta t\right). \tag{10.36b}$$

This is known as the *modified Euler approximation.*

In general, consider any two-stage method of the following form:

$$\mathbf{u}^{(1)} = \mathbf{u}^n + \Delta t a_{10} \mathbf{R}(\mathbf{u}^n, t^n + c_1 \Delta t), \tag{10.37a}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t (a_{20} \mathbf{R}(\mathbf{u}^n, t^n + c_1 \Delta t) + a_{21} \Delta t \mathbf{R}(\mathbf{u}^{(1)}, t^n + c_2 \Delta t)). \tag{10.37b}$$

It can be shown that this two-stage method is second-order accurate if

$$c_1 = 0, \tag{10.38}$$

$$c_2 = a_{10}, \tag{10.39}$$

$$a_{20} + a_{21} = 1, \tag{10.40}$$

$$a_{21} a_{10} = \frac{1}{2}. \tag{10.41}$$

There are four equations in five unknowns, which leaves one degree of freedom. For example, the improved Euler method has $a_{10} = 1$, $a_{20} = 1/2$, $a_{21} = 1/2$, $c_1 = 0$, and $c_2 = 1$. For another example, the modified Euler method has $a_{10} = 1/2$, $a_{20} = 0$, $a_{21} = 1$, $c_1 = 0$, and $c_2 = 1/2$.

If Equation (10.40) holds, then Equation (10.37) can be written as

$$\mathbf{u}^{n+1} = a_{20}\mathbf{y}_1 + (1 - a_{20})\mathbf{y}_2,$$

where

$$\mathbf{y}_1 = \mathbf{u}^n + \Delta t \mathbf{R}(\mathbf{u}^n, t^n + c_1 \Delta t), \tag{10.42}$$

$$\mathbf{y}_2 = \mathbf{u}^n + \Delta t \mathbf{R}\left(\mathbf{u}^{(1)}, t^n + c_2 \Delta t\right). \tag{10.43}$$

Thus, assuming $0 \leq a_{20} \leq 1$, $\mathbf{u}^{n+1}$ is a *convex linear combination* of two guesses $\mathbf{y}_1$ and $\mathbf{y}_2$, where both guesses look like the forward-Euler method, the only difference being the starting times and the starting values for $\mathbf{u}$. This is now the third time we have seen convex linear combinations (the last time was in Section 10.1.2).

Let us generalize the preceding approach to any number of stages. An *explicit m-stage Runge–Kutta method* is defined as follows:

♦
$$\begin{aligned}
\mathbf{u}^{(1)} &= \mathbf{u}^{(0)} + \Delta t a_{10}\mathbf{R}^{(0)},\\
\mathbf{u}^{(2)} &= \mathbf{u}^{(0)} + \Delta t\left(a_{20}\mathbf{R}^{(0)} + a_{21}\mathbf{R}^{(1)}\right),\\
\mathbf{u}^{(3)} &= \mathbf{u}^{(0)} + \Delta t\left(a_{30}\mathbf{R}^{(0)} + a_{31}\mathbf{R}^{(1)} + a_{32}\mathbf{R}^{(2)}\right),\\
&\vdots\\
\mathbf{u}^{(m)} &= \mathbf{u}^{(0)} + \Delta t\left(a_{m0}\mathbf{R}^{(0)} + a_{m1}\mathbf{R}^{(1)} + \cdots + a_{m,m-1}\mathbf{R}^{(m-1)}\right),
\end{aligned} \tag{10.44}$$

where

$$\begin{aligned}
\mathbf{u}^{(0)} &= \mathbf{u}^{n},\\
\mathbf{u}^{(m)} &= \mathbf{u}^{n+1},
\end{aligned}$$

and where

$$\mathbf{R}^{(i)} = R\left(\mathbf{u}^{(i)}, t^n + c_{i+1}\Delta t\right).$$

The coefficients a_{ij} and c_i are called the *Runge–Kutta coefficients*. The Runge–Kutta coefficients typically satisfy

$$a_{i0} + a_{i1} + \cdots + a_{i,i-1} = c_{i+1} \tag{10.45}$$

for $i = 1, \ldots, m-1$ and

$$a_{m0} + a_{m1} + \cdots + a_{m,m-1} = 1. \tag{10.46}$$

The Runge–Kutta coefficients also typically satisfy $0 \le a_{ij} \le 1$ and $0 \le c_i \le 1$. In the rare cases where the Runge–Kutta coefficients are negative, they usually at least satisfy $|a_{ij}| \le 1$ and $|c_i| \le 1$. Finally, the Runge–Kutta coefficient c_1 typically satisfies $c_1 = 0$.

If Equation (10.46) holds, the last stage in the Runge–Kutta method can be written as

$$\mathbf{u}^{n+1} = a_{m0}\mathbf{y}_1 + a_{m1}\mathbf{y}_2 + \cdots + a_{m,m-1}\mathbf{y}_m, \tag{10.47}$$

where

$$\mathbf{y}_i = \mathbf{u}^n + \Delta t\mathbf{R}^{(i)}. \tag{10.48}$$

If $0 \le a_{mj} \le 1$ and Equation (10.46) holds, then $\mathbf{u}^{n+1}$ is a *convex linear combination* of m guesses $\mathbf{y}_1, \ldots, \mathbf{y}_m$, where all the guesses look like the forward-Euler method, the only difference being the starting times and the starting values for $\mathbf{u}$.

As the best interpretation, the Runge–Kutta method is nothing more than a simple general form. In other words, a Runge–Kutta method is any method that can be written in the form of Equation (10.44), where each guess is a linear combination of previous guesses, and the final guess is a convex linear combination of all m guesses. Many useful multistep methods can be written in Runge–Kutta form. Unfortunately, many useful methods cannot be written in Runge–Kutta form, whereas many useless methods can be.

Example 10.2 A three-stage Runge–Kutta method is third-order accurate if

$$
\begin{aligned}
&c_1 = 0,\\
&c_2 = a_{10},\\
&c_3 = a_{20} + a_{21},\\
&a_{30} + a_{31} + a_{32} = 1,\\
&a_{31}c_2 + a_{32}c_3 = \frac{1}{2},\\
&a_{31}c_2^2 + a_{32}c_3^2 = \frac{1}{3},\\
&a_{32}a_{21}c_2 = \frac{1}{6}.
\end{aligned}
$$

This is proven in Lambert (1991). Lambert also proves similar results for four- and five-stage Runge–Kutta methods.

To compactly specify a Runge–Kutta method, the Runge–Kutta coefficients are often arranged into an array as follows:

$$
\begin{array}{c|cccc}
c_1 & & & & \\
c_2 & a_{10} & & & \\
c_3 & a_{20} & a_{21} & & \\
\vdots & \vdots & \vdots & & \\
c_m & a_{m-1,0} & a_{m-1,1} & \cdots & a_{m-1,m-2} \\
\hline
 & a_{m0} & a_{m1} & a_{m,m-2} & a_{m,m-1}
\end{array}
\tag{10.49}
$$

which is called a *Butcher array*.

Example 10.3 The Butcher array for the improved Euler method is as follows:

$$
\begin{array}{c|cc}
0 & & \\
1 & 1 & \\
\hline
 & 1/2 & 1/2
\end{array}
$$

Example 10.4 Consider the following fourth-order-accurate four-stage Runge–Kutta method:

$$
\begin{aligned}
\mathbf{u}^{(1)} &= \mathbf{u}^{(0)} + \frac{1}{2}\Delta t\mathbf{R}\left(\mathbf{u}^{(0)}, t^n\right),\\
\mathbf{u}^{(2)} &= \mathbf{u}^{(0)} + \frac{1}{2}\Delta t\mathbf{R}\left(\mathbf{u}^{(1)}, t^n + \frac{1}{2}\Delta t\right),\\
\mathbf{u}^{(3)} &= \mathbf{u}^{(0)} + \Delta t\mathbf{R}\left(\mathbf{u}^{(2)}, t^n + \frac{1}{2}\Delta t\right),\\
\mathbf{u}^{(4)} &= \mathbf{u}^{(0)} + \frac{1}{6}\Delta t\mathbf{R}\left(\mathbf{u}^{(0)}, t^n\right) + \frac{1}{3}\Delta t\mathbf{R}\left(\mathbf{u}^{(1)}, t^n + \frac{1}{2}\Delta t\right)\\
&\quad + \frac{1}{3}\Delta t\mathbf{R}\left(\mathbf{u}^{(2)}, t^n + \frac{1}{2}\Delta t\right) + \frac{1}{6}\Delta t\mathbf{R}\left(\mathbf{u}^{(2)}, t^{n+1}\right).
\end{aligned}
$$

The Butcher array for this method is as follows:

0				
1/2	1/2			
1/2	0	1/2		
1	0	0	1	
	1/6	1/3	1/3	1/6

This Runge–Kutta method is so popular that it is sometimes called *the* Runge–Kutta method. Thus if someone refers to the Runge–Kutta method without any specifics, they often mean this particular Runge–Kutta method. Despite its popularity, this Runge–Kutta method does not have any unique advantages over other four-stage Runge–Kutta methods.

Since the Runge–Kutta method is just a general form, its success depends entirely on the choice of the Runge–Kutta coefficients. As shown in Chapter 5 of Lambert (1991), the order of accuracy of an m-stage Runge–Kutta method is governed by the following results:

- Suppose $1 \leq m \leq 4$. Then an m-stage Runge–Kutta method can have mth-order accuracy, at best. Vector Runge–Kutta methods always have the same order of accuracy as scalar Runge–Kutta methods. In other words, Runge–Kutta methods yield the same order of accuracy for single ordinary differential equations as for systems of ordinary differential equations.
- Suppose $m \geq 5$. Then an m-stage Runge–Kutta method can have no more than mth-order accuracy but, in fact, the maximum order of accuracy may be less than m. For example, 5-stage Runge–Kutta methods are fourth-order accurate, at best. Unfortunately, the maximum attainable order of accuracy for arbitrary m is unknown. Surprisingly, for $m \geq 5$, vector Runge–Kutta methods may have a lower order of accuracy than scalar Runge–Kutta methods. These results imply that the Runge–Kutta form is not entirely natural for $m \geq 5$. However, in computational gasdynamics, Runge–Kutta methods with $1 \leq m \leq 4$ are usually quite sufficient.

The preceding results speak to the order of accuracy of Runge–Kutta methods. However, methods with the same order of accuracy may have wildly different accuracies. For example, as seen later in Chapters 11, 15, and 16, instability can result in unbounded oscillations regardless of the order of accuracy. Thus, in addition to order of accuracy constraints, Runge–Kutta coefficients are often chosen to optimize stability. For example, see Section 5.12 of Lambert (1991) for a linear stability analysis.

This section has concerned only *explicit* Runge–Kutta methods. It is also possible to construct *implicit* Runge–Kutta methods as follows:

$$\mathbf{u}^{(i)} = \mathbf{u}^{(0)} + \Delta t\left(a_{i0}\mathbf{R}^{(0)} + a_{i1}\mathbf{R}^{(1)} + \cdots + a_{i,m-1}\mathbf{R}^{(m-1)}\right). \tag{10.50}$$

Then each $\mathbf{u}^{(i)}$ depends on itself as well as every other value $\mathbf{u}^{(j)}$. Whereas the Butcher arrays for explicit Runge–Kutta methods are triangular, the Butcher arrays for implicit Runge–Kutta methods are square. Implicit Runge–Kutta methods are discussed in Lambert (1991).

There are, of course, many other numerical methods for solving ordinary differential equations beside Runge–Kutta methods; once again, see Lambert (1991). However, Runge–Kutta methods are far and away the most popular choice in computational gasdynamics.

In fact, some of the most famous and efficient numerical methods in computational gasdynamics rely heavily on the Runge–Kutta method. For two prominent examples of the use of Runge–Kutta methods in computational gasdynamics, see Sections 21.4 and 22.3.

References

Abramowitz, M., and Stegun, A. 1964. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, New York: Dover, Chapter 25.

Lambert, J. D. 1991. *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*, Chichester: Wiley, Chapter 5.

Mathews, J. H. 1992. *Numerical Methods for Mathematics, Science, and Engineering*, 2nd ed., Englewood Cliffs, NJ: Prentice-Hall, Chapters 6, 7, and 9.

Problems

10.1 Suppose you are given the samples $(x_0, f(x_0))$, $(x_0+h, f(x_0+h))$, and $(x_0+3h, f(x_0+3h))$.
(a) Approximate $f'(x_0+h)$. What is the order of accuracy of the approximation?
(b) Approximate $f''(x_0+2h)$.
(c) Approximate $\int_{x_0}^{x_0+2h} f(x)dx$. Is this integration formula a Newton–Cotes quadrature? If so, is this integration formula a *closed* Newton–Cotes quadrature?

10.2 (a) Approximate $f'(x)$, $f''(x)$, and $f'''(x)$ given the samples $(-3, 0)$, $(-1, 50)$, $(0, 53.25)$, $(3, 174)$, and $(4, 341.25)$. The interpolation polynomial passing through these samples is $\frac{1}{4}x^4 + 3x^3 + \frac{3}{2}x^2 + 2x + 53.25$. What sort of accuracy would you expect from your approximation of $f''(7)$?
(b) Find an approximation to the primitive function $f(x) = \int_{-1}^{x} f(y)dy$.

10.3 Consider Example 8.10. Using the expressions for $p_N(x)$ for $N = 2, \ldots, 6$, approximate $f'(1/4)$ and $f''(1/4)$. Give a table listing the error of $f'(1/4)$ and $f''(1/4)$ as a function of N for $N = 2, \ldots, 6$.

10.4 (a) Prove Equation (10.17).
(b) Generalize Equation (10.17) to find an expression that works for nonconstant sample spacings.

10.5 Consider interpolation polynomials with Runge oscillations such as, for example, the interpolation polynomial for a square wave seen in Figure 8.5. Notice that, in all cases, the interpolation polynomial has the least error (zero error) at the sample points.
(a) Where does the derivative of the interpolation polynomial have the least error? The greatest error? Argue that the error of the derivative is out of phase with the error of the interpolation polynomial, that is, the error of the derivative is roughly the greatest where the error of the interpolation polynomial is the least, and vice versa.
(b) If the lower limit of integration is any sample point, where does the primitive function of the interpolation polynomial have the least error? The greatest error? Argue that the error of the primitive function has maxima and minima roughly at sample points, in an odd–even fashion. Argue that the primitive function has less error than the interpolation polynomial, due to cancellation of positive and negative error.

10.6 Consider Problem 9.7. Use your solutions for parts (a) though (c) to estimate the derivative $f'(x)$ and the primitive function $f(x) = \int_{-1}^{x} f(y)\,dy$. In each case, plot the true derivative versus the approximation based on interpolation, and the true primitive function versus the approximation based on interpolation, on the domain $[-1, 1]$. Also, in each case, find the

error of the derivative and integral approximation on the domain $[-1, 1]$ in the 1-norm, the 2-norm, and the ∞-norm.

10.7 Consider the following ordinary differential equation:

$$\frac{du}{dt} = t - u,$$

where $u(0) = 0.9$.

(a) Approximate $u(5)$ using the forward-Euler method for $\Delta t = 0.5$ and $\Delta t = 0.125$.
(b) Approximate $u(5)$ using the improved Euler method for $\Delta t = 0.5$ and $\Delta t = 0.125$.
(c) Approximate $u(5)$ using the modified Euler method for $\Delta t = 0.5$ and $\Delta t = 0.125$.
(d) Approximate $u(5)$ using the Runge–Kutta method given in Example 10.4 for $\Delta t = 0.5$ and $\Delta t = 0.125$.