CHAPTER 2

Numerical Integration & Differentiation

Contents

2.1 Integration . . . 3
2.2 Existence, Uniqueness, and Conditioning . . . 5
2.3 Numerical Quadrature . . . 5
2.3.1 Newton-Cotes Quadrature . . . 8
2.3.2 Clenshaw-Curtis Quadrature . . . 11
2.3.3 Gaussian Quadrature . . . 12
2.3.4 Progressive Gaussian Quadrature . . . 14
2.3.5 Composite Quadrature . . . 14
2.3.6 Adaptive Quadrature . . . 15
2.4 Other Integration Problems . . . 17
2.4.1 Tabular Data . . . 17
2.4.2 Improper Integrals . . . 17
2.4.3 Double Integrals . . . 18
2.4.4 Multiple Integrals . . . 19
2.5 Integral Equations . . . 19
2.6 Numerical Differentiation . . . 21
2.6.1 Finite Difference Approximations . . . 21
2.6.2 Automatic Differentiation . . . 23

2.1. Integration

In elementary geometry, one learns to calculate the area of simple shapes like rectangles, triangles, and circles, as well as the volume of basic solids such as cubes and spheres. A key motivation for the development of integral calculus was the need to determine areas and volumes of regions with irregular shapes, a crucial problem in classical mechanics. One of the foundational principles in this field is that a rigid body can often be approximated as a point mass located at its centroid, and a distributed force can be treated as an equivalent concentrated force applied at a specific point.
For example, consider a beam of length L bearing a distributed load, as illustrated in Fig. 2.1. If $\rho(x)$ denotes the density of the load as a function of the horizontal coordinate x, then the total load W on the beam corresponds to the area under the curve $\rho(x)$, given by the integral:

$$W = \int_0^L \rho(x)\, dx$$

Furthermore, the location of the equivalent concentrated load, known as the centroid C of the area under the curve, has a horizontal coordinate $\bar{x}$ given by:

$$\bar{x} = \frac{1}{W} \int_0^L x\rho(x)\,dx$$

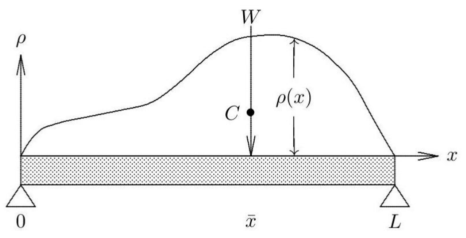

Figure 2.1: Concentrated load equivalent to distributed load on beam

Ancient mathematicians, such as Archimedes, devised methods to approximate the area of irregular regions by tiling them with small squares of known size and counting the number of squares that fit within the region. While these techniques lacked a rigorous theoretical foundation, such as the modern concept of a limit, they are conceptually similar to numerical methods used today for approximating integrals.
For a function $f : \mathbb{R} \to \mathbb{R}$ on an interval $[a, b]$, the modern definition of an integral,

$$I(f) = \int_a^b f(x)\,dx$$

is built upon Riemann sums:

$$R_n = \sum_{i=1}^{n} (x_{i+1} - x_i)\, f(\xi_i)$$

where $a = x_1 < x_2 < \cdots < x_n < x_{n+1} = b$ and $\xi_i \in [x_i, x_{i+1}]$ for $i = 1, \ldots, n$. Define

$$h_n = \max\{x_{i+1} - x_i : i = 1, \ldots, n-1\}.$$

If, as $h_n \to 0$, R_n converges to a finite value R for any choice of x_i and ξ_i, then f is Riemann integrable on $[a, b]$, and the value of the integral is R. This definition naturally leads to a numerical method: use a finite Riemann sum with a sufficiently large n to achieve the desired accuracy. However, unless x_i and ξ_i are chosen carefully, many more evaluations of f may be needed than necessary. In this lecture, we will explore efficient methods that provide high accuracy at a relatively low cost, measured by the number of function evaluations required.
Although there are more advanced concepts of integration, such as Lebesgue integration, which serve as powerful theoretical tools, their nonconstructive nature makes them unsuitable for numerical computation. Therefore, we will focus on Riemann integrals. Integration's utility extends well beyond the initial geometric and mechanical applications and includes:

- Integral transforms like the Laplace, Fourier, and Hankel transforms
- Special functions in applied mathematics and mathematical physics, many of which have integral representations, including the gamma, beta, Bessel, and error functions, as well as Fresnel and elliptic integrals
- Finite element and boundary element methods for solving partial differential equations
- Integral equations and variational methods
- Probability and statistics, where fundamental concepts such as probability distributions, moments, and expectations are defined using integrals
- Classical and quantum physics, where many systems' potential or free energy is represented as an integral, such as the electrostatic potential generated by a charge distribution

2.2. Existence, Uniqueness, and Conditioning

If $f : \mathbb{R} \to \mathbb{R}$ is bounded and continuous almost everywhere (i.e., continuous except on a set of measure zero, which is a set that can be covered by the union of a countable number of open intervals with arbitrarily small total length) on an interval $[a, b]$, then the Riemann integral $I(f)$ exists. This sufficient condition is also necessary, meaning that unbounded functions are not properly Riemann integrable. In practical terms, integrable functions are bounded functions with at most a finite number of discontinuities within the integration interval. Since all Riemann sums defining the Riemann integral of a given function on a given interval must converge to the same limit, the uniqueness of the Riemann integral is inherent in the definition.
The conditioning of an integration problem refers to how sensitive the result is to small changes in the input data, which in this case are the integrand function f and the integration interval $[a, b]$. To quantify this sensitivity, we use a norm for functions defined on a given interval. Analogous to the ∞-norm for finite-dimensional vectors, for a function f on a closed interval $[a, b]$, we define

$$\|f\|_\infty = \max_{x\in[a,b]} |f(x)|$$

Now, let $\hat{f}$ be a perturbed version of the integrand f. We then have

$$\begin{aligned} |I(\hat{f}) - I(f)| &= \left| \int_a^b \hat{f}(x)\,dx - \int_a^b f(x)\,dx \right| \\ &\le \int_a^b |\hat{f}(x) - f(x)|\,dx \\ &\le (b-a)\|\hat{f} - f\|_\infty \end{aligned}$$

This shows that the absolute condition number of the integration problem is at most $b - a$. This bound is realized when $\hat{f}(x) = f(x) + c$, where c is a positive constant, making the absolute condition number equal to $b - a$. This is a satisfactory result, as the effect of a perturbation in the integrand is at most proportional to the length of the integration interval. For the relative condition number, the preceding inequality implies

$$\frac{|I(\hat{f}) - I(f)|/|I(f)|}{\|\hat{f} - f\|_\infty/\|f\|_\infty} \le \frac{(b-a)\|f\|_\infty}{|I(f)|}$$

This quantity is always greater than or equal to 1 and can become arbitrarily large for some integrands because $|I(f)|$ may be small (or even zero) while $\|f\|_\infty$ remains large. In such cases, the absolute condition number is more appropriate. Nevertheless, the relatively small size of the condition number indicates that integration problems are generally well-conditioned with respect to perturbations in the integrand. This result is intuitive, as integration is an averaging process that tends to smooth out small changes in the integrand.
Now consider a perturbation $\hat{b}$ of b, with $\hat{b} > b$ (a similar analysis applies if $\hat{b} < b$, or for analogous perturbations of a). We have

$$\left| \int_a^{\hat{b}} f(x)\,dx - \int_a^b f(x)\,dx \right| = \left| \int_b^{\hat{b}} f(x)\,dx \right| \le (\hat{b} - b) \max_{x\in[b,\hat{b}]} |f(x)|$$

Thus, the absolute condition number with respect to perturbations in the limits of integration is usually moderate, but it can become very large if the integrand has a nearby singularity.

2.3. Numerical Quadrature

In elementary calculus one learns to evaluate a definite integral

$$I(f) = \int_a^b f(x)dx$$

analytically by finding an antiderivative F of the integrand function f (i.e., a function F such that $F'(x) = f(x)$), and then using the Fundamental Theorem of Calculus to evaluate

$$I(f) = F(b) - F(a)$$

Unfortunately, some integrals cannot be evaluated in such a closed form (e.g., $f(x) = \exp\left(-x^2\right)$), and many integrals arising in practice are too complicated to evaluate analytically even if this were possible in principle. Thus, we are often forced to employ numerical methods to evaluate definite integrals approximately.

The numerical approximation of definite integrals is known as numerical quadrature. This name, which derives from ancient methods for approximating areas of irregular or curved figures by tiling them with small squares, helps distinguish this topic from the numerical integration of differential equations, which we will consider later in this notes. In approximating integrals, we will take our cue from the Riemann sums that define the integral: the integral will be approximated by a weighted sum of integrand values at a finite number of sample points in the interval of integration. Specifically, the integral $I(f)$ is approximated by an n-point quadrature rule, which has the form

$$Q_n(f)=\sum_{i=1}^{n} w_i f\left(x_i\right)$$

where $a \leq x_1<x_2<\cdots<x_n \leq b$. The points x_i at which the integrand f is evaluated are called nodes or abscissas, and the multipliers w_i are called weights or coefficients. A quadrature rule is said to be open if $a<x_1$ and $x_n<b$, and closed if $a=x_1$ and $x_n=b$. Our main objective will be to choose the nodes and weights so that we obtain a desired level of accuracy at a reasonable computational cost in terms of the number of integrand evaluations required.

Quadrature rules can be derived using polynomial interpolation. In effect, the integrand function f is evaluated at the points $x_i, i=1, \ldots, n$, the polynomial of degree $n-1$ that interpolates the function values at those points is determined, and the integral of the interpolant is then taken as an approximation to the integral of the original function. In practice, the interpolating polynomial is not determined explicitly each time a particular integral is to be evaluated. Instead, polynomial interpolation is used to determine the nodes and weights for a given quadrature rule, which can then be used in approximating the integral of any function over the interval. In particular, if Lagrange interpolation is used, then the weights are given by the integrals of the corresponding Lagrange basis functions for the given set of points $x_i, i=1, \ldots, n$,

$$w_i=\int_a^b \ell_i(x) d x, \quad i=1, \ldots, n$$

and these are the same for any integrand. For obvious reasons, the resulting quadrature rule is said to be interpolatory.

An alternative method for deriving interpolatory quadrature rules, called the method of undetermined coefficients, is to choose the weights so that the rule integrates the first n polynomial basis functions exactly, which results in a system of n equations in n unknowns. With the monomial basis, for example, this strategy results in the system of moment equations

$$\begin{aligned}
w_1 \cdot 1+w_2 \cdot 1+\cdots+w_n \cdot 1 &=\int_a^b 1 d x=b-a \\
w_1 \cdot x_1+w_2 \cdot x_2+\cdots+w_n \cdot x_n &=\int_a^b x d x=\left(b^2-a^2\right) / 2 \\
&\vdots \\
w_1 \cdot x_1^{n-1}+w_2 \cdot x_2^{n-1}+\cdots+w_n \cdot x_n^{n-1} &=\int_a^b x^{n-1} d x=\left(b^n-a^n\right) / n
\end{aligned}$$

Writing this system in matrix form, we have

$$\left[\begin{array}{cccc}
1 & 1 & \cdots & 1 \\
x_1 & x_2 & \cdots & x_n \\
\vdots & \vdots & \ddots & \vdots \\
x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1}
\end{array}\right]\left[\begin{array}{c}
w_1 \\
w_2 \\
\vdots \\
w_n
\end{array}\right]=\left[\begin{array}{c}
b-a \\
\left(b^2-a^2\right) / 2 \\
\vdots \\
\left(b^n-a^n\right) / n
\end{array}\right]$$

The matrix of this system is a Vandermonde matrix, which is nonsingular because the nodes x_i are assumed to be distinct. The unique solution of this system yields the weights $w_1, \ldots, w_n$, which are the same as those given by integrating the Lagrange basis functions.

Example 1: Method of Undetermined Coefficients

We illustrate the method of undetermined coefficients by deriving a three-point quadrature rule

$$Q_3(f)=w_1 f\left(x_1\right)+w_2 f\left(x_2\right)+w_3 f\left(x_3\right)$$

for the interval $[a, b]$ using the monomial basis. We take the two endpoints and midpoint as the three nodes, i.e., $x_1 = a, x_2 = (a+b)/2$, and $x_3 = b$. Written in matrix form, the moment equations give the Vandermonde system

$$\begin{bmatrix} 1 & 1 & 1 \\ a & (a+b)/2 & b \\ a^2 & ((a+b)/2)^2 & b^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} b-a \\ \left(b^2-a^2\right)/2 \\ \left(b^3-a^3\right)/3 \end{bmatrix}$$

Solving this system by Gaussian elimination, we obtain the weights

$$w_1 = (b-a)/6, \quad w_2 = 2(b-a)/3, \quad w_3 = (b-a)/6$$

The resulting quadrature rule is known as Simpson's rule.

By construction, an n-point interpolatory quadrature rule integrates each of the first $n-1$ monomial basis functions exactly, and hence by linearity it integrates any polynomial of degree at most $n-1$ exactly. A quadrature rule is said to be of degree d if it is exact (i.e., the error is zero) for every polynomial of degree d but is not exact for some polynomial of degree $d+1$. As we have just seen, an n-point interpolatory quadrature rule is of degree at least $n-1$. Conversely, any quadrature rule with degree at least $n-1$ must be interpolatory, since it satisfies the moment equations.

The significance of the degree is that it conveniently characterizes the accuracy of a given rule. If Q_n is an interpolatory quadrature rule, and p_{n-1} is the polynomial of degree at most $n-1$ interpolating a sufficiently smooth integrand f at the nodes $x_1, \ldots, x_n$, then the error bound for the polynomial interpolant yields the following rough error bound for the approximate integral:

$$\begin{aligned} |I(f) - Q_n(f)| &= |I(f) - I(p_{n-1})| = |I(f - p_{n-1})| \\ &\leq (b-a) \|f - p_{n-1}\|_\infty \\ &\leq \frac{b-a}{4n} h^n \left\| f^{(n)} \right\|_\infty \\ &\leq \frac{1}{4} h^{n+1} \left\| f^{(n)} \right\|_\infty \end{aligned}$$

where $h = \max\{x_{i+1} - x_i : i = 1, \ldots, n-1\}$.
We will be able to make sharper error estimates when we consider specific quadrature rules, but the preceding general bound already indicates that we can obtain higher accuracy by taking n larger, or h smaller, or both. Indeed, the bound shows that $Q_n(f) \to I(f)$ as $n \to \infty$, as well as the minimum convergence rate we can expect, provided $f^{(n)}$ remains well behaved. In the absence of the latter assumption, however, convergence may not obtain or the rate may be arbitrarily slow. That some regularity assumptions on the integrand function are necessary to obtain satisfactory results should not be surprising, as otherwise any method based on sampling the integrand at only a finite number of points can be completely wrong. When the number of sample points is increased, say from n to m, an important factor affecting efficiency is whether the n function values already computed can be reused in the new rule, so that only $m-n$ new function values need be computed. A sequence of quadrature rules is said to be progressive if the nodes of Q_{n_1} are a subset of those of Q_{n_2} for $n_2 > n_1$.

Instead of (or in addition to) increasing the number of points (and hence the degree), the preceding bound also suggests that the error can be reduced by subdividing the interval of integration into smaller subintervals and applying the quadrature rule separately in each, since this will reduce h. This approach, which is equivalent to using piecewise polynomial interpolation on the original interval, leads to composite (or compound) quadrature rules, which we will consider in Section §2.3.5. For now, we will focus on simple quadrature rules, in which a single rule is applied over the entire given interval.

The error bound just given, and others we will see later based on Taylor series expansions, depend on higher derivatives of the integrand function for which we may have no convenient bound. In practice, therefore, error estimates for quadrature rules are usually based on the difference between the results obtained when two different rules are used to approximate the same integral. The second rule may be obtained from the base rule either by increasing the number of points (and hence the degree) or by subdividing the interval of integration and applying the base rule in each subinterval. We will see examples of both of these approaches. To save on function evaluations, it is highly desirable for the two rules to be progressive.

In addition to its accuracy, we must also be concerned with the stability of a quadrature rule. If $\hat{f}$ is a perturbation to the integrand function f, then we have

$$\begin{aligned}\left|Q_n(\hat{f}) - Q_n(f)\right| &= \left|Q_n(\hat{f} - f)\right| \\ &= \left|\sum_{i=1}^{n} w_i \left(\hat{f}(x_i) - f(x_i)\right)\right| \\ &\leq \sum_{i=1}^{n} \left(|w_i| \cdot \left|\hat{f}(x_i) - f(x_i)\right|\right) \\ &\leq \left(\sum_{i=1}^{n} |w_i|\right) \|\hat{f} - f\|_\infty\end{aligned}$$

which says that the absolute condition number of the quadrature rule is at most $\sum_{i=1}^n |w_i|$. The preceding bound is attainable for an appropriately chosen perturbation, so the absolute condition number is in fact equal to $\sum_{i=1}^n |w_i|$. Recall from the first moment equation that for an interpolatory quadrature rule we have $\sum_{i=1}^n w_i = b-a$. If the weights are all nonnegative, then the absolute condition number of the quadrature rule is $b-a$, which the same as that of the underlying integration problem, and thus the quadrature rule is stable. If some of the weights are negative, however, then the absolute condition number can be much larger and the quadrature rule can be unstable.

2.3.1. Newton-Cotes Quadrature

The simplest placement of nodes for an interpolatory quadrature rule is to choose equally spaced points in the interval $[a, b]$, which is the defining property of NewtonCotes quadrature. An n-point open Newton-Cotes rule has nodes

$$x_i = a + i(b-a)/(n+1), \quad i = 1, \ldots, n$$

and an n-point closed Newton-Cotes rule has nodes

$$x_i = a + (i-1)(b-a)/(n-1), \quad i = 1, \ldots, n$$

The following are some of the simplest and best known examples of Newton-Cotes quadrature rules:

- Interpolating the function value at the midpoint of the interval by a polynomial of degree zero (i.e., a constant) gives the one-point open Newton-Cotes rule known as the midpoint rule:

$$M(f) = (b-a) f\left(\frac{a+b}{2}\right)$$

- Interpolating the function values at the two endpoints of the interval by a polynomial of degree one (i.e., a straight line) gives the two-point closed Newton-Cotes rule known as the trapezoid rule:

$$T(f) = \frac{b-a}{2}(f(a) + f(b))$$

- Interpolating the function values at the two endpoints and the midpoint by a polynomial of degree two (i.e., a quadratic) gives the three-point closed NewtonCotes rule known as Simpson's rule:

$$S(f) = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)$$

which we derived in Example 1.

Example 2: Newton-Cotes Quadrature

To illustrate the application of Newton-Cotes quadrature rules, we approximate the integral

$$I(f) = \int_0^1 e^{-x^2} dx$$

using each of the three Newton-Cotes quadrature rules just given.

$$
\begin{aligned}
M(f) &= (1-0)\exp(-0.25) \approx 0.778801 \\
T(f) &= \frac{1}{2}(\exp(0)+\exp(-1)) \approx 0.683940 \\
S(f) &= \frac{1}{6}(\exp(0)+4\exp(-0.25)+\exp(-1)) \approx 0.747180
\end{aligned}
$$

The integrand and the interpolating polynomial for each rule are shown in Fig. 2.2. The correctly rounded result for this problem is 0.746824. It is somewhat surprising to see that the magnitude of the error from the trapezoid rule (0.062884) is about twice that from the midpoint rule (0.031977), and that Simpson's rule, with an error of only 0.000356, seems remarkably accurate considering the size of the interval over which it is applied. We will soon see explanations for these phenomena.

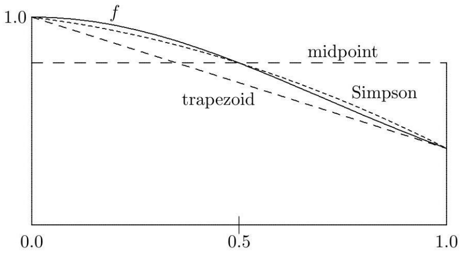

Figure 2.2: Integration of $f(x) = e^{-x^2}$ by Newton-Cotes quadrature rules

The error in the midpoint quadrature rule can be estimated using a Taylor series expansion about the midpoint $m = (a+b)/2$ of the interval $[a, b]$:

$$
\begin{aligned}
f(x) =& f(m) + f'(m)(x-m) + \frac{f''(m)}{2}(x-m)^2 \\
&+ \frac{f^{(3)}(m)}{6}(x-m)^3 + \frac{f^{(4)}(m)}{24}(x-m)^4 + \cdots
\end{aligned}
$$

Integrating this expression from a to b, the odd-order terms drop out, yielding

$$
\begin{aligned}
I(f) &= f(m)(b-a) + \frac{f''(m)}{24}(b-a)^3 + \frac{f^{(4)}(m)}{1920}(b-a)^5 + \cdots \\
&= M(f) + E(f) + F(f) + \cdots
\end{aligned}
$$

where $E(f)$ and $F(f)$ represent the first two terms in the error expansion for the midpoint rule.
To derive a comparable error expansion for the trapezoid quadrature rule, we substitute $x = a$ and $x = b$ into the Taylor series, add the two resulting series together, observe once again that the odd-order terms drop out, solve for $f(m)$, and substitute into the midpoint expansion to obtain

$$I(f) = T(f) - 2E(f) - 4F(f) - \cdots$$

Note that

$$T(f) - M(f) = 3E(f) + 5F(f) + \cdots$$

and hence the difference between the two quadrature rules provides an estimate for the dominant term in their error expansions,

$$E(f) \approx \frac{T(f) - M(f)}{3}$$

provided that the length of the interval is sufficiently small that $(b-a)^5 \ll (b-a)^3$, and the integrand f is such that $f^{(4)}$ is well-behaved. Under these assumptions, we may draw several conclusions from the preceding derivations:

- The midpoint rule is about twice as accurate as the trapezoid rule (as we saw in Example 2), despite being based on a polynomial interpolant of degree one less.
- The difference between the midpoint rule and the trapezoid rule can be used to estimate the error in either of them.

- Halving the length of the interval decreases the error in either rule by a factor of about $1/8$.

An appropriately weighted combination of the midpoint and trapezoid rules eliminates the leading term, $E(f)$, from the error expansion,

$$\begin{aligned} I(f) &= \frac{2}{3}M(f) + \frac{1}{3}T(f) - \frac{2}{3}F(f) + \cdots \\ &= S(f) - \frac{2}{3}F(f) + \cdots \end{aligned}$$

which provides an alternative derivation for Simpson's rule as well as an expression for its dominant error term.

Example 3: Error Estimation

We illustrate these error estimates by computing the approximate value for the integral $\int_0^1 x^2 dx$. Using the midpoint rule, we obtain

$$M(f) = (1-0)\left(\frac{1}{2}\right)^2 = \frac{1}{4}$$

and using the trapezoid rule we obtain

$$T(f) = \frac{1-0}{2}\left(0^2 + 1^2\right) = \frac{1}{2}$$

Thus, we have the estimate

$$E(f) \approx \frac{T(f) - M(f)}{3} = \frac{1/4}{3} = \frac{1}{12}$$

We conclude that the error in $M(f)$ is about $\frac{1}{12}$, and the error in $T(f)$ is about $-\frac{1}{6}$. In addition, we can now compute the approximate value given by Simpson's rule for this integral,

$$S(f) = \frac{2}{3}M(f) + \frac{1}{3}T(f) = \frac{2}{3}\cdot\frac{1}{4} + \frac{1}{3}\cdot\frac{1}{2} = \frac{1}{3}$$

which is exact for this integral (as is to be expected since, by design, Simpson's rule is exact for quadratic polynomials). Thus, the error estimates for $M(f)$ and $T(f)$ are exact for this integrand (though this would not be true in general).

We observed previously that an n-point interpolatory quadrature has degree at least $n-1$. Thus, we would expect the midpoint rule to have degree zero, the trapezoid rule degree one, Simpson's rule degree two, and so on. We saw from the Taylor series expansion, however, that the error for the midpoint rule depends on the second and higher derivatives of the integrand, which vanish for linear as well as for constant polynomials. This implies that the midpoint rule integrates linear polynomials exactly, and hence its degree is one rather than zero. Similarly, the error for Simpson's rule depends on the fourth and higher derivatives, which vanish for cubic as well as quadratic polynomials, so that Simpson's rule is of degree three rather than two (which explains the surprisingly high accuracy obtained in Example 2).

In general, for any odd value of n, an n-point Newton-Cotes rule has degree one greater than that of the polynomial interpolant on which it is based. This phenomenon is due to cancellation of positive and negative errors, as illustrated for the midpoint and Simpson rules in Fig. 2.3, which, on the left, shows a linear polynomial and the constant function interpolating it at the midpoint and, on the right, a cubic and the quadratic interpolating it at the midpoint and endpoints. Integration of the linear polynomial by the midpoint rule yields two congruent triangles of equal area. The inclusion of one of the triangles compensates exactly for the omission of the other. A similar phenomenon occurs for the cubic polynomial, where the two shaded regions also have equal areas, so that the addition of one compensates for the subtraction of the other. Such cancellation does not occur, however, for an n-point Newton-Cotes rule if n is even. Thus, in general, an n point Newton-Cotes rule is of degree $n-1$ if n is even, but of degree n if n is odd.

Newton-Cotes quadrature rules are relatively easy to derive and to apply, but they have some serious drawbacks. Recall from the previous lecture that interpolation of a continuous function at equally spaced points by a high-degree polynomial may suffer from unwanted oscillation, and as the number of interpolation points grows, convergence to the underlying function is not guaranteed. The consequence for quadrature rules based on such interpolation is that every n-point Newton-Cotes rule with $n \geq 11$ has at least one negative weight. Indeed,

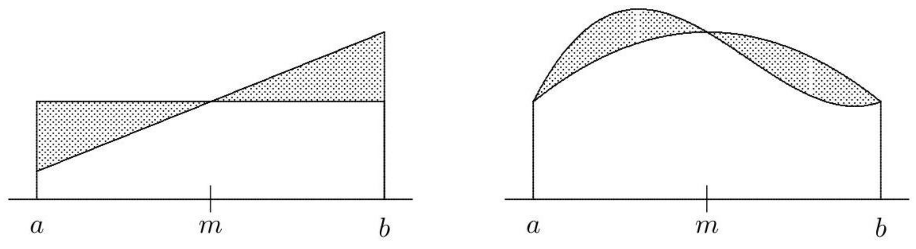

Figure 2.3: Cancellation of errors in midpoint (left) and Simpson (right) rules

it can be shown that $\sum_{i=1}^{n} |w_i| \to \infty$ as $n \to \infty$, which means that Newton-Cotes rules become arbitrarily ill-conditioned, and hence unstable, as the number of points grows. The presence of large positive and negative weights also means that the value of the integral is computed as a sum of large quantities of differing sign, and hence substantial cancellation is likely in finite-precision arithmetic.

For the reasons just given, we cannot expect to attain arbitrarily high accuracy on a given interval by using a Newton-Cotes rule with a large number of points. In practice, therefore, Newton-Cotes rules are usually restricted to a modest number of points, and if higher accuracy is required, then the interval is subdivided and the rule is applied in each subinterval separately (such strategies will be discussed in Section §2.3.5). In this regard, a positive feature of Newton-Cotes rules is that they are progressive, but on the other hand, Newton-Cotes rules do not have the highest possible degree (and hence accuracy) for the number of points used (and hence the number of function evaluations required), and we will soon see that we can do much better.

2.3.2. Clenshaw-Curtis Quadrature

We saw in the previous lecture that the Chebyshev points, suitably transformed from $[-1, 1]$ to a given interval of interest, have distinct advantages over equally spaced points for interpolating a continuous function by a polynomial. In particular, the maximum error over the interval is generally much smaller with the Chebyshev points, and the resulting interpolants converge to any sufficiently smooth function as the number of points $n \to \infty$. One might conjecture, therefore, that the Chebyshev points would also be a better choice of nodes for interpolatory quadrature rules, as was first suggested by Fejér, and indeed this expectation turns out to be the case.

With the Chebyshev points as nodes for a given n, the corresponding weights could be computed in the usual manner by integrating the Lagrange basis functions or by the method of undetermined coefficients. It can be shown that the resulting weights are always positive for any n, and that the resulting approximate values converge to the exact integral as $n \to \infty$. Thus, quadrature rules based on the Chebyshev points are extremely attractive in that they are always stable and significantly more accurate than Newton-Cotes rules for the same number of nodes.

Another attractive feature of such quadrature rules is that they can be implemented in a self-contained manner: as Clenshaw and Curtis first observed, the weights corresponding to the Chebyshev points need not be tabulated in advance or even computed explicitly at all. Instead, based on the trigonometric definition of the Chebyshev polynomials, they developed remarkably simple and efficient procedures for expressing the polynomial interpolant to the integrand function as a linear combination of Chebyshev polynomials, and then integrating the resulting approximation in closed form to arrive at an approximate value for the integral. Alternatively, techniques based on the fast Fourier transform can be used to implement this type of quadrature rule. These efficient, self-contained implementations of quadrature rules based on the Chebyshev points have become known as Clenshaw-Curtis quadrature.

Although the zeros and extrema of the Chebyshev polynomials have similar properties in terms of the stability and accuracy of the resulting quadrature rules, the Chebyshev extrema, unlike the Chebyshev zeros, have the advantage of yielding progressive quadrature rules. If the number of Chebyshev extrema is increased from n to $2n - 1$, then only $n - 1$ new points are added, so that only $n - 1$ new function evaluations are required. For this reason, use of the Chebyshev extrema is sometimes called practical Clenshaw-Curtis quadrature, whereas use of the Chebyshev zeros is called classical Clenshaw-Curtis quadrature or Fejér quadrature.

We have seen that Clenshaw-Curtis quadrature rules have many virtues: stability, accuracy, simplicity, self-containment, and progressiveness. Nevertheless, the degree of an n-point rule is only $n - 1$, which is well below the maximum possible. Next we will see that quadrature rules of maximum degree can be derived by exploiting all of the available degrees of freedom.

2.3.3. Gaussian Quadrature

In the quadrature rules we have seen thus far, the n nodes were prespecified and the n corresponding weights were then optimally chosen to maximize the degree of the resulting quadrature rule. With only n parameters free to be chosen, the resulting degree is generally $n-1$. If the locations of the nodes were also freely chosen, however, then there would be $2n$ free parameters, so that a degree of $2n-1$ should be achievable. In Gaussian quadrature, both the nodes and the weights are optimally chosen to maximize the degree of the resulting quadrature rule. In general, for each n there is a unique n-point Gaussian rule, and it is of degree $2n-1$. Gaussian quadrature rules therefore have the highest possible accuracy for the number of nodes used, but they are significantly more difficult to derive than Newton-Cotes rules. The nodes and weights can still be determined by the method of undetermined coefficients, but the resulting system of equations is nonlinear.

Example 4: Gaussian Quadrature Rule

To illustrate the derivation of a Gaussian quadrature rule, we will derive a two-point rule on the interval $[-1,1]$,

$$I(f)=\int_{-1}^{1} f(x)dx \approx w_1 f\left(x_1\right)+w_2 f\left(x_2\right)=G_2(f)$$

where the nodes x_1, x_2 as well as the weights w_1, w_2 are to be chosen to maximize the resulting degree. Requiring that the rule integrate the first four monomials exactly gives the system of four moment equations

$$\begin{aligned}
w_1+w_2 &= \int_{-1}^{1} 1dx=2 \\
w_1x_1+w_2x_2 &= \int_{-1}^{1} xdx=0 \\
w_1x_1^2+w_2x_2^2 &= \int_{-1}^{1} x^2dx=\frac{2}{3} \\
w_1x_1^3+w_2x_2^3 &= \int_{-1}^{1} x^3dx=0
\end{aligned}$$

One solution for this nonlinear system is given by

$$x_1=-1/\sqrt{3}, \quad x_2=1/\sqrt{3}, \quad w_1=1, \quad w_2=1$$

and the other solution is obtained by reversing the signs of x_1 and x_2. Thus, the two-point Gaussian quadrature rule has the form

$$G_2(f)=f(-1/\sqrt{3})+f(1/\sqrt{3})$$

and by construction it has degree three.

Alternatively, the nodes of a Gaussian quadrature rule can be obtained by using orthogonal polynomials. If p is a polynomial of degree n such that

$$\int_a^b p(x)x^k dx=0, \quad k=0,\ldots,n-1$$

and hence p is orthogonal to all polynomials on $[a,b]$ of degree less than n, then it is fairly easy to show that

1. The n zeros of p are real, simple, and lie in the open interval (a,b).
2. The n-point interpolatory quadrature rule on $[a,b]$ whose nodes are the zeros of p has degree $2n-1$; i.e., it is the unique n-point Gaussian rule.
   The Legendre polynomial P_n is just such a polynomial. For this reason, the resulting rule is often called a Gauss-Legendre quadrature rule. Of course, the zeros of the Legendre polynomial must still be computed, and then the corresponding weights for the quadrature rule can be determined in the usual way. This method also extends naturally to various other weight functions and intervals corresponding to other families of orthogonal polynomials. The nodes and weights for a Gaussian quadrature rule can also be

computed by solving an eigenvalue problem for a tridiagonal matrix associated with the corresponding orthogonal polynomials and weight function.

Example 4 is typical in that for any n the Gaussian nodes are symmetrically placed about the midpoint of the interval; for odd values of n the midpoint itself is always a node. Example 4 is also typical in that the nodes are usually irrational numbers even when the endpoints a and b are rational. This feature makes Gaussian rules relatively inconvenient for hand computation, compared with simple NewtonCotes rules. When using a computer, however, the nodes and weights are usually tabulated in advance and contained in a subroutine that can be called when needed, so the user need not compute or even know their actual values.
Gaussian quadrature rules are also more difficult to apply than Newton-Cotes rules because the weights and nodes are derived for some specific interval, such as $[-1,1]$, and thus any other interval of integration $[a,b]$ must be transformed into the standard interval for which the nodes and weights have been tabulated. If we wish to use a quadrature rule that is tabulated on the interval $[\alpha,\beta]$,

$$\int_\alpha^\beta f(x)dx \approx \sum_{i=1}^n w_i f\left(x_i\right)$$

to approximate an integral on the interval $[a,b]$,

$$I(g)=\int_a^b g(t)dt$$

then we must use a change of variable from x in $[\alpha,\beta]$ to t in $[a,b]$. Many such transformations are possible, but a simple linear transformation

$$t=\frac{(b-a)x+a\beta-b\alpha}{\beta-\alpha}$$

has the advantage of preserving the degree of the quadrature rule. The integral is then given by

$$\begin{aligned} I(g) &= \frac{b-a}{\beta-\alpha}\int_\alpha^\beta g\left(\frac{(b-a)x+a\beta-b\alpha}{\beta-\alpha}\right)dx \\ &\approx \frac{b-a}{\beta-\alpha}\sum_{i=1}^n w_i g\left(\frac{(b-a)x_i+a\beta-b\alpha}{\beta-\alpha}\right) \end{aligned}$$

Example 5: Change of Interval

To illustrate a change of interval, we use the two-point Gaussian quadrature rule G_2 derived for the interval $[-1,1]$ in Example 4 to approximate the integral

$$I(g)=\int_0^1 e^{-t^2}dt$$

from Example 2. Using the linear transformation of variable just given, we have

$$t=\frac{x+1}{2}$$

so that the integral is approximated by $G_2(g)=$

$$\frac{1}{2}\left[\exp\left(-\left(\frac{(-1/\sqrt{3})+1}{2}\right)^2\right)+\exp\left(-\left(\frac{(1/\sqrt{3})+1}{2}\right)^2\right)\right]\approx 0.746595$$

which is slightly more accurate than the result given by Simpson's rule for this integral (see Example 2), despite using only two points instead of three.

By design, Gaussian quadrature rules have maximal degree, and hence optimal accuracy, for the number of points used. Moreover, it can be shown that the resulting weights are always positive for any n, so that Gaussian quadrature rules are always stable and the resulting approximate values converge to the exact integral as $n\to\infty$. Unfortunately, Gaussian quadrature rules also have a serious drawback: for $m\neq n, G_m$ and G_n have no nodes in common (except for the midpoint when m and n are both odd). Thus, Gaussian rules are not progressive, which means that when the number of nodes is increased, say from n to m, m new evaluations of the integrand are required rather than $m-n$. We will see next how this deficiency can be remedied, but at some cost in the degree attainable.

2.3.4. Progressive Gaussian Quadrature

We have just observed that Gaussian quadrature rules are not progressive: if all the nodes and weights are freely chosen to maximize the degree for any given number of nodes, then rules with different numbers of nodes will have essentially no nodes in common, which means that integrand values computed for one set of nodes cannot be reused in evaluating another rule with a different number of nodes.
Avoiding this additional work is the motivation for Kronrod quadrature rules. Such rules come in pairs: an n-point Gaussian rule G_n and a $(2n+1)$-point Kronrod rule K_{2n+1} whose nodes are optimally chosen subject to the constraint that all of the nodes of G_n are reused in K_{2n+1}. Thus, n of the nodes used in K_{2n+1} are prespecified, leaving the remaining $n+1$ nodes, as well as all $2n+1$ of the weights (including those corresponding to the nodes of G_n), free to be chosen to maximize the degree of the resulting rule. The rule K_{2n+1} is therefore of degree $3n+1$, whereas a true $(2n+1)$-point Gaussian rule would be of degree $4n+1$. Thus, there is a tradeoff between accuracy and efficiency.
One of the main reasons for using two quadrature rules with different numbers of points is to obtain an error estimate for the approximate value of the integral based on the difference between the values given by the two rules. In using a GaussKronrod pair, the value of K_{2n+1} is taken as the approximation to the integral, and a realistic but conservative estimate for the error, based partly on theory and partly on experience, is given by

$$\left(200\left|G_n-K_{2n+1}\right|\right)^{1.5}$$

Because they efficiently provide both high accuracy and a reliable error estimate, Gauss-Kronrod rules are among the most effective quadrature methods available, and they form the basis for many of the quadrature routines in major software libraries. The pair of rules (G_7, K_{15}), in particular, has become a commonly used standard.
This approach of adding optimally chosen new nodes to a prespecified set from another rule is taken a step further in Patterson quadrature rules. In particular, adding $2n+2$ optimally chosen nodes to the $2n+1$ nodes of the Kronrod rule K_{2n+1} yields a quadrature rule with degree $6n+4$ that reuses all of the $2n+1$ integrand values already computed for G_n and K_{2n+1}. For certain values of n, further extensions of this type are possible, and some of these have been tabulated for rules of up to 511 points.
Before we leave this topic, we note that much more modest extensions of Gaussian quadrature rules are sometimes useful. A true Gaussian quadrature rule is always open, i.e., the nodes never include the endpoints of the interval of integration. But for some purposes, it is useful if one or both endpoints are included. In Gauss-Radau quadrature, one of the endpoints of the interval of integration is prespecified as a node, leaving the remaining $n-1$ nodes, as well as all n weights, free to be chosen to maximize the degree of the resulting rule. There are $2n-1$ free parameters, so an n-point Gauss-Radau rule has degree $2n-2$. Similarly, in Gauss-Lobatto quadrature, both endpoints of the interval of integration are prespecified as nodes, leaving the remaining $n-2$ nodes, as well as all n weights, free
to be chosen to maximize the degree of the resulting rule. There are $2n-2$ free parameters, so an n-point Gauss-Lobatto rule has degree $2n-3$.

2.3.5. Composite Quadrature

Thus far we have considered simple quadrature rules obtained by interpolating the integrand function by a single polynomial over the entire interval of integration. The accuracy of such a rule can be increased, and the error estimated, by increasing the number of interpolation points, and hence the corresponding degree of the polynomial interpolant. Another alternative is to subdivide the original interval into two or more subintervals and apply a simple quadrature rule in each subinterval. Summing these partial results then yields an approximation to the overall integral. Such an approach is equivalent to using piecewise polynomial interpolation on the original interval and then integrating the piecewise interpolant to approximate the integral.
A composite, or compound, quadrature rule on a given interval $[a, b]$ results from subdividing the interval into k subintervals, typically of uniform length $h=(b-a)/k$, applying an n-point simple quadrature rule Q_n in each subinterval, and then taking the sum of these results as the approximate value of the integral. If the rule Q_n is open, then evaluating the composite rule will require kn evaluations of the integrand function. If Q_n is closed, on the other hand, then some of the points are repeated, so that only $k(n-1)+1$ evaluations of the integrand are required.

Example 6: Composite Quadrature Rules

If the interval $[a, b]$ is subdivided into k subintervals of length $h=(b-a)/k$ and $x_j=a+jh, j=0, \ldots, k$, then the composite midpoint rule is given by

$$M_k(f)=\sum_{j=1}^k\left(x_j-x_{j-1}\right) f\left(\frac{x_{j-1}+x_j}{2}\right)=h \sum_{j=1}^k f\left(\frac{x_{j-1}+x_j}{2}\right)$$

and the composite trapezoid rule is given by

$$\begin{aligned} T_k(f) & =\sum_{j=1}^k \frac{\left(x_j-x_{j-1}\right)}{2}\left(f\left(x_{j-1}\right)+f\left(x_j\right)\right) \\ & =h\left(\frac{1}{2} f(a)+f\left(x_1\right)+\cdots+f\left(x_{k-1}\right)+\frac{1}{2} f(b)\right)\end{aligned}$$

A composite rule is always stable provided the underlying rule Q_n is stable. Convergence to the exact integral as the number of subintervals $k \rightarrow \infty$ is guaranteed, provided the underlying rule Q_n has degree at least zero (i.e., it integrates constants exactly). To see why, let the composite rule be given by

$$C_k(f)=\sum_{j=1}^k\left(\sum_{i=1}^n w_i f\left(x_{i j}\right)\right)$$

where w_i is the i th weight of Q_n and $x_{i j}$ is the i th node of Q_n in the j th subinterval. Interchanging the order of summation, we have

$$C_k(f)=\sum_{i=1}^n\left(\sum_{j=1}^k w_i f\left(x_{i j}\right)\right)=\sum_{i=1}^n w_i\left(\sum_{j=1}^k f\left(x_{i j}\right)\right)=\frac{1}{h} \sum_{i=1}^n w_i\left(\sum_{j=1}^k h f\left(x_{i j}\right)\right)$$

The latter expression in parentheses is a Riemann sum, which by definition has limit $I(f)$ as $k \rightarrow \infty$. Thus, we have

$$\lim _{k \rightarrow \infty} C_k(f)=\frac{1}{h} \sum_{i=1}^n w_i \lim _{k \rightarrow \infty}\left(\sum_{j=1}^k h f\left(x_{i j}\right)\right)=I(f) \frac{1}{h} \sum_{i=1}^n w_i=I(f)$$

The last equality follows from the fact that Q_n integrates constants exactly, so that the sum of the weights is equal to h, the subinterval length. Thus, in principle, by taking k sufficiently large it is possible to achieve arbitrarily high accuracy (up to the limit of the arithmetic precision) using a composite rule, even with an underlying rule Q_n of low degree, although this may not be the most efficient way to attain a given level of accuracy. Indeed, the general error bound in Section §2.3 suggests that more can usually be gained by increasing n than by decreasing h. In practice, a compromise between these is usually most appropriate.
Composite quadrature rules offer a particularly simple means of estimating the error by using different levels of subdivision, which can easily be made progressive. We observed in Section §2.3.1 that halving the interval length reduces the error in the midpoint or trapezoid rules by a factor of about $1/8$. Halving their length doubles the number of subintervals, however, so the overall reduction in the error is by a factor of about $1/4$. If the number of subintervals is k, and hence the subinterval length is $h=(b-a)/k$, then the dominant term in the remainder for the composite midpoint or trapezoid rules is $\mathcal{O}\left(k h^3\right)=\mathcal{O}\left(h^2\right)$, so the accuracy of these rules is said to be of second order. Similarly, the composite Simpson's rule is of fourth-order accuracy, meaning that the dominant term in its remainder is $\mathcal{O}\left(h^4\right)$, and hence halving the subinterval length reduces the overall error by a factor of about $1/16$.

2.3.6. Adaptive Quadrature

A composite quadrature rule with an error estimate suggests a simple automatic quadrature procedure: continue subdividing all the subintervals until the estimated overall error meets the desired accuracy tolerance. Maintaining uniform subdivisions is grossly inefficient for many integrands, however, as large numbers of function evaluations may be expended in regions where the integrand function is well behaved and the accuracy tolerance is easily met. A more intelligent approach is adaptive quadrature, in which the interval of integration is selectively refined to reflect the behavior of any particular integrand function.

A typical adaptive quadrature strategy works as follows. First we need a pair of quadrature rules, say Q_{n_1} and Q_{n_2}, whose difference provides an error estimate. A simple example is the trapezoid and midpoint rules, whose difference overestimates the error in the more accurate rule by a factor of three, as we saw in Section §2.3.1. Greater efficiency is usually obtained with rules of higher degree, however, such as the Gauss-Kronrod pair (G_7, K_{15}). Another alternative is to use a single rule at two different levels of subdivision; Simpson's rule is a popular choice in this approach. In any case, to minimize the number of function evaluations required, the pair of rules should be progressive.

The adaptive procedure is now conceptually simple: apply both rules Q_{n_1} and Q_{n_2} on the initial interval of integration $[a, b]$. If the resulting approximate values for the integral differ by more than the desired tolerance, divide the interval into two or more subintervals and repeat the procedure on each subinterval. If the tolerance is met on a given subinterval, then no further subdivision of that subinterval will be required. If the tolerance is not met on a given subinterval, then the subdivision process is repeated again, and so on until the tolerance is met on all subintervals. Such a strategy leads to a nonuniform sampling of the integrand function that places many sample points in regions where the function is difficult to integrate and relatively few points where the function is easily integrated, as illustrated in Fig. 2.4.

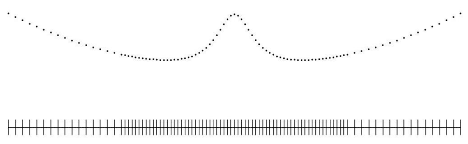

Figure 2.4: Typical placement of evaluation points by adaptive quadrature routine

The high-level description just given glosses over a number of important implementation issues, including

- How should the stopping criterion be implemented? For example, should the error tolerance be relative (usually preferable), or absolute (in case the value of the integral is near zero), or a combination of the two?
- Can the error tolerance always eventually be met? Will the recursion always terminate?
- How can we avoid wasting time subdividing unconverged subintervals that make a negligible contribution to the total integral?
- How should we allow for the effects of finite-precision arithmetic? For example, what if the length of a subinterval becomes so small that it contains no machine numbers other than its endpoints?

In many adaptive quadrature routines these issues are addressed by using some combination of relative and absolute error tests involving machine-dependent parameters (such as the machine precision ϵ_{mach}), together with an upper limit on the number of levels of subdivision allowed. When confronted with a difficult problem, such as a noisy or unsmooth integrand or an unrealistically tight error tolerance, such routines may expend a large number of function evaluations before returning
an inaccurate answer accompanied by a warning message that the subdivision limit was exceeded.

Gander and Gautschi suggested an alternative approach that often does considerably better. The key idea is to develop a termination criterion that is robust, machine independent, and avoids arbitrary limits on the depth of the recursion. An important ingredient in such a criterion is a rough prior estimate of the magnitude of the integral, call it $\hat{I}$. Such a rough estimate, which need merely be of the correct order of magnitude, can be obtained in various ways, such as sampling the integrand f at a few randomly chosen points within the interval of integration $[a, b]$. For a given subinterval, relative (or absolute) agreement of the quadrature rules Q_{n_1} and Q_{n_2} to within machine precision can then be checked by testing whether $\hat{I} + (Q_{n_2} - Q_{n_1})$ differs from $\hat{I}$. A user-supplied tolerance $\epsilon \geq \epsilon_{\text{mach}}$ can be implemented by increasing $\hat{I}$ by a factor of $\epsilon/\epsilon_{\text{mach}}$. Finally, an arbitrary limit on subdivision levels can be avoided in a similarly machine-independent way by testing whether the computed midpoint of a subinterval lies strictly within the subinterval. A generic adaptive quadrature procedure with these features is summarized in Algorithm 8.1.

```
Algorithm 8.1 Adaptive Quadrature
$\operatorname{procedure}$ adaptquad $(f, a, b, \hat{I})$
    $I_{1}=Q_{n_{1}}(f, a, b)$
    $I_{2}=Q_{n_{2}}(f, a, b)$
```

```
$m=a+(b-a) / 2$
if $(m \leq a$ or $m \geq b)$ then
    issue warning
    return $I_{2}$
end
if $\hat{I}+\left(I_{2}-I_{1}\right)=\hat{I}$ then
    return $I_{2}$
else
    return $(\operatorname{adaptquad}(f, a, m, \hat{I})+$
        $\operatorname{adaptquad}(f, m, b, \hat{I}))$
end
```

Although adaptive quadrature procedures tend to be very effective in practice, they can be fooled: both the approximate integral and the error estimate can be completely wrong. The reason is that the integrand function is sampled at only a finite number of points, so it is possible that significant features of the integrand may be missed. For example, it may happen that the interval of integration is very wide, but all of the "interesting" behavior of the integrand is confined to a very narrow range. In this case, sampling by the adaptive routine may completely miss the interesting part of the integrand's behavior, and the resulting value for the integral may be completely wrong. This situation may seem unlikely, but it can happen, for example, if we are trying to evaluate an integral over an unbounded interval and have truncated it unwisely.
Another potential difficulty with adaptive quadrature routines is that they may be very inefficient in handling discontinuities (finite jumps) in the integrand or its derivatives. For example, an adaptive routine may expend a great many function evaluations in refining the region around a discontinuity of the integrand because it assumes that the integrand is smooth (but very steep) at such a point. A good way to avoid such behavior is to split the interval at the point of discontinuity into two subintervals and call the quadrature routine separately in each, thereby obviating the need for the routine to resolve the discontinuity.

2.4. Other Integration Problems

2.4.1. Tabular Data

Thus far we have assumed that the integrand function can be evaluated at any desired point within the interval of integration. This assumption may not be valid if the integrand is defined only by a table of its values at selected discrete points, as is typical of empirical measurements, for example. A reasonable approach to integrating such tabular data is by piecewise interpolation. For example, integrating the piecewise linear interpolant to tabular data gives a composite trapezoid rule. An excellent method for integrating tabular data is provided by Hermite cubic or cubic spline interpolation. In effect, the overall integral is computed by integrating analytically each of the cubic pieces that make up the interpolant.

2.4.2. Improper Integrals

Boundedness of both the integrand function and the interval of integration are inherent in the definition of the Riemann integral. If either the integrand or the interval is unbounded, then it may still be possible to define an improper integral. For an unbounded interval, say $[a, \infty)$, the improper integral is defined as

$$\int_a^\infty f(x)dx = \lim_{b\to\infty}\int_a^b f(x)dx$$

provided f is integrable on $[a, b]$ for any finite b, and the indicated limit exists and is finite. Unbounded intervals of the form $(-\infty, b]$ or $(-\infty, \infty)$ are treated similarly. For an integrand that is unbounded at a point $c \in [a, b]$, i.e., a vertical asymptote or singularity of the integrand, the improper integral is defined as

$$\int_a^b f(x)dx = \lim_{\gamma\to c-}\int_a^\gamma f(x)dx + \lim_{\gamma\to c+}\int_\gamma^b f(x)dx$$

provided that f is integrable on any subinterval $[a, \gamma] \subseteq [a, c)$ and $[\gamma, b] \subseteq (c, b]$, and the indicated limits exist and are finite. There are a number of ways of computing such improper integrals, assuming of course that the integral exists.
For an unbounded interval of integration, one may be able to compute the improper integral using a standard quadrature routine for a finite interval. A number of approaches are possible:

- Replace any infinite limit of integration by a finite value. Such a finite limit should be chosen carefully so that the omitted tail is negligible or its contribution to the integral can be estimated. But the remaining finite interval should not be so wide that an adaptive quadrature routine will be fooled into sampling the integrand badly.
- Transform the variable of integration so that the new interval is finite. Typical transformations include $x = -\log t$ or $x = t/(1-t)$. Care must be taken not to introduce singularities or other difficulties by such a transformation.

Another alternative is to use a quadrature rule, such as Gauss-Laguerre or GaussHermite, that is designed for an unbounded interval.
For an integrand having an integrable singularity within the interval of integration, one may be tempted simply to try an adaptive quadrature routine and hope that it will work, but such an approach is unlikely to prove satisfactory. Outright failure will result if the integrand happens to be evaluated at the singularity, which will likely occur if the singularity lies at one of the endpoints, as singularities often do. Even if the routine is lucky enough to avoid evaluating the integrand at the singularity, an adaptive quadrature routine will generally be extremely inefficient for an integrand having a singularity because polynomials, which never have vertical asymptotes, cannot efficiently approximate functions that do (recall that our error bounds depend on higher derivatives of the integrand, which will inevitably be large near a singularity).
A better approach for dealing with a singularity in the integrand is to remove the singularity either by transforming the variable of integration or by dividing out or subtracting off an analytically integrable function having the same singularity. As an example of the first approach, the integral

$$\int_0^{\pi/2} \frac{\cos(x)}{\sqrt{x}} dx$$

has a singularity at $x = 0$, but under the transformation $x = t^2$ it becomes the innocuous integral

$$\int_0^{\pi/2} \frac{\cos(x)}{\sqrt{x}} dx = \int_0^{\sqrt{\pi/2}} \frac{\cos\left(t^2\right)}{t} 2t dt = 2\int_0^{\sqrt{\pi/2}} \cos\left(t^2\right) dt$$

which has no singularity and is easily integrable by an adaptive routine. As an example of the second approach, the integral

$$\int_0^{\pi/2} \frac{1}{\sqrt{\sin x}} dx$$

has a singularity at $x = 0$, but by subtracting off $1/\sqrt{x}$ we have

$$\int_0^{\pi/2} \frac{1}{\sqrt{\sin x}} dx = \int_0^{\pi/2} \left(\frac{1}{\sqrt{\sin x}} - \frac{1}{\sqrt{x}}\right) dx + \int_0^{\pi/2} \frac{1}{\sqrt{x}} dx$$

The first of the two resulting integrands is now well behaved near 0 and can safely be replaced by 0 at 0 , and the second is analytically integrable to give the value
$\sqrt{2\pi}$. There is often some art involved in finding such transformations, and not all singularities are removable in this manner. If there is more than one singularity, then the transformations required to remove them may conflict, in which case a remedy is to break the interval of integration into subintervals, each of which contains at most one singularity, typically at one endpoint.

2.4.3. Double Integrals

Thus far we have considered only one-dimensional integrals, where we wish to determine the area under a curve over an interval. In evaluating a two-dimensional, or double integral, we wish to compute the volume under a surface over a planar region. For a rectangular region $[a,b] \times [c,d] \subseteq \mathbb{R}^2$, a double integral has the form

$$\int_a^b \int_c^d f(x,y) dx dy$$

For a more general domain $\Omega \subseteq \mathbb{R}^2$, the integral takes the form

$$\iint_\Omega f(x,y) dA$$

By analogy with numerical quadrature for one-dimensional integrals, the numerical approximation of two-dimensional integrals is sometimes called numerical cubature.
To evaluate a double integral, a number of approaches are available, including the following:

- Use a pair of adaptive one-dimensional quadrature routines, one for the outer integral and the other for the inner integral. Each time the outer routine calls its integrand function, the latter in turn calls the inner quadrature routine. This approach requires some care in setting the error tolerances for the respective quadrature routines.
- Use a Cartesian product rule. Such rules result from applying one-dimensional quadrature rules in successive dimensions. This approach is limited to regions that can be decomposed into rectangles.
- Use a nonproduct interpolatory cubature rule. Such rules, with error estimates, are available for a number of standard regions, the most important of which for adaptive use is triangles, since many two-dimensional regions can be efficiently triangulated to any desired level of refinement.

2.4.4. Multiple Integrals

To evaluate a multiple integral in dimensions higher than two, the options just listed for double integrals still work in principle, but their cost grows rapidly with the number of dimensions. The only generally viable approach for computing integrals in higher dimensions is the Monte Carlo method. The function is sampled at n points distributed randomly in the domain of integration, and then the mean of these function values is multiplied by the area (or volume, etc.) of the domain to obtain an estimate for the integral. The error in this estimate goes to zero as $1/\sqrt{n}$,
which means, for example, that to gain an additional decimal digit of accuracy the number of sample points must be increased by a factor of 100. For this reason, it is not unusual for Monte Carlo calculations of integrals to require millions of evaluations of the integrand.
The Monte Carlo method is not competitive for integrals in one or two dimensions, but the beauty of the method is that its convergence rate is independent of the number of dimensions. Thus, for example, one million points in six dimensions amounts to only ten points per dimension, which is much better than any type of conventional quadrature rule would require for the same level of accuracy. The efficiency of Monte Carlo integration can be enhanced by various methods for biasing the sampling, either to achieve more uniform coverage of the sampled volume (e.g., by avoiding undesirable random clumping of the sample points) or to concentrate sampling in regions where the integrand is largest in magnitude (importance sampling) or in variability (stratified sampling), in a spirit similar to adaptive quadrature.

2.5. Integral Equations

An integral equation is an equation in which the unknown to be determined is a function inside an integral sign. An integral equation can be thought of as a continuous analogue, or limiting case, of a system of algebraic equations. For example, the analogue of a linear system $\boldsymbol{Ax} = \boldsymbol{y}$ is a Fredholm integral equation of the first kind, which has the form

$$\int_a^b K(s,t)u(t)dt = f(s)$$

where the functions K, called the kernel, and f are known, and the function u is to be determined. Integral equations arise naturally in many fields of science and engineering, particularly observational sciences (e.g., astronomy, seismology, spectrometry, tomography), where the kernel K represents the response function of an instrument (determined by calibration with known signals), f represents measured data, and u represents the underlying signal that is sought. In effect, we are trying to resolve the measured data f as a (continuous) linear combination of standard signals. Integral equations also result from Green's function or boundary element methods [289, 290] for solving differential equations.
Establishing the existence and uniqueness of solutions to integral equations is much more problematic than with algebraic equations. Moreover, when a solution does exist, it may be extremely sensitive to perturbations in the input data, which are often subject to random experimental or measurement errors. The reason for this sensitivity is that integration is a smoothing process, so its inverse (i.e., determining the integrand from the integral) is just the opposite. Integrating an arbitrary function u against a smooth kernel K dampens any high-frequency oscillation, so solving for u tends to introduce high-frequency oscillation in the result.
According to the Riemann-Lebesgue Lemma,

$$\lim_{n\to\infty} \int_a^b K(s,t)\sin(nt)dt = 0$$

for any integrable kernel K, which implies that an arbitrarily high-frequency component of u has an arbitrarily small effect on f. Thus, integral equations of the first kind with smooth kernels are always ill-conditioned.

A standard technique for solving integral equations numerically is to use a quadrature rule to replace the integral by an approximating finite sum. Denote the nodes and weights of the quadrature rule by t_j and $w_j, j = 1, \ldots, n$. We also choose n points s_i for the variable s, often the same as the t_j, but not necessarily so. Then the approximation to the integral equation becomes

$$\sum_{j=1}^{n} w_j K\left(s_i, t_j\right) u\left(t_j\right) = f\left(s_i\right), \quad i = 1, \ldots n$$

This is a system of linear algebraic equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$, where $a_{ij} = w_j K\left(s_i, t_j\right)$, $y_i = f\left(s_i\right)$, and $x_j = u\left(t_j\right)$, which can be solved for $\boldsymbol{x}$ to obtain a discrete sample of approximate values of the solution function u.

Example 7: Integral Equation

Consider the integral equation

$$\int_{-1}^{1} (1 + \alpha st) u(t) dt = 1$$

i.e., $K(s, t) = 1 + \alpha st$ and $f(s) = 1$, where α is a known positive constant whose value is unspecified for now. Using the composite midpoint quadrature rule with two subintervals, taking $t_1 = -\frac{1}{2}, t_2 = \frac{1}{2}$, and $w_1 = w_2 = 1$, and also taking $s_1 = -\frac{1}{2}$ and $s_2 = \frac{1}{2}$, we obtain the linear system

$$\boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} 1 + \alpha/4 & 1 - \alpha/4 \\ 1 - \alpha/4 & 1 + \alpha/4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \boldsymbol{y}$$

It is easily verified that the solution to this linear system is $\boldsymbol{x} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \end{bmatrix}^T$, independent of the value of α.
Now suppose that the errors in the measured values of $y_1 = f\left(s_1\right)$ and $y_2 = f\left(s_2\right)$ are ϵ_1 and ϵ_2, respectively. Then by linearity, the change in the solution $\boldsymbol{x}$ is given by the same linear system, but with right-hand side $\begin{bmatrix} \epsilon_1 & \epsilon_2 \end{bmatrix}^T$. The resulting change in $\boldsymbol{x}$ is therefore given by

$$\Delta \boldsymbol{x} = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = \begin{bmatrix} \left(\epsilon_1 - \epsilon_2\right)/\alpha + \left(\epsilon_1 + \epsilon_2\right)/4 \\ \left(\epsilon_2 - \epsilon_1\right)/\alpha + \left(\epsilon_1 + \epsilon_2\right)/4 \end{bmatrix}$$

Thus, if α is sufficiently small, the relative error in the computed value for $\boldsymbol{x}$ can be arbitrarily large. A very small value for α in this particular kernel corresponds to a very insensitive instrument with a very flat response. This is reflected in
the conditioning of the matrix $\boldsymbol{A}$, whose columns become more nearly linearly dependent as α decreases in magnitude. This simple example is typical of integral equations with smooth kernels.

Note that the sensitivity in the previous example is inherent in the problem and is not due to the method of solving it. In general, such an integral operator with a smooth kernel has zero as an eigenvalue (i.e., there are nonzero functions that it annihilates), and hence using a more accurate quadrature rule makes the conditioning of the linear system worse and the resulting solution more erratic. Because of this behavior, additional information may be required to obtain a physically meaningful solution. Such techniques include:

- Truncated singular value decomposition. The solution to the system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ is computed using the SVD of $\boldsymbol{A}$; but the small singular values of $\boldsymbol{A}$, which reflect the ill-conditioning, are omitted from the solution.
- Regularization. A damped solution is obtained by solving the minimization problem

$$\min_{\boldsymbol{x}} \left(\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \mu\|\boldsymbol{x}\|_2^2\right)$$

where the nonnegative parameter μ determines the relative weight given to the norm of the residual and the norm of the solution. This minimization problem is equivalent to the linear least squares problem

$$\begin{bmatrix} \boldsymbol{A} \\ \sqrt{\mu}\boldsymbol{I} \end{bmatrix} \boldsymbol{x} \cong \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{0} \end{bmatrix}$$

More generally, other norms, usually based on first or second differences between its components, can also be used to weight the smoothness of the solution. The LevenbergMarquardt method for nonlinear least squares problems is another example of regularization.

- Constrained optimization. Some norm of the residual $\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|$ is minimized subject to constraints on $\boldsymbol{x}$ that disallow nonphysical solutions. In many applications, for example, the components of the solution $\boldsymbol{x}$ are required to be nonnegative or monotonic.

We have considered only Fredholm integral equations of the first kind. Many other types arise in practice, including integral equations of the second kind (eigenvalue problems), Volterra integral equations (which differ from Fredholm integral equations in that the upper limit of integration is the variable s instead of the fixed value b), singular integral equations (in which one or both of the limits of integration are infinite), and nonlinear integral equations. All types of integral equations can be discretized by means of numerical quadrature, yielding a system of algebraic equations. Alternatively, the unknown function u can be approximated by a linear combination $u(t) \approx \sum_{j=1}^{n} c_j \phi_j(t)$ of suitably chosen basis functions ϕ_j, which leads
to a system of algebraic equations for the coefficients c_j.

2.6. Numerical Differentiation

We now turn briefly to numerical differentiation. It is important to realize that differentiation is an inherently sensitive problem, as small perturbations in the data can cause large changes in the result. Integration, on the other hand, is a smoothing process and is inherently stable in this respect. The contrast between differentiation and integration should not be surprising, since they are inverse processes to each other. The difference between them is illustrated in Fig. 2.5, which shows two functions that have equal definite integrals but very different derivatives.

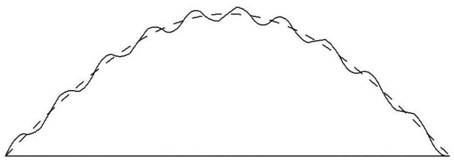
Figure 2.5: Two functions whose integrals are equal but whose derivatives are not

When approximating the derivative of a function whose values are known only at a discrete set of points, a good approach is to fit some smooth function to the given discrete data and then differentiate the approximating function to approximate the derivatives of the original function. If the given data are sufficiently smooth, then interpolation may be appropriate; but if the given data are noisy, then a smoothing approximating function, such as a least squares polynomial or spline, is more appropriate.

2.6.1. Finite Difference Approximations

Although finite difference formulas are generally inappropriate for discrete or noisy data, they are very useful for approximating derivatives of a smooth function that is known analytically, or can be evaluated accurately for any given argument, or is defined implicitly by a differential equation. We now develop some finite difference formulas that will be useful in our study of the numerical solution of differential equations.
Given a smooth function $f:\mathbb{R}\to\mathbb{R}$, we wish to approximate its first and second derivatives at a point x. For a given step size h, consider the Taylor series expansions

$$f(x+h)=f(x)+f'(x)h+\frac{f''(x)}{2}h^2+\frac{f'''(x)}{6}h^3+\cdots$$

and

$$f(x-h)=f(x)-f'(x)h+\frac{f''(x)}{2}h^2-\frac{f'''(x)}{6}h^3+\cdots$$

Solving for $f'(x)$ in the first series, we obtain the forward difference formula

$$\begin{aligned}f'(x)&=\frac{f(x+h)-f(x)}{h}-\frac{f''(x)}{2}h+\cdots\\&\approx\frac{f(x+h)-f(x)}{h}\end{aligned}$$

which gives an approximation that is first-order accurate since the dominant term in the remainder of the series is $\mathcal{O}(h)$. Similarly, from the second series we derive the backward difference formula

$$\begin{aligned} f'(x) &= \frac{f(x)-f(x-h)}{h} + \frac{f''(x)}{2}h + \cdots \\ &\approx \frac{f(x)-f(x-h)}{h} \end{aligned}$$

which is also first-order accurate. Subtracting the second series from the first gives the centered difference formula

$$\begin{aligned} f'(x) &= \frac{f(x+h)-f(x-h)}{2h} - \frac{f'''(x)}{6}h^2 + \cdots \\ &\approx \frac{f(x+h)-f(x-h)}{2h} \end{aligned}$$

which is second-order accurate. Finally, adding the two series together gives a centered difference formula for the second derivative

$$\begin{aligned} f''(x) &= \frac{f(x+h)-2f(x)+f(x-h)}{h^2} - \frac{f^{(4)}(x)}{12}h^2 + \cdots \\ &\approx \frac{f(x+h)-2f(x)+f(x-h)}{h^2} \end{aligned}$$

which is also second-order accurate. By using function values at additional points, $x \pm 2h, x \pm 3h, \ldots$, we can derive similar finite difference approximations with still higher accuracy or for higher-order derivatives.
Note that higher-accuracy difference formulas require more function values. Whether these translate into higher overall cost depends on the particular situation, since a more accurate formula may permit the use of a larger step size h and correspondingly fewer steps. In choosing a value for h, rounding error must also be considered in addition to the truncation error given by the series expansion.
Using Taylor series expansions to derive finite difference formulas becomes increasingly cumbersome for approximations of increasingly high accuracy or higherorder derivatives. We next consider an alternative method based on polynomial interpolation that is not only more convenient, but will also more readily permit later generalization, for example to unequally spaced points. Let $t_i, i=1,\ldots,n$ be equally spaced points in $\mathbb{R}$, with step size $h = t_{i+1}-t_i, i=1,\ldots,n-1$. Let $y_i = f(t_i), i=1,\ldots,n$, where $f:\mathbb{R}\to\mathbb{R}$ is a smooth function.
For $i=1,\ldots,n-1$, the polynomial of degree one interpolating the two data points $(t_i,y_i),(t_{i+1},y_{i+1})$ is given by the Lagrange interpolant

$$p(t) = y_i\frac{t-t_{i+1}}{t_i-t_{i+1}} + y_{i+1}\frac{t-t_i}{t_{i+1}-t_i} = y_i\frac{t-t_{i+1}}{-h} + y_{i+1}\frac{t-t_i}{h}$$

Differentiating this polynomial with respect to t, we have

$$p'(t) = \frac{y_{i+1}-y_i}{h}$$

which is the same as the first-order, forward difference formula for the first derivative that we derived earlier using Taylor series. The first-order, backward difference formula for the first derivative can be derived similarly by interpolating $(t_{i-1},y_{i-1}),(t_i,y_i)$
For $i=2,\ldots,n-1$, the polynomial of degree two interpolating the three data points $(t_{i-1},y_{i-1}),(t_i,y_i),(t_{i+1},y_{i+1})$ is given by the Lagrange interpolant

$$\begin{aligned} p(t) =& y_{i-1}\frac{(t-t_i)(t-t_{i+1})}{(t_{i-1}-t_i)(t_{i-1}-t_{i+1})} + y_i\frac{(t-t_{i-1})(t-t_{i+1})}{(t_i-t_{i-1})(t_i-t_{i+1})} \\ &+ y_{i+1}\frac{(t-t_{i-1})(t-t_i)}{(t_{i+1}-t_{i-1})(t_{i+1}-t_i)} \\ =& y_{i-1}\frac{(t-t_i)(t-t_{i+1})}{2h^2} + y_i\frac{(t-t_{i-1})(t-t_{i+1})}{-h^2} + y_{i+1}\frac{(t-t_{i-1})(t-t_i)}{2h^2} \end{aligned}$$

Differentiating this polynomial with respect to t, we have

$$p'(t) = y_{i-1}\frac{(t-t_i)+(t-t_{i+1})}{2h^2} + y_i\frac{(t-t_{i-1})+(t-t_{i+1})}{-h^2} + y_{i+1}\frac{(t-t_{i-1})+(t-t_i)}{2h^2},$$

and evaluating the derivative at $t=t_i$ then gives

$$p'(t_i) = \frac{y_{i+1} - y_{i-1}}{2h}$$

which is the same as the second-order, centered difference formula for the first derivative that we derived earlier using Taylor series. Finally, differentiating a second time gives

$$p''(t) = y_{i-1}\frac{2}{2h^2} + y_i\frac{2}{-h^2} + y_{i+1}\frac{2}{2h^2} = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}$$

which is the same as the second-order, centered difference formula for the second derivative that we derived earlier using Taylor series.
We can continue in this manner to compute approximate derivatives of higher accuracy or higher order by interpolating more points and using correspondingly higher degree polynomials. Indeed, we can achieve the highest possible accuracy or order for a given number of data points by interpolating all of the data points with a single polynomial. Moreover, this interpolatory approach is easily generalized. We could use other representations of the interpolating polynomial. We could use unequally spaced points, for example the Chebyshev points to achieve higher accuracy. We could use interpolating functions other than polynomials, for example trigonometric functions for periodic data.

2.6.2. Automatic Differentiation

A number of alternatives are available for computing derivatives of a function, including finite difference approximations or closed-form formulas determined either by hand or by a computer algebra package. Each of these methods has significant drawbacks, however: manual differentiation is tedious and error-prone; symbolic derivatives tend to be unwieldy for complicated functions; finite difference approximations require a sometimes delicate choice of step size, and their accuracy is limited by discretization error.
Another alternative, at least for any function expressed by a computer program, is automatic differentiation, often abbreviated as AD. The basic idea of AD is simple: a computer program consists of basic arithmetic operations and elementary functions, each of whose derivatives is easily computed. Thus, the function computed by the program is, in effect, a composite of many simple functions whose derivatives can be propagated through the program by repeated use of the chain rule, effectively computing the derivative of the function step by step along with the function itself. The result is the true derivative of the original function, subject only to rounding error but suffering no discretization error.
Though AD is conceptually simple, its practical implementation is more complicated, requiring careful analysis of the input program and clever strategies for reducing the potentially explosive complexity of the resulting derivative code. Fortunately, most of these practical impediments have been successfully overcome, and a number of effective software packages are now available for automatic differentiation. Some of these packages accept a Fortran or C input program and then output a second program for computing the desired derivatives, whereas other packages use operator overloading to perform derivative computations automatically along with the function evaluation. When applicable, AD can be much easier, more efficient, and more accurate than other methods for computing derivatives. AD can also be useful for determining the sensitivity of the output of a program to perturbations in its input parameters. Such information might otherwise be obtainable only through many repeated runs of the program, which could be prohibitively expensive for a large, complex program.

8.7 Richardson Extrapolation

In many problems, such as numerical integration or differentiation, we compute an approximate value for some quantity based on some step size. Ideally, we would like to obtain the limiting value as the step size goes to zero, but we cannot take the step size to be arbitrarily small because of excessive cost or rounding error. Based on values for nonzero step sizes, however, we may be able to estimate what the value would be for a step size of zero.
Let $F(h)$ denote the value obtained with step size h. If we compute the value of F for some nonzero step sizes, and if we know the theoretical behavior of $F(h)$ as $h \to 0$, then we can extrapolate from the known values to obtain an approximate value for $F(0)$. This extrapolated value should have a higher-order accuracy than the values on which it is based. We emphasize, however, that the extrapolated value, though an improvement, is still only an approximation, not the exact solution, and its accuracy is still limited by the step size and arithmetic precision used.
To be more specific, suppose that

$$F(h) = a_0 + a_1h^p + \mathcal{O}(h^r)$$

as $h \to 0$ for some p and r, with $r > p$. We assume that we know the values of p and r, but not a_0 or a_1. Indeed, $F(0) = a_0$ is the quantity we seek. Suppose that we have computed F for two step sizes, say, h and h/q for some positive integer q. Then we have

$$F(h) = a_0 + a_1 h^p + \mathcal{O}\left(h^r\right)$$

and

$$F(h/q) = a_0 + a_1 (h/q)^p + \mathcal{O}\left(h^r\right) = a_0 + a_1 q^{-p} h^p + \mathcal{O}\left(h^r\right)$$

This system of two linear equations in the two unknowns a_0 and a_1 is easily solved to obtain

$$a_0 = F(h) + \frac{F(h) - F(h/q)}{q^{-p} - 1} + \mathcal{O}\left(h^r\right)$$

Thus, the accuracy of the improved value, a_0, is $\mathcal{O}\left(h^r\right)$ rather than $\mathcal{O}\left(h^p\right)$.
If $F(h)$ is known for several values of h, then the extrapolation process can be repeated to produce still more accurate approximations, up to the limitations imposed by finite-precision arithmetic. For example, if we have computed F for the values $h, h/2$, and $h/4$, then the extrapolated value based on h and $h/2$ can be combined with the extrapolated value based on $h/2$ and $h/4$ in a further extrapolation to produce a still more accurate estimate for $F(0)$.

Example 8: Richardson Extrapolation

To illustrate Richardson extrapolation, we use it to improve the accuracy of a finite difference approximation to the derivative of the function $\sin(x)$ at the point $x = 1$. Using the first-order accurate, forward difference formula derived in Section §2.6.1, we have for this problem

$$F(h) = a_0 + a_1 h + \mathcal{O}\left(h^2\right)$$

which means that $p = 1$ and $r = 2$ in this case. Using step sizes of $h = 0.5$ and $h/2 = 0.25$ (i.e., $q = 2$), we obtain

$$F(h) = \frac{\sin(1.5) - \sin(1)}{0.5} = 0.312048$$

and

$$F(h/2) = \frac{\sin(1.25) - \sin(1)}{0.25} = 0.430055$$

The extrapolated value is then given by

$$F(0) = a_0 = F(h) + \frac{F(h) - F(h/2)}{(1/2) - 1} = 2F(h/2) - F(h) = 0.548061$$

For comparison, the correctly rounded result is given by $\cos(1) = 0.540302$.

Example 9: Romberg Integration

As another example of Richardson extrapolation, we evaluate the integral

$$\int_0^{\pi/2} \sin(x) dx$$

If we use the composite trapezoid quadrature rule, we recall from Section §2.3.5 that

$$F(h) = a_0 + a_1 h^2 + \mathcal{O}\left(h^4\right)$$

which means that $p = 2$ and $r = 4$ in this case. With $h = \pi/2$, we obtain the value $F(h) = 0.785398$. Taking $q = 2$, we obtain the value $F(h/2) = F(\pi/4) = 0.948059$. The extrapolated value is then given by

$$F(0)=a_0=F(h)+\frac{F(h)-F(h/2)}{2^{-2}-1}=\frac{4F(h/2)-F(h)}{3}=1.002280$$

which is substantially more accurate than either value previously computed (the exact answer is 1). In this example the extrapolation is quadratic, as can be seen on the right in Fig. 8.6, because the lowest-order term in h is quadratic.

For any integer $k \geq 0$, let $T_{k,0}$ denote the approximation to the integral $\int_a^b f(x)dx$ given by the composite trapezoid rule with step size $h_k=(b-a)/2^k$. Then for any integer $j, j=1,\ldots,k$, we can recursively define the successive extrapolated values

$$T_{k,j}=\frac{4^j T_{k,j-1}-T_{k-1,j-1}}{4^j-1}$$

which form a triangular array

$$\begin{array}{lllll}
T_{0,0} & & & & \\
T_{1,0} & T_{1,1} & & & \\
T_{2,0} & T_{2,1} & T_{2,2} & & \\
T_{3,0} & T_{3,1} & T_{3,2} & T_{3,3} & \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{array}$$

.

In this example we have already computed $T_{0,0}=0.785398, T_{1,0}=0.948059$, and the extrapolated value $T_{1,1}=1.002280$. If we reduce the step size by another factor of two in the composite trapezoid rule, we obtain $T_{2,0}=F(h/4)=F(\pi/8)=$ 0.987116 . We can now combine the results for $h/2$ and $h/4$ to obtain the extrapolated value

$$T_{2,1}=F(h/2)+\frac{F(h/2)-F(h/4)}{2^{-2}-1}=\frac{4T_{2,0}-T_{1,0}}{4-1}=1.000135$$

Because we have eliminated the leading $\mathcal{O}\left(h^2\right)$ error term for the composite trapezoid rule, the accuracy of the first level of extrapolated values is $\mathcal{O}\left(h^4\right)$. Thus, we can further extrapolate on these values, but now with $p=4$, to obtain

$$T_{2,2}=\frac{4^2 T_{2,1}-T_{1,1}}{4^2-1}=\frac{16\times 1.000135-1.002280}{15}=0.999992$$

which is still more accurate than any of the values computed previously. Recursive computation of extrapolated values in this manner, based on the composite trapezoid rule with successively halved step sizes, is called Romberg integration. It is capable of producing very high accuracy (up to the limit imposed by the arithmetic precision) for very smooth integrands. It is often implemented in an automatic (though nonadaptive) fashion, with the recurrence continuing until the difference in successive diagonal entries of the triangular array falls below a specified error tolerance.