8 Numerical integration

In this chapter, we study algorithms for approximating the definite integral

$$\int_a^b f(x)dx.$$

We assume that $[a, b]$ is finite and $f(x)$ is continuous. We have the experience with Calculus that finding the elementary antiderivative of $f(x)$ can be rather challenging, and in many cases impossible even for $f(x)$ with quite simple expressions, such as $f(x) = \sqrt[3]{x^2+1}$, $\frac{1}{\ln x}$, $\frac{\sin x}{x}$, e^{-x^2}, and so on. In addition, we may not have an analytic expression of $f(x)$ but instead can only evaluate it wherever convenient. In these cases, a most commonly used solution is to find an approximate value of the integral by numerical integration (or quadrature). Our focus would be on a variety of quadrature rules that strike different levels of balance between the accuracy and evaluation cost.

8.1 Preliminaries

A fundamental idea for quadrature is to use a polynomial $p_n(x)$ to approximate $f(x)$ on $[a, b]$, so that $\int_a^b f(x)dx$ can be approximated by $\int_a^b p_n(x)dx$, and the integration of polynomials is relatively easy. The first thought here is to let $p_n(x) = \sum_{k=0}^n f(x_k)L_k(x)$ be a Lagrange interpolation of $f(x)$ at distinct nodes $x_0, x_1, \ldots, x_n \in [a, b]$. Let $w_k = \int_a^b L_k(x)dx$, and it follows that

$$Q(f) \equiv \sum_{k=0}^n w_k f(x_k) = \int_a^b \sum_{k=0}^n f(x_k)L_k(x)dx = \int_a^b p_n(x)dx.$$

Here, $\{x_k\}_{k=0}^n$ and $\{w_k\}_{k=0}^n$ are called the quadrature nodes and weights, respectively. The weights usually depend on the nodes, which should be *independent* of $f(x)$. A quadrature rule is defined by the choice of nodes and weights.

The generic quadrature rule above is a linear functional. For any continuous functions f, g and scalars α, β, $Q(\alpha f + \beta g) = \sum_{k=0}^n w_k \left(\alpha f(x_k) + \beta g(x_k)\right) = \alpha \left(\sum_{k=0}^n w_k f(x_k)\right) + \beta \left(\sum_{k=0}^n w_k g(x_k)\right) = \alpha Q(f) + \beta Q(g)$.

Note that $\sum_{k=0}^n w_k = b - a$. In fact, consider the function $f(x) \equiv 1$ on $[a, b]$, which is a special polynomial of degree $\leq n$. The polynomial interpolation $p_n(x)$ of $f(x)$ at $\{(x_k, f(x_k))\}_{k=0}^n$ of degree $\leq n$ is unique, and therefore $p_n(x)$ must be $f(x)$ itself. It follows that $p_n(x) = \sum_{k=0}^n f(x_k)L_k(x) = \sum_{k=0}^n L_k(x) \equiv 1$, and therefore $\sum_{k=0}^n w_k = \sum_{k=0}^n \int_a^b L_k(x)dx = \int_a^b \sum_{k=0}^n L_k(x)dx = b - a$.

The *degree of accuracy m* for a quadrature rule is defined the highest degree of *all* polynomial integrands for which the rule gives exact value of the definite integral. For example, consider the midpoint rule $Q(f) = w_0 f(x_0)$, where $x_0 = \frac{a+b}{2}$ and $w_0 = \int_a^b L_0(x)dx = b-a$. If $f(x) = \alpha x + \beta$, then $\int_a^b f(x)dx = \alpha\frac{(b-a)^2}{2} + \beta(b-a)$, and the quadrature is $Q(f) = w_0 f(x_0) = (b-a)\left[\alpha\left(\frac{a+b}{2}\right) + \beta\right] = \int_a^b f(x)dx$. We can verify that the midpoint rule is generally not exact for a quadratic integrand. The degree of accuracy of this rule is 1.

Theorem 57. *The degree of accuracy m for quadrature rules based on Lagrange interpolation at $n+1$ distinct nodes satisfies $m \geq n$, and $m \geq n+1$ if $x_{\frac{n}{2}} = \frac{a+b}{2}$ and $x_{\frac{n}{2}-k}$ and $x_{\frac{n}{2}+k}$ $(k = 1, \ldots, \frac{n}{2})$ are symmetric with respect to $x_{\frac{n}{2}}$.*

Proof. Given arbitrary $n+1$ distinct nodes $x_0, \ldots, x_n$, there is a unique polynomial $p_n(x)$ of degree $\leq n$ going through the data points $\{x_k, f(x_k)\}_{k=0}^n$. If the integrand $f(x)$ itself is a polynomial of degree $\leq n$, the corresponding interpolation $p_n(x)$ must be identical to $f(x)$. Therefore, the quadrature rule based on $\int_a^b p_n(x)dx$ gives exact value of $\int_a^b f(x)dx$. In other words, any quadrature rule based on Lagrange interpolation of $f(x)$ necessarily has a degree of accuracy $m \geq n$, wherever the $n+1$ distinct nodes are located on $[a, b]$.

Next we want show that $m \geq n+1$ if n is even, $x_{\frac{n}{2}} = \frac{a+b}{2}$, and $x_{\frac{n}{2}-k}$ and $x_{\frac{n}{2}+k}$ $(k = 1, \ldots, \frac{n}{2})$ are symmetric with respect to $x_{\frac{n}{2}}$. Let $p_{n+1}(x) = c_{n+1}x^{n+1} + c_n x^n + \ldots + c_0$ be an arbitrary given polynomial of degree $\leq n+1$. Define $p_{n+1}^c(x) = \left(x - \frac{a+b}{2}\right)^{n+1}$ such that $p_{n+1}(x) = c_{n+1}p_{n+1}^c(x) + r_n(x)$, where r_n is a polynomial of degree $\leq n$. Due to the linearity of quadrature rules,

$$Q(p_{n+1}) = c_{n+1}Q(p_{n+1}^c) + Q(r_n) = c_{n+1}Q(p_{n+1}^c) + \int_a^b r_n(x)dx$$

$$= \quad c_{n+1}\sum_{k=0}^{n} w_k \left(x_k - \frac{a+b}{2}\right)^{n+1} + \int_a^b r_n(x)dx = \int_a^b r_n(x)dx,$$

where $\sum_{k=0}^n w_k \left(x_k - \frac{a+b}{2}\right)^{n+1} = 0$ because $n+1$ is odd, $x_{\frac{n}{2}} = \frac{a+b}{2}$, $x_{\frac{n}{2}-k}$ and $x_{\frac{n}{2}+k}$ $(k = 1, \ldots, \frac{n}{2})$ are symmetric with respect to $x_{\frac{n}{2}}$, and $w_{\frac{n}{2}-k} = w_{\frac{n}{2}+k}$ due to the symmetry of the nodes (can be shown without difficulty).

Meanwhile, it is also easy to see that $\int_a^b p_{n+1}(x)dx = c_{n+1}\int_a^b p_{n+1}^c(x)dx + \int_a^b r_n(x)dx = \int_a^b r_n(x)dx$, due to the symmetry of $p_{n+1}^c(x)$ with respect to the midpoint $\frac{a+b}{2}$. Therefore such a quadrature rule with an even n and symmetry in nodes and weights has a degree of accuracy $m \geq n+1$. □

An immediate result following Theorem 57 is summarized as follows.

Theorem 58. *Suppose that the degree of accuracy of a quadrature rule is m, and all $w_k > 0$. Then for $f \in C^{m+1}([a,b])$, the error of the quadrature satisfies*

$$\left| \int_a^b f(x)dx - Q(f) \right| \leq \frac{m+3}{(m+2)!} \max_{c\in[a,b]} \left| f^{(m+1)}(c) \right| (b-a)^{m+2}.$$

Proof. Write $f(x)$ on $[a,b]$ in a Taylor expansion at expansion point a as

$$f(x) = p_m(x) + \frac{f^{(m+1)}(c_x)}{(m+1)!}(x-a)^{m+1}, \tag{8.1}$$

where $a \leq c_x \leq x \leq b$, and $p_m(x) = \sum_{k=0}^{m} \frac{f^{(k)}(a)}{k!}(x-a)^k$ is polynomial of degree $\leq m$. Applying the quadrature rule to both sides of (8.1), we have

$$\begin{aligned} Q(f) &= Q\left(p_m + \frac{f^{(m+1)}(c_x)}{(m+1)!}(x-a)^{m+1} \right) \\ &= Q(p_m) + \frac{1}{(m+1)!} Q\left(f^{(m+1)}(c_x)(x-a)^{m+1} \right) \\ &= \int_a^b p_m(x)dx + \frac{1}{(m+1)!} Q\left(f^{(m+1)}(c_x)(x-a)^{m+1} \right). \end{aligned}$$

Meanwhile, $\int_a^b f(x)dx = \int_a^b p_m(x)dx + \frac{1}{(m+1)!} \int_a^b f^{(m+1)}(c_x)(x-a)^{m+1}dx$. Subtracting $Q(f)$ from $\int_a^b f(x)dx$ and multiplying by $(m+1)!$, we have

$$\begin{aligned} & (m+1)! \left| \int_a^b f(x)dx - Q(f) \right| \qquad (8.2) \\ = & \left| \int_a^b f^{(m+1)}(c_x)(x-a)^{m+1}dx - Q\left(f^{(m+1)}(c_x)(x-a)^{m+1} \right) \right| \\ \leq & \left| \sum_{k=0}^{n} w_k f^{(m+1)}(c_{x_k})(x_k-a)^{m+1} \right| + \left| \int_a^b f^{(m+1)}(c_x)(x-a)^{m+1}dx \right| \\ \leq & \max_{c\in[a,b]} \left| f^{(m+1)}(c) \right| (b-a)^{m+1} \sum_{k=0}^{n} |w_k| + \max_{c\in[a,b]} \left| f^{(m+1)}(c) \right| \int_a^b (x-a)^{m+1}dx \\ = & \max_{c\in[a,b]} \left| f^{(m+1)}(c) \right| (b-a)^{m+2} \left(1 + \frac{1}{m+2} \right), \end{aligned}$$

where we used $\sum_{k=0}^{n} |w_k| = \sum_{k=0}^{n} w_k = b-a$ since all $w_k > 0$. Equivalently, we have $\left| \int_a^b f(x)dx - Q(f) \right| \leq \frac{m+3}{(m+2)!} \max_{c\in[a,b]} \left| f^{(m+1)}(c) \right| (b-a)^{m+2}$. □

8.2 Newton-Cotes rules

The Newton-Cotes quadrature rules are based on interpolating $f(x)$ at equally spaced (or equispaced) nodes. These rules are called *open* if the quadrature nodes

$$x_k = a + (k+1)h,\ (0 \le k \le n) \quad \text{where} \quad h = \frac{b-a}{n+2},$$

and they are called *closed* if

$$x_k = a + kh,\ (0 \le k \le n) \quad \text{where} \quad h = \frac{b-a}{n}.$$

We will consider only one open rule, namely, the midpoint rule, and all others are closed. Composite rules based on closed Newton-Cotes rules on each subinterval are slightly more efficient than those using open rules if $n \ge 1$, because the former can use the same node at the common endpoint of two adjacent subintervals.

By Theorem 57, for both open and closed Newton-Cotes rules, their degree of accuracy is $m \ge n$ if n is odd, and $m \ge n+1$ if n is even due to the symmetry in nodes. One can verify that these are actually equalities, meaning that there is no additional degree of accuracy achieved by Newton-Cotes rules.

To derive the Newton-Cotes quadrature rule formulas, let us begin with the midpoint rule. This rule is open, with only one node $x_0 = \frac{a+b}{2}$, and the corresponding weight is $w_0 = \int_a^b L_0(x)dx = \int_a^b 1dx = b-a$.

The simplest closed rule is the trapezoidal rule ($n = 1$) with nodes $x_0 = a$ and $x_1 = b$, and weights $w_0 = \int_a^b L_0(x)dx = \int_a^b \frac{x-x_1}{x_0-x_1}dx = \frac{b-a}{2}$ and $w_1 = \int_a^b L_1(x)dx = \int_a^b \frac{x-x_0}{x_1-x_0}dx = \frac{b-a}{2}$.

Simpson's rule ($n = 2$) is one of the most widely used closed rules, with nodes $x_0 = a$, $x_1 = \frac{a+b}{2}$ and $x_2 = b$, and weights $w_0 = \int_a^b L_0(x)dx = \int_a^b \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}dx = \frac{b-a}{6}$, $w_1 = \int_a^b L_1(x)dx = \int_a^b \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}dx = \frac{2(b-a)}{3}$, and $w_2 = \int_a^b L_2(x)dx = \int_a^b \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}dx = \frac{b-a}{6}$.

The above basic Newton-Cotes rules are illustrated in Figure 8.1.

We can continue deriving closed rules with more nodes, but the derivation becomes increasingly complex and tedious. Instead, we simply provide a table of these Newton-Cotes rules, their number of nodes $n+1$, the degree of accuracy m, together with their errors $\int_a^b f(x)dx - Q(f)$. Note that alternative error bounds can be obtained directly from Theorems 57 and 58. Though those bounds are not sharp in the constant factor, as they are derived without consideration of the equispacing of nodes and specific values of weights, they do have correct exponents of $(b-a)$ and the order of derivatives of $f(x)$. For example, Boole's rule has $n+1 = 5$ nodes, $x_2 = \frac{a+b}{2}$, and the nodes are symmetric. Therefore by Theorems 57 and 58, the degree of accuracy is $m = 5$, and an error bound is

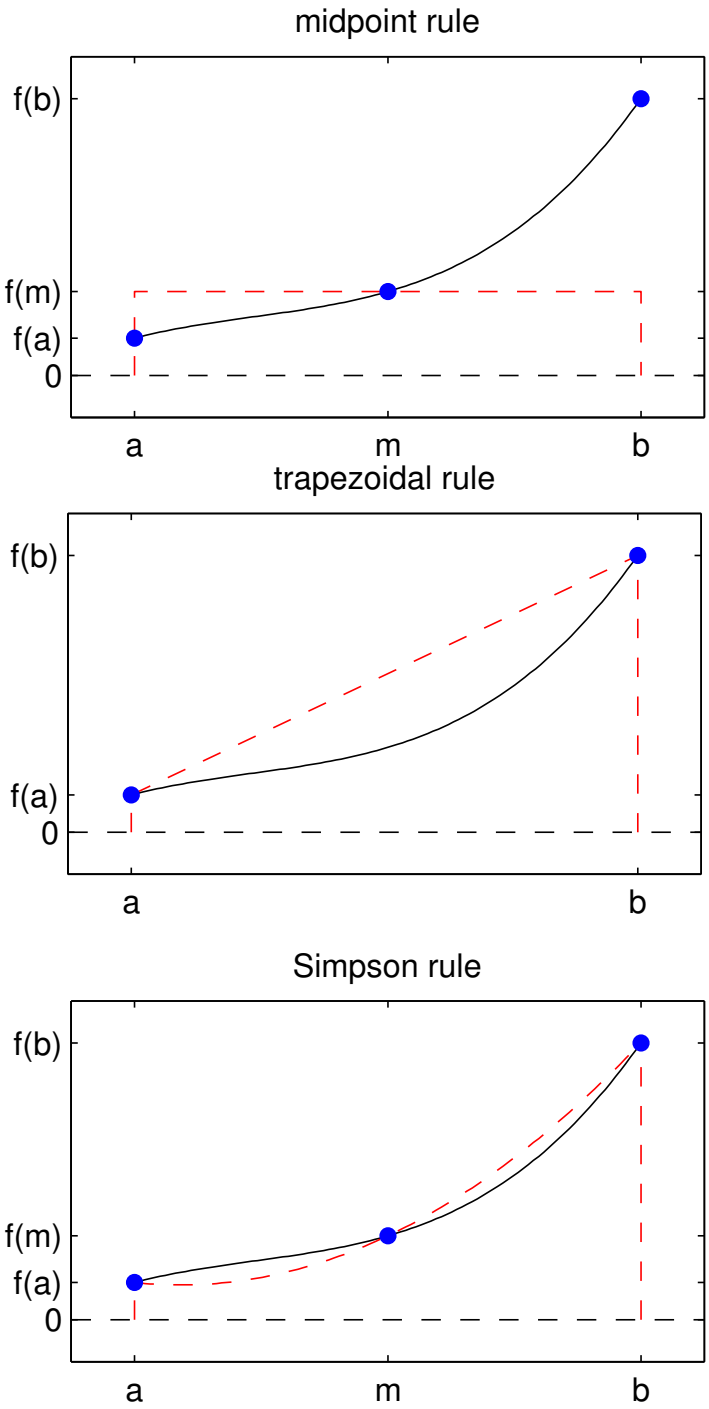

Fig. 8.1: Several low order Newton-Cotes quadrature rules

$\frac{m+3}{(m+2)!}\max_{c\in[a,b]}|f^{(m+1)}(c)|(b-a)^{m+2} = \frac{1}{630}\max_{c\in[a,b]}|f^{(6)}(c)|(b-a)^7$, which has a constant factor much larger than that of the error given in Table 8.1. In other words, if we simply want to construct quadrature rules based on Lagrange interpolation with error bounds of the form $C\max_{c\in[a,b]}|f^{(m+1)}(c)|(b-a)^{m+2}$ (m is the degree of accuracy), there is significant degree of freedom to do so; equispaced nodes are not necessary for this purpose.

Name	n	m	**formula**	**error** $I_f - Q(f)$
Midpoint	0	1	$(b-a)f\left(\frac{a+b}{2}\right)$	$+\frac{(b-a)^3}{24}f^{(2)}(c)$
Trapezoid	1	1	$\frac{b-a}{2}\left[f(a)+f(b)\right]$	$-\frac{(b-a)^3}{12}f^{(2)}(c)$
Simpson	2	3	$\frac{b-a}{6}\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]$	$-\frac{(b-a)^5}{2880}f^{(4)}(c)$
$\frac{3}{8}$**-rule**	3	3	$\frac{b-a}{8}\left[f(a)+3f\left(\frac{2a+b}{3}\right)+3f\left(\frac{a+2b}{3}\right)+f(b)\right]$	$-\frac{(b-a)^5}{6480}f^{(4)}(c)$
Boole	4	5	$\frac{b-a}{90}\left[7f(a)+32f\left(\frac{3a+b}{4}\right)+12f\left(\frac{a+b}{2}\right)\right.$ $\left.+32f\left(\frac{a+3b}{4}\right)+7f(b)\right]$	$-\frac{(b-a)^7}{1935360}f^{(6)}(c)$
–	5	5	$\frac{b-a}{288}\left[19f(a)+75f\left(\frac{4a+b}{5}\right)+50f\left(\frac{3a+2b}{5}\right)\right.$ $\left.+50f\left(\frac{2a+3b}{5}\right)+75f\left(\frac{a+4b}{5}\right)+19f(b)\right]$	$-\frac{11(b-a)^7}{37800000}f^{(6)}(c)$
Weddle	6	7	$\frac{b-a}{840}\left[41f(a)+216f\left(\frac{5a+b}{6}\right)+27f\left(\frac{2a+b}{3}\right)+272\times\right.$ $\left.f\left(\frac{a+b}{2}\right)+27f\left(\frac{a+2b}{3}\right)+216f\left(\frac{a+5b}{6}\right)+41f(b)\right]$	$-\frac{(b-a)^9}{1567641600}f^{(8)}(c)$

Table 8.1: Low order Newton-Cotes quadrature rules

The closed rule with 8 nodes ($n = 7$) has an error comparable to that of Weddle's rule ($n = 6$). The closed rule with $n = 8$ is the first one with negative weights, and the one with $n = 9$ is the last one with all positive weights. Starting from $n = 10$, all closed Newton-Cotes rules have both positive and negative weights, whose maximum magnitude overall increases to infinity as n increases. For the sake of numerical stability, quadrature rules with all weights positive are preferred to those with negative weights, especially those with large negative weights. The following theorem shows why.

Theorem 59. *For a quadrature rule* $Q(f) = \sum_{k=0}^{n} w_k f(x_k)$ *with all* $w_k > 0$*, if* $f, g \in C([a,b])$ *satisfy* $\max_{x\in[a,b]}|f(x)-g(x)| \le \delta$*, then* $|Q(f)-Q(g)| \le \delta(b-a)$.

Proof. Direct evaluation leads to $|Q(f)-Q(g)| = \left|\sum_{k=0}^{n} w_k\left(f(x_k)-g(x_k)\right)\right| \leq \sum_{k=0}^{n}|w_k|\left|f(x_k)-g(x_k)\right| \leq \sum_{k=0}^{n} w_k\delta = \delta(b-a)$. □

The last inequality in the proof would break down if there are negative weights, as $\sum_{k=0}^{n}|w_k| > \sum_{k=0} w_k = b-a$. Moreover, assume that for any given $M>0$, there is a number n, such that an unstable quadrature rule has both positive weights $\{w_k\}$ where $i \in S^+ \subset \{0,1,\ldots,n\}$, and negative weights $\{w_k\}$ where $i \in S^- = \{0,1,\ldots,n\}\setminus S^+$, with $\max_{i\in S^+}|w_k| \geq M$ or $\max_{i\in S^-}|w_k| \geq M$. Let $f,g \in C([a,b])$ be such that $f(x_k)-g(x_k)=\delta>0$ at x_k if $i\in S^+$, and $f(x_k)-g(x_k)=-\delta$ at x_k if $i\in S^-$. Then $|Q(f)-Q(g)| = \left|\sum_{k=0}^{n} w_k\left(f(x_k)-g(x_k)\right)\right| = \left|\sum_{i\in S^+} w_k\delta + \sum_{i\in S^-} w_k(-\delta)\right| = \delta\sum_{k=0}^{n}|w_k| \geq \delta M$.

In other words, such a quadrature of two functions very close to each other may have quite different values. Quadrature rules of this type, e.g., higher order Newton-Cotes, have stability issues and cannot be used.

Example 60. *Approximate $\int_1^3 \ln x dx$, using the midpoint, trapezoidal, Simpson's, Boole's and Weddle's rules. Here, $f(x)=\ln x$, $a=1$ and $b=3$, and the exact integral is $x\ln x - x\big|_1^3 = 3\ln 3 - 2 = 1.29583687$.*

The midpoint rule has $w_0 = b-a = 2$ and $x_0 = \frac{a+b}{2} = 2$. Therefore $Q(f) = w_0 f(x_0) = 2\ln 2 = 1.38629436$.

The trapezoidal rule has $w_0 = w_1 = \frac{b-a}{2} = 1$, and $x_0 = 1$, $x_1 = 3$. It follows that $Q(f) = w_0 f(x_0) + w_1 f(x_1) = \ln 3 = 1.09861229$.

Simpson's rule has $w_0 = w_2 = \frac{b-a}{6} = \frac{1}{3}$ and $w_1 = \frac{4(b-a)}{6} = \frac{4}{3}$, and $x_0 = 1$, $x_1 = 2$ and $x_2 = 3$. Consequently, $Q(f) = w_0 f(x_0) + w_1 f(x_1) + w_2 f(x_2) = \frac{4}{3}\ln 2 + \frac{1}{3}\ln 3 = 1.29040034$.

Boole's rule has $w_0 = w_4 = \frac{7(b-a)}{90}$, $w_1 = w_3 = \frac{32(b-a)}{90}$, $w_2 = \frac{12(b-a)}{90}$, and $x_0 = 1$, $x_1 = 1.5$, $x_2 = 2$, $x_3 = 2.5$ and $x_4 = 3$. It follows that $Q(f) = \sum_{k=0}^{4} w_k f(x_k) = 1.29564976$. Similarly, Weddle's rule gives $\sum_{k=0}^{6} w_k f(x_k) = 1.29582599$, using the nodes $x_k = a + k\frac{b-a}{6}$ $(0 \leq k \leq 6)$ and the weights in Table 8.1. For this problem, higher order rules give more accurate approximations.

Example 61. *Use Newton-Cotes rules to approximate $I_1 = \int_{-0.5}^{0.5} \frac{1}{1+36x^2} dx = 0.4163485908$ and $I_2 = \int_{0.5}^{1.5} \frac{1}{1+36x^2} dx = 0.0351822222$. The quadrature values are summarized as follows, where n is a multiple of 4.*

In this example, the quadrature seems to converge to I_2 but fail to converge to I_1 as n increases. This is because the Lagrange interpolation polynomial $p_n(x)$ of $f(x)$ based on $n+1$ equispaced nodes approximates $f(x)$ accurately on $[0.5, 1.5]$ for $n \approx 20$, but it does not approximate $f(x)$ well on $[-0.5, 0.5]$ for all n. The failure of convergence is precisely the well-known *Runge phenomenon.*

n	I_1	I_2
0	1.0000000000	0.0270270270
4	0.3676923077	0.0352724944
8	0.3691018013	0.0351827526
12	0.3598308365	0.0351822243
16	0.3337916510	0.0351822222
20	0.2811316793	0.0351822222

Table 8.2: Newton-Cotes quadrature for approximating $\int_{-0.5}^{0.5} \frac{dx}{1+36x^2}$ and $\int_{0.5}^{1.5} \frac{dx}{1+36x^2}$

To make sure that the quadrature converges to the integral, one approach is to use composite rules based on Newton-Cotes, discussed in the next section.

8.3 Composite rules

Assume that the degree of accuracy of a Newton-Cotes rule is m, $f(x) \in C^{m+2}$ on any closed interval in its domain, and $f^{(m+2)}$ is uniformly bounded on all such closed intervals. Under these assumptions, we see from the absolute errors of Newton-Cotes rules in Table 8.1 that these errors tend to be smaller on a shorter interval. In addition, if $f(x)$ does not change sign and is bounded away from zero and bounded in modulus on $[a, b]$, then $\int_a^b f(x)dx$ is typically and roughly proportional to the length of the interval $b - a$. It follows that the *relative* error of Newton-Cotes rules is $\frac{C_1(b-a)^{m+2}}{I_f} \approx \frac{C_1(b-a)^{m+2}}{C_2(b-a)} = C_3(b-a)^{m+1}$. Therefore, the relative error will usually also decrease on a shorter interval.

The above informal discussion justifies the use of composite quadrature rules: divide the entire interval $[a, b]$ into sufficiently many subintervals, apply regular quadrature rules such as Newton-Cotes on each subinterval and sum up the quadratures. For example, we may divide $[a, b]$ into ℓ subintervals $[a, a + h], [a + h, a + 2h], \ldots, [b - h, b]$ with $h = \frac{b-a}{\ell}$, and use the midpoint rule on each subinterval. The composite midpoint rule is therefore

$$Q_{CM}(f) = hf\left(a + \frac{1}{2}h\right) + hf\left(a + \frac{3}{2}h\right) + \ldots + hf\left(b - \frac{1}{2}h\right) = h\sum_{k=0}^{\ell-1} f\left(a + \left(k + \frac{1}{2}\right)h\right).$$

This composite rule is illustrated in Figure 8.2.

Similarly, we may also approximate the integral on each subinterval by the trapezoidal rule. The composite trapezoidal rule is

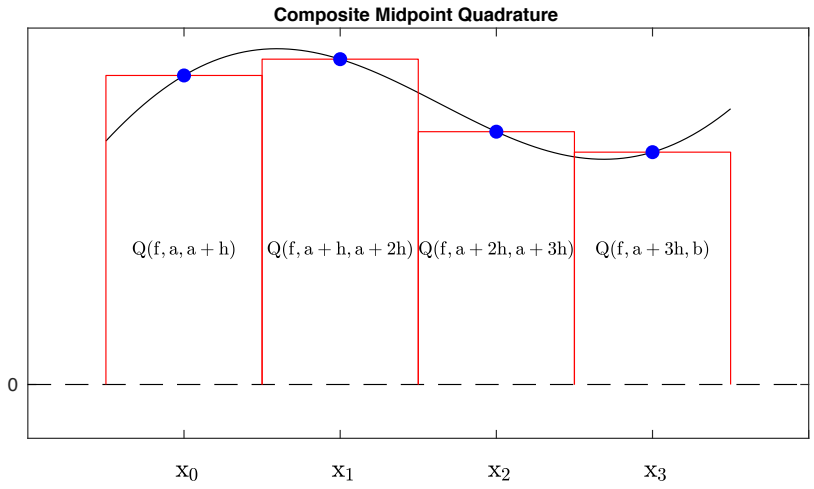

Fig. 8.2: Composite midpoint quadrature rule

$$Q_{CT}(f) = \frac{h}{2}\left(f(a) + f(a+h)\right) + \frac{h}{2}\left(f(a+h) + f(a+2h)\right) + \ldots + \frac{h}{2}\left(f(b-h) + f(b)\right)$$
$$= \frac{h}{2}\left(f(a) + 2f(a+h) + \ldots + 2f(b-h) + f(b)\right) = \frac{h}{2}\left(f(a) + 2\sum_{k=1}^{\ell-1} f(a+kh) + f(b)\right).$$

The composite Simpson's rule is as follows. If we choose to evaluate $f(x)$ at the end points (and not midpoints) of subintervals only, the subinterval count ℓ must be even as the Simpson's rule is applied to two adjacent subintervals.

$$\begin{aligned} Q_{CS}(f) &= \frac{2h}{6}\left(f(a) + 4f(a+h) + f(a+2h)\right) + \frac{2h}{6}\left(f(a+2h) + 4f(a+3h) + f(a+4h)\right) \\ &\quad + \ldots + \frac{2h}{6}\left(f(b-2h) + 4f(b-h) + f(b)\right) \\ &= \frac{h}{3}\left(f(a) + 4\sum_{k=1}^{\ell/2} f(a+(2k-1)h) + 2\sum_{k=1}^{\ell/2-1} f(a+2kh) + f(b)\right). \end{aligned}$$

Higher-order composite quadrature rules can be constructed, e.g., based on Boole's and Weddle's rules. Similar to the practice of the composite Simpson's rule, the number of subintervals ℓ should be a multiple of 4 and 6, respectively.

How to estimate the error of composite quadrature rules? Consider the composite trapezoidal rule, for example. From Table 8.1, note that the error on each subinterval is bounded by $\frac{h^3}{12}\max|f^{(2)}(c)|$, and therefore the total error is bounded by $\ell\frac{h^3}{12}\max|f^{(2)}(c)| = \frac{h^2}{12}(h\ell)\max|f^{(2)}(c)| = \frac{b-a}{12}\max|f^{(2)}(c)|h^2$

(note that $h\ell = b - a$). In general, if the degree of accuracy of a regular Newton-Cotes rule is m, the error of the corresponding composite rule is bounded by $C(b-a)\max|f^{(m+1)}(c)|h^{m+1}$. In particular, the error of composite Simpson's, Boole's and Weddle's rules is bounded by $(b-a)$ times $\mathcal{O}(h^4)$, $\mathcal{O}(h^6)$ and $\mathcal{O}(h^8)$, respectively. These orders of accuracy are usually sufficient for applications.

composite rules	error bound
Midpoint	$\frac{\max\|f^{(2)}(c)\|}{24}(b-a)h^2$
Trapezoid	$\frac{\max\|f^{(2)}(c)\|}{12}(b-a)h^2$
Simpson	$\frac{\max\|f^{(4)}(c)\|}{90}(b-a)h^4$
Boole	$\frac{8\max\|f^{(6)}(c)\|}{945}(b-a)h^6$
Weddle	$\frac{9\max\|f^{(8)}(c)\|}{1400}(b-a)h^8$

Table 8.3: Composite Newton-Cotes rule error bounds ($h = \frac{b-a}{\ell}$, $\ell+1$ equispaced nodes)

Example 62. *Approximate* $\int_1^{2.2} \ln x dx = 0.534606$ *by composite trapezoidal and Simpson's rules based on* 6 *subintervals. Here,* $a = 1$, $b = 2.2$, *and the end points of the* 6 *subintervals are* 1, 1.2, 1.4, 1.6, 1.8, 2.0 *and* 2.2. *Composite trapezoidal rule gives* $\frac{h}{2}(f(1) + 2f(1.2) + 2f(1.4) + 2f(1.6) + 2f(1.8) + 2f(2.0) + f(2.2)) = 0.532792$ (*two digits of accuracy*), *and our composite Simpson's rule leads to* $\frac{h}{3}(f(1) + 4f(1.2) + 2f(1.4) + 4f(1.6) + 2f(1.8) + 4f(2.0) + f(2.2)) = 0.534591$, *more accurate than the composite trapezoidal quadrature.*

Example 63. *Consider* $I_f = \int_{-0.6}^{0.6} \frac{1}{1+36x^2} dx = 0.4332831588188253$, *for which regular high-order Newton-Cotes rules have difficulties approximating. We divide* $[-0.6, 0.6]$ *into* $\ell = 192$ *subintervals of the same length* $h = \frac{0.6-(-0.6)}{192} = \frac{1}{160}$, *and use composite trapezoidal, Simpson's, Boole's and Weddle's rules to approximate* I_f. *Since* $\ell = 192$ *is a multiple of* 2, 4 *and* 6, *all these composite rules can be used. The results are summarized in the following table. The correct digits in each result are underlined. Clearly, with a sufficiently small subinterval of length* h, *higher-order composite rules give more accurate approximations.*

Finally, we point out the composite trapezoidal rule works particularly well for numerically integrating periodic functions. In particular, the error of the quadrature decreases *exponentially* with the number of subintervals.

composite rules	$Q(f)$
trapezoid	0.4332817156597703
Simpson	0.4332831587192119
Boole	0.4332831588187392
Weddle	0.4332831588188242

Table 8.4: Composite Newton-Cotes quadrature for approximating $\int_{-0.6}^{0.6} \frac{1}{1+36x^2}dx$

Theorem 64. *Suppose that the real function $f \in C^\infty([a,b])$ with period $T = b-a$ can be extended to the complex plane, $f(z)$ is analytic in the horizontal strip $-\mu < \mathrm{Im}(z) < \mu$ for some $\mu > 0$, and $|f(z)| \le M$ in this strip. Let $Q^\ell_{CT}(f)$ be the composite trapezoidal quadrature of f on $[a,b]$ based on ℓ subintervals of length $h = \frac{b-a}{\ell}$. Then $\left|Q^\ell_{CT}(f) - \int_a^b f(x)dx\right| \le \frac{2TM}{e^{2\pi\mu\ell/T}-1} \approx 2TMe^{-\frac{2\pi\mu\ell}{T}}$.*

Example 65 (Poisson's elliptic integral, 1827)**.** *Use the composite trapezoidal rule to approximate $I = \frac{1}{2\pi}\int_0^{2\pi}\sqrt{1-0.36\sin^2\theta}d\theta = 0.9027799277721939$. The results are summarized as follows. Whenever the number of subintervals ℓ increases by 4, we get roughly 2 more digits of accuracy.*

subinterval		subinterval	
# ℓ	$Q^\ell_{CT}(f)$	# ℓ	$Q^\ell_{CT}(f)$
8	0.9027692569068708	12	0.9027798586495662
16	0.9027799272275734	20	0.9027799277674322
24	0.9027799277721495	28	0.9027799277721936

Table 8.5: Composite trapezoidal rule for approximating $\frac{1}{2\pi}\int_0^{2\pi}\sqrt{1-0.36\sin^2\theta}d\theta$

8.4 Clenshaw-Curtis quadrature

Quadrature rules based on equispaced nodes seem natural, as one would wonder what we can achieve if non-equispaced nodes are used. In fact, using carefully chosen non-equispaced nodes may develop highly efficient and accurate quadrature rules, which converge to the integrals of $f \in C^\infty([a,b])$ *exponentially* as the number of nodes increases, asymptotically outperforming all previously discussed

quadrature rules with errors on the order of $\mathcal{O}(h^p)$ for any integer $p > 0$. We shall discuss two rules: Clenshaw-Curtis and Gauss.[1]

Clenshaw-Curtis Quadrature. The Clenshaw-Curtis quadrature for $\int_a^b f(x)dx$ is $Q_{2C}^n(f) = \int_a^b p_n(x)dx = \sum_{k=0}^n w_k f(x_k)$, where $p_n(x)$ is the Chebyshev interpolant for $f(x)$, and the quadrature nodes $x_k = -\cos\left(\frac{k\pi}{n}\right)\frac{b-a}{2} + \frac{a+b}{2}$ are the Chebyshev points. For an odd n, the weights $w_k = \int_a^b L_k(x)dx$ satisfy

$$w_k = \begin{cases} \frac{b-a}{2n^2} & k = 0, n \\ \frac{b-a}{n}\left\{1 - \sum_{j=1}^{\frac{n-1}{2}} \frac{2}{4j^2-1}\cos\left(\frac{2kj\pi}{n}\right)\right\} & 1 \le k \le n-1 \end{cases}, \tag{8.3}$$

and for an even n,

$$w_k = \begin{cases} \frac{b-a}{2(n^2-1)} & k = 0, n \\ \frac{b-a}{n}\left\{1 - \frac{(-1)^k}{n^2-1} - \sum_{j=1}^{\frac{n}{2}-1} \frac{2}{4j^2-1}\cos\left(\frac{2kj\pi}{n}\right)\right\} & 1 \le k \le n-1 \end{cases}. \tag{8.4}$$

The first few Clenshaw-Curtis quadrature rules are summarized in Table 8.6. The degree of accuracy and upper bound on the errors of these quadrature rules are similar to those of the Newton-Cotes rules (see Theorems 57 and 58).

n	**Clenshaw-Curtis rules**
1	$\frac{b-a}{2}\left[f(a) + f(b)\right]$
2	$\frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$
3	$\frac{b-a}{18}\left[f(a) + 9f\left(\frac{3a+b}{4}\right) + 9f\left(\frac{a+3b}{4}\right) + f(b)\right]$
4	$\frac{b-a}{30}\left[f(a) + 8f\left(\frac{2+\sqrt{2}}{4}a + \frac{2-\sqrt{2}}{4}b\right) + 12f\left(\frac{a+b}{2}\right) + 8f\left(\frac{2-\sqrt{2}}{4}a + \frac{2+\sqrt{2}}{4}b\right) + f(b)\right]$
6	$\frac{b-a}{630}\left[9f(a) + 80f\left(\frac{2+\sqrt{3}}{4}a + \frac{2-\sqrt{3}}{4}b\right) + 144f\left(\frac{3a+b}{4}\right) + 164f\left(\frac{a+b}{2}\right) + 144f\left(\frac{a+3b}{4}\right)\right.$ $\left. +80f\left(\frac{2-\sqrt{3}}{4}a + \frac{2+\sqrt{3}}{4}b\right) + 9f(b)\right]$

Table 8.6: Low order Clenshaw-Curtis quadrature

Though Chenshaw-Curtis could be used to construct composite quadrature rules as Newton-Cotes, this is not the most efficient way to make full use of its

1 More details of these quadrature rules can be found in *Approximation Theory and Approximation Practice* by Lloyd N. Trefethen, SIAM, 2013.

approximation power. Instead, we should use a high-order Clenshaw-Curtis rule, whose nodes and weights can be computed very efficiently.

Convergence. Simply speaking, the Clenshaw-Curtis quadrature converges to the integral exponentially with the number of nodes if the integrand $f(x)$ is analytic. The following results are presented for reference purposes only.

Suppose that $f^{(k)}$ $(0 \le k \le m-1)$ are absolutely continuous, and $f^{(m+1)}$ is such that $V = \int_{-1}^{1} \left| \frac{f^{(m+1)}}{\sqrt{1-x^2}} \right| dx < \infty$. Let $Q_{2C}^n(f)$ be the $(n+1)$-point Clenshaw-Curtis quadrature for $I_f = \int_{-1}^{1} f(x)dx$. For sufficiently large n, $|I_f - Q_{2C}^n(f)| \le \frac{32V}{15\pi m(2n+1-m)^m}$. In addition, if $f \in C^\infty([-1,1])$ and can be analytically continued to the ellipse $E_\rho = \frac{\rho e^{i\theta} + \rho^{-1} e^{-i\theta}}{2}$ $(\rho > 1,\ \theta \in [0, 2\pi))$ in the complex plane, and $|f(z)| \le M$ inside E_ρ, then $|I_f - Q_{2C}^n(f)| \le \frac{64M\rho}{15(\rho^2-1)\rho^n}$.

An efficient implementation of the Clenshaw-Curtis quadrature based on the Fast Fourier Transform (FFT) is as follows.

```
function Q = clenshawcurtis(fun,a,b,n)
% (n+1)—pt Clenshaw—Curtis quadrature for integral(f,a,b)

% Compute Chebyshev points and evaluate f at these points:
x = —cos(pi*(0:n)'/n)*(b—a)/2+(a+b)/2;
fx = feval(fun,x)/(2*n);

% apply Fast Fourier Transform:
g = real(fft(fx([1:n+1 n:—1:2])));

% compute Chebyshev coefficients:
coef = [g(1); g(2:n)+g(2*n:—1:n+2); g(n+1)];

% compute vector with weights:
w = zeros(n+1,1);
w(1:2:end) = 2./(1—(0:2:n).^2);

% compute result:
Q = w'*coef*(b—a)/2;
```

The cost for computing all weights $\{w_k\}$ for $(n+1)$-point Clenshaw-Curtis is $\mathcal{O}(n \log n)$, almost linear with n, thanks to the use of FFT. Also, whenever n doubles, a half of the new nodes coincide with the old ones, such that function evaluations at these nodes need not be done repeatedly.

Example 66. *Approximate* $\int_1^3 \ln x dx$ *by the Clenshaw-Curtis quadrature using* $n+1 = 5$ *nodes* $x_k = -\cos\left(\frac{k\pi}{n}\right)\frac{b-a}{2} + \frac{a+b}{2}$, *where* $a = 1$, $b = 3$ *and* $n = 4$. *Direct evaluation gives* $x_0 = 1$, $x_1 = 2 - \frac{\sqrt{2}}{2}$, $x_2 = 2$, $x_3 = 2 + \frac{\sqrt{2}}{2}$, *and* $x_4 = 3$. *From*

(8.4), *we have* $w_0 = w_4 = \frac{1}{15}$, $w_1 = w_3 = \frac{8}{15}$ *and* $w_2 = \frac{12}{15}$. *The Clenshaw-Curtis quadrature is therefore* $Q^4_{2C}(f) = \sum_{k=0}^{n} w_k f(x_k) = 1.2958988$ *with* 5 *digits of accuracy. The quadrature with* $n+1 = 20$ *nodes is accurate to machine precision.*

8.5 Gauss quadrature

Another type of quadrature that converges exponentially for analytic integrand is the Gauss quadrature. Though Gauss quadrature is much more well-known than Clenshaw-Curtis, they are comparable in many aspects.

Consider the quadrature $I_f = \int_a^b f(x)\rho(x)dx$, where $\rho(x) > 0$ is a weight function. A quadrature $Q(f) = \sum_{k=0}^{n} w_k f(x_k)$ (note that it does not evaluate $\rho(x)$ anywhere) is an $(n+1)$-node *Gauss quadrature* if its degree of accuracy is $2n+1$. The nodes and weights of Gauss quadrature can be constructed directly: we set up the nonlinear system of equations $\int_a^b x^\ell \rho(x)dx = \sum_{k=0}^{n} w_k x_k^\ell$ for $0 \le \ell \le 2n+1$, and solve for all nodes and weights, for example, by Newton's method. This approach is fine for small n, but not the method of choice for large n, because each iteration of Newton's method involves solution to a linear system of equations involving $2n+2$ unknowns, taking $\mathcal{O}(n^3)$ flops per iteration.

Example 67. *Determine the 2-point* $(n = 1)$ *Gauss quadrature rule for* $\rho(x) = 1$ *on* $[-1, 1]$ *by hand. Let the quadrature rule be* $Q = w_0 f(x_0) + w_1 f(x_1)$, *which is exact for polynomial integrand of degree* $\le 2n+1 = 3$. *Therefore, we have the following equations for the unknowns* w_0, w_1, x_1 *and* x_2.

$$w_0 + w_1 = \int_{-1}^{1} 1dx = 2 \qquad w_0 x_0 + w_1 x_1 = \int_{-1}^{1} x dx = 0$$

$$w_0 x_0^2 + w_1 x_1^2 = \int_{-1}^{1} x^2 dx = \frac{2}{3} \qquad w_0 x_0^3 + w_1 x_1^3 = \int_{-1}^{1} x^3 dx = 0$$

This system of nonlinear equations seem hard to solve by hand. However, we note that the quadrature rule should be symmetric with respect to the origin; that is, $x_0 = -x_1$ *and* $w_0 = w_1$. *This observation easily leads to* $w_0 = w_1 = 1$, $x_0 = -\frac{\sqrt{3}}{3}$ *and* $x_1 = \frac{\sqrt{3}}{3}$. *We can determine the 3-point Gauss quadrature similarly.*

Legendre Polynomials. We focus on the most common case where $\rho(x) \equiv 1$. For $f(x) \in C^{n+1}([a, b])$, let us consider the Lagrange interpolation $p_n(x) = \sum_{k=0}^{n} f(x_k)L_k(x)$, such that $f(x) = p_n(x) + \frac{1}{(n+1)!} f^{(n+1)}(c_x)\omega_{n+1}(x)$, where $c_x \in (a, b)$ and $\omega_{n+1}(x) = (x - x_0)(x - x_1)\cdots(x - x_n)$. It follows that

$$\begin{aligned}\int_a^b f(x)dx &= \sum_{k=0}^{n} f(x_k)\int_a^b L_k(x)dx + \frac{1}{(n+1)!}\int_a^b f^{(n+1)}(c_x)\omega_{n+1}(x)dx \qquad (8.5)\\ &= \sum_{k=0}^{n} w_k f(x_k) + \frac{1}{(n+1)!}\int_a^b f^{(n+1)}(c_x)\omega_{n+1}(x)dx = Q(f) + R_n,\end{aligned}$$

where $R_n = \frac{1}{(n+1)!}\int_a^b f^{(n+1)}(c_x)\omega_{n+1}(x)dx$. For the quadrature to have a degree of accuracy $2n+1$, R_n must be zero for all polynomial integrand f of degree $\leq 2n+1$, whose $(n+1)$-st derivative is a polynomial of degree $\leq n$. Therefore, if the nodes $\{x_k\}_{k=0}^n$ are such that $\omega_{n+1}(x)$ with zeros $\{x_k\}_{k=0}^n$ is orthogonal to all polynomials of degree $\leq n$ under the inner product $(u,w) = \int_a^b u(x)w(x)dx$, then the degree of accuracy is $2n+1$. Such a polynomial $\omega_{n+1}(x)$ is the Legendre polynomial of degree $n+1$, up to a scaling factor.

Let us again consider the standard interval $[-1,1]$. *Legendre polynomials* are defined as $P_0(x) = 1$, $P_1(x) = x$, and $P_{n+1}(x) = \frac{2n+1}{n+1}xP_n(x) - \frac{n}{n+1}P_{n-1}(x)$ for $n \geq 1$. They satisfy $P_n(1) = 1$ and $P_n(-1) = (-1)^n$ for all n, and they are orthogonal in the sense that $\int_{-1}^1 P_m(x)P_n(x)dx = \frac{2}{2n+1}\delta_{mn}$. In addition, $P_n(x)$ has precisely n simple roots in $(-1,1)$.

Let $\{x_k\}_{k=0}^n$ be the roots of $P_{n+1}(x)$, which are the nodes for the $(n+1)$-point Gauss quadrature on $[-1,1]$. Let $L_k(x)$ be the Lagrange basis associated with x_k. It can be shown that the corresponding weight is

$$w_k = \int_{-1}^1 L_k(x)dx = \frac{2(1-x_k^2)}{\big((n+1)P_n(x_k)\big)^2} > 0.$$

It is not difficult to following the above discussion to derive the first few Legendre polynomials and then compute their roots (quadrature nodes) and weights. The results are summarized in the following table.

In particular, the 1-, 2-, and 3-point Gauss quadrature are, respectively

$$\begin{aligned}Q_{G1} &= (b-a)f\left(\tfrac{a+b}{2}\right),\\ Q_{G2} &= \tfrac{b-a}{2}\left[f\left(\tfrac{3+\sqrt{3}}{6}a + \tfrac{3-\sqrt{3}}{6}b\right) + f\left(\tfrac{3-\sqrt{3}}{6}a + \tfrac{3+\sqrt{3}}{6}b\right)\right],\\ Q_{G3} &= \tfrac{b-a}{18}\left[5f\left(\tfrac{5+\sqrt{15}}{10}a + \tfrac{5-\sqrt{15}}{10}b\right) + 8f\left(\tfrac{a+b}{2}\right) + 5f\left(\tfrac{5-\sqrt{15}}{10}a + \tfrac{5+\sqrt{15}}{10}b\right)\right].\end{aligned}$$

Computation of nodes and weights for large n. Unlike Clenshaw-Curtis, there are no elementary expressions for the nodes and weights for the Gauss quadrature. Though the values of these quantities can be found exactly for $n \leq 4$, we need to compute them for larger n to make full use of the Gauss quadrature. The mathematics needed for this purpose is advanced and shall not be discussed, but we can still present the major results for the quadrature on $[a,b]$.

n	**nodes** x_i	**weights** w_i
0	0	2
1	$-\frac{\sqrt{3}}{3}$, $\frac{\sqrt{3}}{3}$	1, 1
2	$-\sqrt{\frac{3}{5}}$, 0, $\sqrt{\frac{3}{5}}$	$\frac{5}{9}$, $\frac{8}{9}$, $\frac{5}{9}$
3	$-\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$, $-\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}$,	$\frac{18-\sqrt{30}}{36}$, $\frac{18+\sqrt{30}}{36}$,
	$\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}$, $\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18+\sqrt{30}}{36}$, $\frac{18-\sqrt{30}}{36}$
4	$-\frac{1}{3}\sqrt{5+2\sqrt{\frac{10}{7}}}$, $-\frac{1}{3}\sqrt{5-2\sqrt{\frac{10}{7}}}$, 0	$\frac{322-13\sqrt{70}}{900}$, $\frac{322+13\sqrt{70}}{900}$, $\frac{128}{255}$,
	$\frac{1}{3}\sqrt{5-2\sqrt{\frac{10}{7}}}$, $\frac{1}{3}\sqrt{5+2\sqrt{\frac{10}{7}}}$	$\frac{322+13\sqrt{70}}{900}$, $\frac{322-13\sqrt{70}}{900}$

Table 8.7: Nodes and weights of low order Gauss quadrature

Theorem 68. *Let* $T_{n+1} = \begin{pmatrix} 0 & \beta_1 & & & \\ \beta_1 & 0 & \beta_2 & & \\ & \beta_2 & \ddots & \ddots & \\ & & \ddots & 0 & \beta_n \\ & & & \beta_n & 0 \end{pmatrix}$ *with* $\beta_k = \frac{1}{2\sqrt{1-(2k)^{-2}}}$. *Let* $T_{n+1} = VDV^T$ *be an eigenvalue decomposition, where* $V = [v_1, \ldots, v_{n+1}]$ *and* $D = \operatorname{diag}(\lambda_1, \ldots, \lambda_{n+1})$. *Then the* $(n+1)$*-point Gauss quadrature nodes are* $x_k = \lambda_{k+1}\frac{b-a}{2} + \frac{a+b}{2}$, *and the weights are* $w_k = (b-a)\left(e_1^T v_{k+1}\right)^2$ $(0 \le k \le n)$.

Since the computation of all eigenpairs of the real symmetric matrix T_{n+1} by the QR iteration takes $\mathcal{O}(n^2)$ flops, this is an upper bound on the cost for computing the nodes and weights for Gauss quadrature. In recent years, new algorithms have been proposed to compute these quantities in $\mathcal{O}(n)$ flops, making it essentially as efficient as Clenshaw-Curtis. Nevertheless, unlike Clenshaw-Curtis, whenever n changes, all nodes change for Gauss quadrature, so that the evaluation of the integrand need be done at all the new nodes.

Convergence. Similar to Clenshaw-Curtis, Gauss quadrature also exhibit exponential convergence to the integral for analytic integrand $f(x)$. The following results are also presented for reference purposes.

Assume that $f^{(k)}$ $(0 \le k \le m-1)$ are absolutely continuous, and $f^{(m)}$ is of bounded variation with $V = \int_{-1}^{1} \left|f^{(m+1)}\right| dx < \infty$. Let $Q_G^n(f)$ be the $(n+1)$-point Gauss quadrature for $I_f = \int_{-1}^{1} f(x)dx$. For all $n \ge \frac{m}{2}$, $|I_f - Q_G^n(f)| \le \frac{32V}{15\pi m(2n+1-m)^m}$. In addition, if $f \in C^\infty([-1,1])$ and can be analytically contin-

ued to the ellipse $E_\rho = \frac{\rho e^{i\theta} + \rho^{-1} e^{-i\theta}}{2}$ ($\rho > 1$, $\theta \in [0, 2\pi)$) in the complex plane, and $|f(z)| \le M$ inside E_ρ, then $|I_f - Q_G^n(f)| \le \frac{64M}{15(\rho^2-1)\rho^{2n}}$.

In other words, Gauss and Clenshaw-Curtis quadrature exhibit similar convergence behavior for integrands with limited regularity, and Clenshaw-Curtis may need up to twice as many nodes as Gauss to achieve the same accuracy for an analytic integrand. In practice, the difference in convergence rate between the two quadrature is even less, especially for a relatively small number of nodes.

We have a MATLAB code of the Gauss quadrature based on Theorem 68.

```
function Q = gausslg(fun,a,b,n)
% (n+1)-pt Gauss-Legendre quadrature for integral(fun,a,b)
m = (n+2)/2;     xg = zeros(n+1,1);     wg = zeros(n+1,1);
for i = 1 : m
    z = cos(pi*(i-0.25)/(n+1+0.5));
    while true
        p1 = 1;          p2 = 0;
        for j = 1 : n+1
            p3 = p2;     p2 = p1;
            p1 = ((2*j-1)*z*p2-(j-1)*p3)/j;
        end
        pp = (n+1)*(z*p1-p2)/(z*z-1);
        z1 = z;          z = z1-p1/pp;
        if abs(z-z1) <= eps/2,    break;      end
    end
    xg(i) = -z;                  xg(n+2-i) = z;
    wg(i) = 2/((1-z*z)*pp*pp);   wg(n+2-i) = wg(i);
end
xg = (b+a)/2+(b-a)/2*xg;         wg = (b-a)/2*wg;
Q = wg'*fun(xg);
```

Example 69. *Approximate* $\int_1^3 \ln x dx$ *by Gauss quadrature with* $n+1 = 3$ *nodes. We first compute the original nodes and weights for the quadrature on* $[-1, 1]$. *Note that the Legendre polynomial* $P_3(x) = \frac{1}{2}(5x^3 - 3x)$, *whose roots are* $x_0 = -\sqrt{\frac{3}{5}}$, $x_1 = 0$ *and* $x_2 = \sqrt{\frac{3}{5}}$. *The weights are* $w_k = \frac{2(1-x_k^2)}{\left(3P_2(x_k)\right)^2}$; *that is* $w_0 = w_2 = \frac{5}{9}$ *and* $w_1 = \frac{8}{9}$. *Therefore, on* $[1, 3]$, *we need* $x_k \leftarrow x_k \frac{b-a}{2} + \frac{b+a}{2}$ *and* $w_k \leftarrow w_k \frac{b-a}{2}$; *that is,* $x_0 = 2 - \sqrt{\frac{3}{5}}$, $x_1 = 2$ *and* $x_2 = 2 + \sqrt{\frac{3}{5}}$, $w_0 = w_2 = \frac{5}{9}$ *and* $w_1 = \frac{8}{9}$. *The Gauss quadrature is* $Q_G^2(f) = \sum_{k=0}^2 w_k f(x_k) = 1.2960060$ *with* 3 *digits of accuracy. Using the code provided above, we find that Gauss quadrature with* $n + 1 = 14$ *nodes is accurate to machine precision.*

Example 70. *Approximate* $\int_a^b e^{-5x}\sin\frac{1}{x}\sin\frac{1}{\sin\frac{1}{x}}dx = 0.02561655631847027$, *where* $a = 0.1593$ *and* $b = 0.3182$, *using* MATLAB*'s integral, composite Simpson's rule, classical Gauss* (*code provided above, complexity* $\mathcal{O}(n^2)$), *new Gauss* (*complexity* $\mathcal{O}(n)$), *and FFT-based Clenshaw-Curtis* (*code provided in previous section, complexity* $\mathcal{O}(n\log n)$), *to machine precision. Also, since the integrand is highly oscillatory near the two end points, but changes very mildly in the middle, we divide* $[a, b]$ *into three subintervals* $[0.1593, 0.16]$, $[0.16, 0.317]$ *and* $[0.317, 0.3182]$ *and apply these quadrature on each subinterval to improve efficiency. The performance is summarized in Table 8.8, where the numbers of nodes are obtained by trial and error and are close to minimal to achieve* 15*-digit accuracy accuracy. In practice, adaptive quadrature need be used to gradually increase the number of nodes wherever* $f(x)$ *changes rapidly to guarantee accuracy.*

Quadrature	**# of nodes**	**time (secs)**
MATLAB's `integral`	–	0.002337
Comp. Simpson $[a, b]$	16778241	1.150686
Comp. Simpson 3 subintervals	4727811	0.294279
Classical Gauss $[a, b]$	8193	0.569958
Classical Gauss 3 subintervals	2563	0.034397
New Gauss $[a, b]$	8193	0.001391
New Gauss 3 subintervals	2563	0.000693
FFT Clenshaw-Curtis $[a, b]$	12289	0.001402
FFT Clenshaw-Curtis 3 subintervals	2563	0.000446

Table 8.8: Comparison of several quadratures in MATLAB for approximating $\int_{0.1593}^{0.3182} f(x)dx$

Clearly, applying quadrature rules on three subintervals needs fewer nodes. Both Gauss and Clenshaw-Curtis need much fewer nodes than composite Simpson. New Gauss quadrature rule with complexity $\mathcal{O}(n)$ *and FFT-based Clenshaw-Curtis with complexity* $\mathcal{O}(n\log n)$ *are the fastest algorithms.*

Newton-Cotes, Clenshaw-Curtis, or Gauss? What do we choose quadrature rules among Newton-Cotes, Clenshaw-Curtis, and Gauss? To achieve convergence, we have to use composite Newton-Cotes rules because high order Newton-Cotes rules are unstable (have both positive and negative weights going to infinity as the number of quadrature nodes grows). By contrast, Clenshaw-Curtis and Gauss quadrature have all positive weights and are guaranteed to converge to the integral as the number of quadrature node increases.

Though Clenshaw-Curtis and Gauss quadrature could be used to construct composite rules, their full power of convergence is only enabled in high-order rules. Fortunately, the nodes and weights of Clenshaw-Curtis and Gauss can be computed in $\mathcal{O}(n \log n)$ (based on FFT) and $\mathcal{O}(n)$ (recent developments) operations, respectively. If the integrand $f(x)$ is analytic and can be evaluated conveniently at any x, Clenshaw-Curtis and Gauss would asymptotically achieve higher accuracy (error is $\mathcal{O}(\rho^{-n})$ or $\mathcal{O}(\rho^{-2n})$ for some $\rho > 1$) than composite Newton-Cotes with the same number $n+1$ nodes (error is $\mathcal{O}(n^{-p})$ for some fixed integer p). As long as we can freely choose quadrature nodes, high order Clenshaw-Curtis and Gauss are preferred to composite Newton-Cotes.

Clenshaw-Curtis and Gauss are overall comparable in cost and accuracy. Clenshaw-Curtis has advantage over Gauss for adaptive quadrature, whereas variants of Gauss quadrature can approximate certain improper integrals $\int_a^b f(x)\rho(x)dx$ with $\rho(x) \not\equiv 1$. We shall not further discuss these issues.

8.6 MATLAB's integral function

MATLAB's built-in function `integral` is a black-box tool to approximate the definite integral $\int_a^b f(x)dx$ numerically. The basic syntax to use this function is

```
Q = integral(fun,xmin,xmax)
```

Here, `fun` is a function handle to the integrand $f(x)$, and `xmin` and `xmas` are the lower and upper limits of the integral, respectively. For instance, to approximate $\int_0^\pi e^{-x}\cos 2x\,dx$, we can run the following MATLAB commands

```
func = @(x)exp(-x).*cos(2*x);
a = 0; b = pi;
format long e;
>> Q = integral(func,a,b)
Q =
   0.191357216347246
```

The result is consistent with the exact value $\frac{1-e^{-\pi}}{5}$ to 15 significant digits. We can also specify the absolute and relative error tolerances of the quadrature.

```
Q = integral(fun,xmin,xmax,'AbsTol',abstol,'RelTol',reltol)
```

Let $I_f = \int_a^b f(x)dx$ and $Q(f)$ be the quadrature. Then `integral` attempts to satisfy $|I_f - Q(f)| \leq \max(\texttt{abstol}, \texttt{reltol}\,|q|)$. For example, to approximate $\int_a^b e^{-5x} \sin\frac{1}{x} \sin\frac{1}{\sin\frac{1}{x}} dx$, where $a = 0.1593$ and $b = 0.3182$, we can run

```
a = 0.1593; b = 0.3182;
func = @(x)exp(-5*x).*sin(1./x).*sin(1./sin(1./x));
>> Q = integral(func,a,b)
Q =
   0.025616556321155

>> Q = integral(func,a,b,'AbsTol',eps,'RelTol',1e-14)
Q =
   0.025616556318470
```

The true value of this integral is $0.02561655631847027\cdots$. We get 9 accurate significant digits by using the default tolerance, and maximum 14 accurate significant digits by specifying the error tolerances.

The `integral` function may also be used to approximate certain improper integrals. For example, to approximate $\int_0^1 \ln x dx = -1$ and $\int_0^\infty e^{-x^2} x^2 dx = \frac{\sqrt{\pi}}{4} = 0.443113462726379$, we have

```
func = @(x)log(x);
a = 0; b = 1;
>> Q = integral(func,a,b)
Q =
  -1.000000010959678

>> func = @(x)exp(-x.^2).*x.^2;
a = 0; b = Inf;
>> Q = integral(func,a,b)
Q =
   0.443113462726377
```

However, `integral` has difficulties with improper integrals that are closer to a divergent one. For example, though it is easy to see $\int_0^1 (1-x)^{-\frac{9}{10}} dx = 10$, `integral` does not converge even if we specify small error tolerances.

```
func = @(x)(1-x).^(-9/10);
a = 0; b = 1;

>> Q = integral(func,a,b,'AbsTol',eps,'RelTol',eps)
Warning: Minimum step size reached near x = 1.
There may be a singularity, or the tolerances may be too tight
for this problem.
> In integralCalc/checkSpacing (line 456)
```

```
  In integralCalc/iterateScalarValued (line 319)
  In integralCalc/vadapt (line 132)
  In integralCalc (line 75)
  In integral (line 88)
Q =
   9.763682888351834
```

Such an improper integral can be approximated accurately and very easily by the Gauss-Jacobi quadrature (see our website for the code `gaussjac.m`). We shall not discuss more details in this book.

8.7 Excercise

1. Derive the $\frac{3}{8}$ rule by hand. For simplicity, let $x_0 = -1, x_1 = -\frac{1}{3}, x_2 = \frac{1}{3}$ and $x_3 = 1$, and we need to evaluate the weights $w_k = \int_{-1}^{1} L_k(x)dx$ where $L_k(x)$ is the Lagrange basis function associated with node x_k. Due to symmetry, we only need to evaluate w_0 and w_1, and let $w_2 = w_1$ and $w_3 = w_0$. Finally, obtain the general nodes $x_k \leftarrow \frac{b-a}{2}x_k + \frac{a+b}{2}$ and weights $w_k \leftarrow \frac{b-a}{2}$.

2. Verify that the degree of accuracy of Simpson's and the $\frac{3}{8}$ rules is exactly 3.

3. Find the 2nd order Taylor expansion of $f(x)$ at $\frac{a+b}{2}$, and show that the error of the midpoint rule is $\int_a^b f(x)dx - Q(f) = \frac{(b-a)^3}{24}f''(c)$ for some $c \in (a, b)$. Similarly, let $p_1(x)$ be the linear interpolant of $f(x)$ at $x_0 = a$ and $x_1 = b$. Find the error $f(x) - p_1(x)$ and show that the error of the trapezoidal rule is $\int_a^b f(x)dx - Q(f) = -\frac{(b-a)^3}{12}f''(c)$.

4. Let us explore the composite Newton-Cotes rules.
   (a) By hand and calculator, approximate $\int_{-1}^{1} \frac{dx}{5x^4+4x^3+3x^2+2x+1}$, with the composite trapezoidal and Simpson's rules, taking $\ell = 4$ subintervals.
   (b) Download the code `compncquad.m`, and use the composite trapezoidal, Simpson's and Boole's rules to approximate the integral in part (a). Take $\ell = 32, 64, 128, 256$ and 512 subintervals of length $h = \frac{2}{\ell}$, and find the errors $|I_f - Q(f)|$ for each ℓ and each quadrature. How much does the error decrease when ℓ is doubled for each quadrature, and why?

5. Use the composite trapezoidal rule to approximate $\int_0^2 \frac{1}{2+\cos \pi x}dx = \frac{2}{\sqrt{3}}$, using even number of subintervals from $\ell = 4$ to 14. Roughly how much does the error decrease as ℓ increases by 2? Explain your observation.

6. Exercise on the Clenshaw-Curtis quadrature.
   (a) By hand and calculator, find the Clenshaw-Curtis quadrature with 5

($n = 4$) nodes to approximate $\int_1^3 \ln x dx = 3\ln 3 - 2$, making use of the symmetry ($w_n = w_0$, $w_{n-1} = w_1$, etc). Verify with `clenshawcurtis.m`.
(b) Use `clenshawcurtis.m` to approximate $\int_{0.16}^{0.317} e^{-5x} \sin\frac{1}{x} \sin\frac{1}{\sin\frac{1}{x}} dx = 0.02561647733568396$. Let n be a multiple of 50, and find the minimal value of n to achieve 15-digit accuracy.

7. Exercise on the Gauss quadrature.
(a) Derive the 3-point Gauss quadrature on $[-1, 1]$, using the fact that its degree of accuracy is 5. (Hint: make full use of symmetry here, i.e., $w_0 = w_2$, $x_0 = -x_2$ and $x_1 = 0$).
(b) Find the order of errors of the composite $(n+1)$-point Gauss quadrature (but note that composite Gauss asymptotically does not converge as rapidly as high-order Gauss with the same total number of quadrature nodes).

8. More exercise on the Gauss quadrature.
(a) Look up the Legendre polynomials of degree 4 and 5, and numerically compute the nodes and weights of the 5-point Gauss quadrature ($n = 4$) (use MATLAB's arithmetic and `root` function). Evaluate this quadrature for $\int_1^3 \ln x dx = 3\ln 3 - 2$ and verify with `gausslg.m`.
(b) Use `gausslg.m` to approximate $\int_{0.16}^{0.317} e^{-5x} \sin\frac{1}{x} \sin\frac{1}{\sin\frac{1}{x}} dx$ (true value 0.02561647733568396). Let n be a multiple of 50, and find the minimal value of n to achieve 15-digit accuracy.

9. Comparison between Clenshaw-Curtis and Gauss.
(a) Write a code to compare the accuracy of Clenshaw-Curtis and Gauss for approximating $\int_{0.16}^{0.317} e^{-5x} \sin\frac{1}{x} \sin\frac{1}{\sin\frac{1}{x}} dx$, using `clenshawcurtis.m` and `gausslg.m`. Plot the integrand on $[0.16, 0.317]$ and the errors of the two quadrature for $n = 10, 20, \ldots, 490, 500$. What do you see, and why?
(b) Repeat the same experiment for $\int_{-1}^{1} e^x \left[\mathrm{sech}(4\sin(40x))\right]^{e^x} dx$ (true value 0.5433840009079006). Plot the integrand on $[-1, 1]$ and the errors of the two quadrature for $n = 50, 100, \ldots, 1950, 2000$. What do you see, and why?