Chapter 2

Numerical integration

This chapter is concerned with algorithms for approximating a definite integral

$$\int_a^b f(x)\,\mathrm{d}x\ , \tag{2.1}$$

for a function $f : [a, b] \to \mathbb{R}$ and specific values of a, b. In Calculus courses it is common to calculate simple integrals like $\int_0^1 e^x\,\mathrm{d}x$ or $\int_0^\pi \cos(x)\,\mathrm{d}x$ using a closed-form primitive for f obtained from a table of integrals or computer algebra systems like Maple, Mathematica, the Symbolic Math Toolbox in MATLAB (which is based on MuPAD) or simply ask WolframAlpha / ChatGPT questions like "What is the primitive of $\exp(-x^2)$?" When trying to compute more complicated expressions like $\int_0^1 e^{x^2}\,\mathrm{d}x$ or $\int_0^\pi \cos(x^2)\,\mathrm{d}x$, one quickly realizes the limitations of such approaches. Indeed, in practice, it is usually *not* possible to find a closed form expression for a primitive. Still, evaluating very complicated definite integrals is a common problem arising in many areas of science and engineering. In these cases, one needs to resort to numerical methods (and understand their limitations).

The goal of this chapter is to introduce and analyze the most popular numerical methods for approximating definite integrals. In the following, unless otherwise stated, we will assume that f is (at least) continuous on $[a, b]$ and hence the integral is well defined.

2.1 A first glimpse at polynomial interpolation

A common principle to derive numerical integration methods is to interpolate f by a polynomial and obtain an approximation by computing the definite integral for this polynomial. We will therefore first take a brief excursion to the world of polynomial interpolation. We will discuss interpolation in more detail in the next chapter.

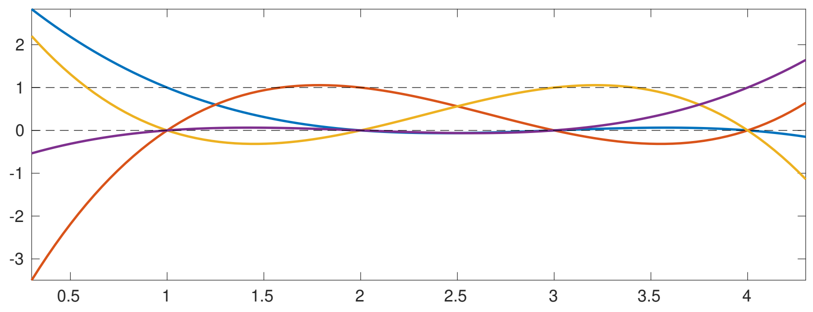


Figure 2.1: The basis of 4 Lagrange polynomials for the nodes $1, 2, 3, 4$.

We let $\mathbb{P}_n$ denote the vector space of real polynomials of degree at most n:

$$\mathbb{P}_n := \operatorname{span}\{x^i \,:\, i = 0, \dots, n\} = \Big\{ \sum_{i=0}^{n} c_i x^i \,:\, c_i \in \mathbb{R} \Big\}.$$

Given interpolation data (x_j, f_j) with $x_j \in \mathbb{R}$ and $f_j \in \mathbb{R}$ for $j = 0, \dots, n$, the task of polynomial interpolation is to find a polynomial $p_n \in \mathbb{P}_n$ such that

$$p_n(x_j) = f_j, \qquad j = 0, \dots, n. \tag{2.2}$$

For $n = 0$ (constant p_0), $n = 1$ (linear p_1), and $n = 2$ (quadratic p_2), it is quite intuitive to see that (2.2) has a unique solution if and only if the **interpolation nodes x_j are pairwise distinct**. To verify this statement for general n, we first need to define a suitable basis for $\mathbb{P}_n$. The monomial basis $\{1, x, x^2, \dots, x^n\}$ appears to be the canonical choice, but it is actually not well suited for interpolation on the real line because it turns (2.2) into an extremely ill-conditioned linear system. Later on, We will learn more about the impediments of ill-conditioning.

Instead of monomials, we will use the following polynomials whose choice depends on the interpolation nodes.

Definition 2.1 *Given $n+1$ pairwise distinct nodes $x_0, \dots, x_n \in \mathbb{R}$, the polynomials $\ell_j \in \mathbb{P}_n$ defined by*

$$\ell_j(x) := \prod_{i=0, i\neq j}^{n} \frac{x - x_i}{x_j - x_i}, \qquad j = 0, \dots, n, \tag{2.3}$$

are called **Lagrange polynomials***.*

The following relation is an important property of Lagrange polynomials:

$$\ell_j(x_i) = \delta_{ij} := \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

Because $x_0, \dots, x_n \in \mathbb{R}$ are pairwise distinct, this implies that $\{\ell_0, \dots, \ell_n\}$ is a linearly independent family and, consequently, it is a basis of $\mathbb{P}_{n+1}$ because $\mathbb{P}_{n+1}$ has dimension $n+1$. Moreover, it also proves the existence and uniqueness of interpolating polynomials.

Theorem 2.2 *Given interpolation data (x_j, f_j), $j = 0, \dots, n$, with pairwise distinct interpolation nodes $x_0, \dots, x_n$, the polynomial*

$$p_n(x) := \sum_{j=0}^{n} f_j \ell_j(x) \tag{2.5}$$

is the unique polynomial of degree at most n that satisfies the interpolation conditions (2.2).

Proof. Because $\ell_j \in \mathbb{P}_n$, it follows that the polynomial p_n defined in (2.5) is also in $\mathbb{P}_n$. Moreover, it follows from (2.4) that p_n satisfies the interpolation conditions (2.2).

It remains to show the uniqueness of the solution. For this purpose, let $\tilde{p}_n \in \mathbb{P}_n$ denote another polynomial satisfying (2.2). Then $q_n := p_n - \tilde{p}_n$ is a polynomial of degree at most n. By definition, $q_n(x_j) = 0$ for $j = 0, \dots, n$, which shows that q_n has $n+1$ has pairwise distinct zeros. According to the fundamental theorem of algebra, the only polynomial of degree at most n having $n+1$ pairwise distinct zeros is the zero polynomial $q_n \equiv 0$. Hence, $\tilde{p}_n \equiv p_n$. □

One important application of interpolation is to replace a complicated function f by a polynomial. The following theorem provides an expression for the interpolation error if f is sufficiently smooth.

Theorem 2.3 *Let $x_0 < x_1 < \dots < x_n$ and let $p_n \in \mathbb{P}_n$ denote the interpolating polynomial satisfying $p_n(x_j) = f(x_j)$, $j = 0, \dots, n$, for some function*

$$f \in C^{n+1}([x_0, x_n]).$$

Given $x_\star \in [x_0, x_n]$, there exists $\xi \in [x_0, x_n]$ such that

$$E_n[f](x_\star) := f(x_\star) - p_n(x_\star) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\, \omega_{n+1}(x_\star)\,, \tag{2.6}$$

where

$$\omega_{n+1}(x) := (x - x_0)(x - x_1) \cdots (x - x_n) \in \mathbb{P}_{n+1}\ . \tag{2.7}$$

Proof. If $x_\star$ equals one of the interpolation nodes x_j then $E_n[f](x_\star) = E_n[f](x_j) = 0$. Hence, we may assume that $x_\star \neq x_j$. For $x \in I := [x_0, x_n]$ we define the function

$$g(x) := E_n[f](x) - \omega_{n+1}(x)\, E_n[f](x_\star)/\omega_{n+1}(x_\star)\,. \tag{2.8}$$

Because $f \in C^{n+1}([x_0, x_n])$ and $\omega_{n+1} \in \mathbb{P}_{n+1}$ it follows that $g \in C^{n+1}([x_0, x_n])$. Moreover, g has at least $n+2$ distinct zeros in $[x_0, x_n]$ because

$$\begin{aligned} g(x_j) &= E_n[f](x_j) - \omega_{n+1}(x_j)\, E_n[f](x_\star)/\omega_{n+1}(x_\star) = 0, \quad j = 0, \ldots, n, \\ g(x_\star) &= E_n[f](x_\star) - \omega_{n+1}(x_\star)\, E_n[f](x_\star)/\omega_{n+1}(x_\star) = 0\,. \end{aligned}$$

By Rolle's theorem, $g' = \frac{dg}{dx}$ has at least $n+1 = n+2-1$ zeros in $[x_0, x_n]$. Continuing this argument, $g^{(i)} = \frac{d^i g}{dx^i}$ has at least $n+2-i$ zeros for $i = 1, \ldots, n+1$. Hence, $g^{(n+1)}$ has at least one zero, which we denote by ξ. Because $E_n^{(n+1)}[f](x) = f^{(n+1)}(x)$ and $\omega_{n+1}^{(n+1)}(x) \equiv (n+1)!$, it follows from (2.8) that

$$0 = g^{(n+1)}(\xi) = f^{(n+1)}(\xi) - (n+1)!\, E_n[f](x_\star)/\omega_{n+1}(x_\star)\,.$$

Rearranging this relation yields (2.6). $\square$

Barycentric representation* The straightforward use of the interpolating polynomial $p_n(x)$ in the representation (2.5) would require $O(n^2)$ operations for *each* evaluation of p_n. This complexity can be reduced by *first* computing the following quantities:

$$\lambda_j = \frac{1}{(x_j - x_0)\cdots(x_j - x_{j-1})(x_j - x_{j+1})\cdots(x_j - x_n)}, \quad j = 0, \ldots, n.$$

This allows us to write

$$\ell_j(x) = \omega_{n+1}(x)\frac{\lambda_j}{x - x_j},$$

with $\omega_{n+1}(x)$ defined as in (2.7). It follows that

$$p_n(x) = \sum_{j=0}^{n} f_j \ell_j(x) = \omega_{n+1}(x) \sum_{j=0}^{n} \frac{\lambda_j f_j}{x - x_j}. \tag{2.9}$$

Now, only $O(n)$ operations are needed for each operation (after λ_j has been computed). We can simplify this even further. When considering the interpolation of the constant function 1, the relation (2.9) reduces to

$$1 = \omega_{n+1}(x) \sum_{j=0}^{n} \frac{\lambda_j}{x - x_j}.$$

Inserted into (2.9), this yields the **barycentric interpolation formula**

$$p_n(x) = \sum_{j=0}^{n} \frac{\lambda_j f_j}{x - x_j} \Bigg/ \sum_{j=0}^{n} \frac{\lambda_j}{x - x_j}.$$

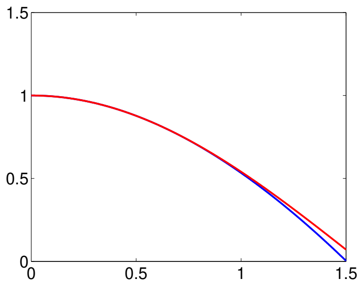
Example Let $x_0 = 0, x_1 = 0.2, x_2 = 0.4, x_3 = 0.6$ and $f_j = \cos(x_j)$. Then the parameters of the barycentric interpolation formula are given by

$$\begin{aligned}
\lambda_0 &= \frac{1}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} = -20.8333,\\
\lambda_1 &= \frac{1}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} = 62.5000,\\
\lambda_2 &= \frac{1}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} = -62.5000,\\
\lambda_3 &= \frac{1}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} = 20.8333.
\end{aligned}$$

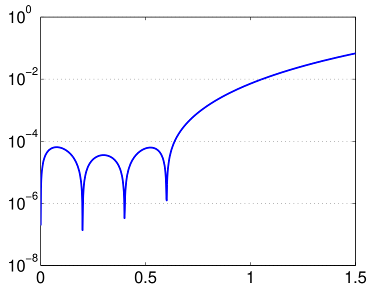
The interpolating polynomial in the Lagrange representation is given by

$$\begin{aligned}
p_3(x) &= y_0\ell_0(x) + y_1\ell_1(x) + y_2\ell_2(x) + y_3\ell_3(x)\\
&= \quad y_0\lambda_0(x-x_1)(x-x_2)(x-x_3) + y_1\lambda_1(x-x_0)(x-x_2)(x-x_3)\\
&\quad + y_2\lambda_2(x-x_0)(x-x_1)(x-x_3) + y_3\lambda_3(x-x_0)(x-x_1)(x-x_2)\\
&= -20.8333(x-x_1)(x-x_2)(x-x_3) + 61.2542(x-x_0)(x-x_2)(x-x_3)\\
&\quad -57.5663(x-x_0)(x-x_1)(x-x_3) + 17.1945(x-x_0)(x-x_1)(x-x_2).
\end{aligned}$$

cos (red) und $p_3(x)$ (blue)



Interpolation error



2.2 Newton-Cotes formulae

We now come back to the idea of approximating the definite integral (2.1) by integrating an interpolating polynomial. If the interpolation nodes are uniformly distributed, the quadrature rule resulting from this approach is called a *Newton-Cotes formula*. In the following, we first treat in detail the three classic cases before discussing the general case.

1. **Midpoint rule**
   The function f is interpolated by a constant polynomial $p_0 \in \mathbb{P}_0$ in the middle of the interval:
   $$p_0(x) \equiv f\Big(\frac{a+b}{2}\Big).$$

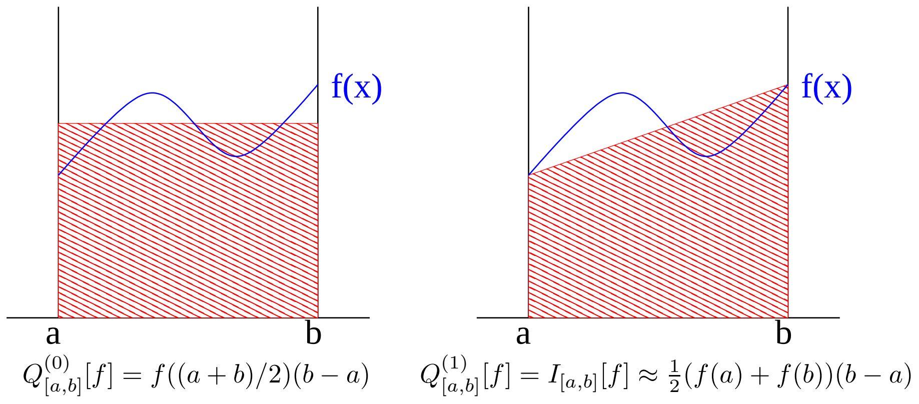

f(x)
a
b
$Q^{(0)}_{[a,b]}[f] = f((a+b)/2)(b-a)$
f(x)
a
b
$Q^{(1)}_{[a,b]}[f] = I_{[a,b]}[f] \approx \frac{1}{2}(f(a)+f(b))(b-a)$

Figure 2.2: Illustration of midpoint and trapezoidal rule for approximating the definite integral $\int_a^b f(x)\,\mathrm{d}x$.

The resulting approximation of the integral is given by

$$\int_a^b f(x)\,\mathrm{d}x \approx \int_a^b p_0(x)\,\mathrm{d}x = (b-a)\cdot f\Big(\frac{a+b}{2}\Big) =: Q^{(0)}_{[a,b]}[f],$$

see Figure 2.2 for an illustration.

2. **Trapezoidal rule**
   The linear polynomial $p_1 \in \mathbb{P}_1$ interpolating f at the two end-points of the interval $[a,b]$ (that is, $p_1(a) = f(a)$ and $p_1(b) = f(b)$ holds) is given by

$$p_1(x) = \frac{x-b}{a-b} f(a) + \frac{x-a}{b-a} f(b).$$

   The resulting approximation of the integral is the *trapezoidal rule* given by

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x &\approx \int_a^b p_1(x)\,\mathrm{d}x = f(a)\int_a^b \frac{x-b}{a-b}\,\mathrm{d}x + f(b)\int_a^b \frac{x-a}{b-a}\,\mathrm{d}x \\ &= f(a)(b-a)\int_0^1 y\,\mathrm{d}y + f(b)(b-a)\int_0^1 y\,\mathrm{d}y \\ &= (b-a)\Big(\frac{1}{2}f(a) + \frac{1}{2}f(b)\Big) =: Q^{(1)}_{[a,b]}[f].\end{aligned}$$

3. **Simpson rule**
   Let $p_2 \in \mathbb{P}_2$ be the quadratic polynomial interpolating f at the mid- and end-points:

$$p_2(a) = f(a), \quad p_2(x_1) = f(x_1), \quad p_2(b) = f(b), \qquad x_1 = \frac{a+b}{2}.$$

This polynomial is given by

$$p_2(x) = \frac{(x-x_1)(x-b)}{(a-x_1)(a-b)}f(a) + \frac{(x-a)(x-b)}{(x_1-a)(x_1-b)}f(x_1) + \frac{(x-a)(x-x_1)}{(b-a)(b-x_1)}f(b).$$

To compute the definite integral of p_2, we again use the variable substitution $y = \frac{x-a}{b-a}$ to obtain

$$\int_a^b \frac{(x-x_1)(x-b)}{(a-x_1)(a-b)}\,\mathrm{d}x = (b-a)\int_0^1 \frac{(x-1/2)(x-1)}{1/2}\,\mathrm{d}y = \frac{1}{6}(b-a).$$

Similarly, one computes

$$\int_a^b \frac{(x-a)(x-b)}{(x_1-a)(x_1-b)}\,\mathrm{d}x = \frac{4}{6}(b-a), \quad \int_a^b \frac{(x-a)(x-x_1)}{(b-a)(b-x_1)}\,\mathrm{d}x = \frac{1}{6}(b-a).$$

In turn, the resulting approximation of the integral is the *Simpson rule* given by

$$\int_a^b f(x)\,\mathrm{d}x \approx (b-a)\Big(\frac{1}{6}f(a) + \frac{4}{6}f\Big(\frac{a+b}{2}\Big) + \frac{1}{6}f(b)\Big) =: Q^{(2)}_{[a,b]}[f].$$

The calculations above can, in fact, be somewhat simplified by first performing the variable substitution

$$\int_a^b f(x)\,\mathrm{d}x = (b-a)\int_1^0 \tilde{f}(y)\,\mathrm{d}y, \quad \tilde{f}(y) := f(a+(b-a)y) \tag{2.10}$$

and then applying interpolation to $\tilde{f}$ on the interval $[0,1]$. This procedure yields the same Simpson rule.

For the general case, we choose $n+1$ interpolation nodes

$$a \le x_0 < x_1 \le \cdots < x_n \le b.$$

We recall the Lagrange representation of the interpolating polynomial:

$$p_n(x) = \sum_{j=0}^n f(x_j)\ell_j(x), \qquad \ell_j(x) = \prod_{\substack{i=0\\ i\neq j}}^n \frac{x-x_i}{x_j-x_i}\ .$$

The definite integral of p_n yields the approximation

$$\begin{aligned} \int_a^b f(x)\,\mathrm{d}x \approx Q^{(n)}_{[a,b]}[f] \quad &:= \quad \int_a^b p_n(x)\,\mathrm{d}x = \int_a^b \sum_{j=0}^n f(x_j)\ell_j(x)\ \mathrm{d}x \\ &= \quad \sum_{j=0}^n f(x_j)\int_a^b \ell_j(x)\ \mathrm{d}x = \sum_{j=0}^n \alpha_j f(x_j), \end{aligned} \tag{2.11}$$

n		ξ_j	$\alpha_j/(b-a)$	Error
0	Midpoint rule	$\frac{1}{2}$	1	$\frac{1}{24}(b-a)^3 f^{(2)}(\xi)$
1	Trap. rule	$0, 1$	$\frac{1}{2}, \frac{1}{2}$	$-\frac{1}{12}(b-a)^3 f^{(2)}(\xi)$
2	Simpson rule	$0, \frac{1}{2}, 1$	$\frac{1}{6}, \frac{4}{6}, \frac{1}{6}$	$-\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\xi)$
3	$\frac{3}{8}$ rule	$0, \frac{1}{3}, \frac{2}{3}, 1$	$\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}$	$-\frac{3}{80}\left(\frac{b-a}{3}\right)^5 f^{(4)}(\xi)$
4	Milne rule	$0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1$	$\frac{7}{90}, \frac{32}{90}, \frac{12}{90}, \frac{32}{90}, \frac{7}{90}$	$-\frac{8}{945}\left(\frac{b-a}{4}\right)^7 f^{(6)}(\xi)$

Table 2.1: Newton-Cotes formulae ($x_j = a + \xi_j(b-a)$).

where the scalars $\alpha_j := \int_a^b \ell_j(x)\,\mathrm{d}x$ are called the **weights** of the quadrature rule $Q^{(n)}_{[a,b]}[f]$. Using the substitution (2.10), it can be seen that $\alpha_j/(b-a)$ is a constant not depending on a, b or f. As mentioned in the beginning, if the points are uniformly distributed,

$$x_j = a + jh, \quad j = 0, \dots, n, \quad h = \frac{b-a}{n},$$

for $n \geq 1$ then $Q^{(n)}_{[a,b]}[f]$ is called a (closed) Newton-Cotes formula. Table 2.1 shows the quadrature points $x_j = a + \xi_j(b-a)$ und weights for $n \leq 4$. The fourth column contains the quadrature error, which will be discussed in the next section.

2.3 Order and error analysis

The order of a quadrature rule is determined by its ability to integrate polynomials up to a certain degree exactly.

Definition 2.4 *A quadrature rule $Q_{[a,b]}$ has* **order s + 1**, *if*

$$\int_a^b p_s(x)\,\mathrm{d}x = Q_{[a,b]}[p_s], \qquad \forall p_s \in \mathbb{P}_s.$$

Every commonly used quadrature rule $Q_{[a,b]}$ is a linear operator and, hence, Definition 2.4 is equivalent to verifying that

$$Q_{[0,1]}[1] = 1,\ Q_{[0,1]}[x] = \frac{1}{2},\ \dots,\ Q_{[0,1]}[x^s] = \frac{1}{s+1}.$$

By construction, the Newton-Cotes formula $Q^{(n)}_{[a,b]}[f]$ has order *at least* $n+1$ because polynomials of degree up to n are interpolated and, hence, integrated exactly. It is easy to verify that the mid-point rule has, in fact, one order higher, order 2. The trapezoidal rule also has order 2 and the Simpson rule has order 4, again one order higher.

The order has a pronounced influence on the quadrature error $\int_a^b f \,\mathrm{d}x - Q_{[a,b]}[f]$ if f is sufficiently smooth and the interval $[a,b]$ is small. To see this intuitively, consider the (truncated) Taylor expansion of f at a

$$f(x) = t_s(x) + \frac{f^{(s+1)}(\xi_x)}{(s+1)!}(x-a)^{s+1}, \qquad t_s \in \Pi_s.$$

Because t_s is integrated exactly, the quadrature error is determined by the second term. The absolute value of the second term is $O(h^{s+1})$ for $h := b - a \to 0$. Integrating it over an interval of length h gives an integration error of $O(h^{s+2})$. Finer estimates can be obtained by applying Theorem 2.3. The following theorem and proof establish the quadrature error for $n = 0$ and $n = 1$; the other entries in Table 2.1 can be established in an analogous fashion.

Theorem 2.5 *Let $f \in C^2[a,b]$.*

i) *There is $\xi \in [a,b]$ such that the error of the midpoint rule satisfies*

$$\int_a^b f(x)\,\mathrm{d}x - (b-a)\, f\Big(\frac{a+b}{2}\Big) = \frac{(b-a)^3}{24}\, f''(\xi).$$

ii) *There is $\xi \in [a,b]$ such that the error of the trapezoidal rule satisfies*

$$\int_a^b f(x)\,\mathrm{d}x - \frac{b-a}{2}\Big[f(a)+f(b)\Big] = -\frac{(b-a)^3}{12}\, f''(\xi).$$

Proof. ii) We first prove the second part, because it follows directly from Theorem 2.3:

$$f(x) - p_1(x) = \frac{f''(\xi_x)}{2}(x-a)(x-b).$$

for some $\xi_x \in [a,b]$ (depending on x). Integrating both sides yields

$$\int_a^b f(x)\,\mathrm{d}x - Q^{(1)}_{[a,b]}[f] = \int_a^b \frac{f''(\xi_x)}{2}(x-a)(x-b)\,\mathrm{d}x. \tag{2.12}$$

Let

$$c = \frac{\int_a^b \frac{f''(\xi_x)}{2}(x-a)(x-b)\;\mathrm{d}x}{\frac{1}{2}\int_a^b (x-a)(x-b)\;\mathrm{d}x} = \frac{\int_a^b f''(\xi_x)(x-a)(x-b)\;\mathrm{d}x}{-\frac{1}{6}(b-a)^3}.$$

Because $(x-a)(x-b) \le 0$ for every x in $[a,b]$, it follows that

$$\min_{x\in[a,b]} f''(x) \le c \le \max_{x\in[a,b]} f''(x).$$

Since f'' is continuous, the intermediate value theorem shows that there exists $\xi \in [a,b]$ (not depending on x) such that $c = f''(\xi)$. Inserting this into (2.12) completes the proof of part ii).

i) Let $x_0 = \frac{a+b}{2}$. Using Taylor expansion at x_0, it follows that

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2} f''(\xi_x)(x - x_0)^2$$

for some ξ_x between x_0 and x. In turn, the quadrature error satisfies

$$\int_a^b f(x)\,\mathrm{d}x - Q^{(0)}_{[a,b]}[f] = f'(x_0) \int_a^b (x - x_0)\,\mathrm{d}x + \frac{1}{2} \int_a^b f''(\xi_x)(x - x_0)^2\,\mathrm{d}x.$$

Note that the first term vanishes because $\int_a^b (x-x_0)\,\mathrm{d}x = 0$. Because $(x-x_0)^2 \geq 0$, we can apply the intermediate value theorem as in part ii) to conclude that there exists $\xi \in [a, b]$ such that

$$\frac{1}{2} \int_a^b f''(\xi_x)(x - x_0)^2\,\mathrm{d}x = \frac{1}{2} f''(\xi) \int_a^b (x - x_0)^2\,\mathrm{d}x = \frac{1}{24} f''(\xi)(b - a)^3.$$

□

Table 2.1 might suggest that it is a good idea to go even further and use very large for n (on relatively small intervals). However, for $n = 8$ and larger some of the weights become negative, which leads to numerical cancellation and limits the usefulness of such high-order Newton-Cotes formulae.

2.4 Composite Newton-Cotes formulae

As mentioned above, the gains in accuracy by increasing the order of Newton-Cotes formulae are limited due to the influence of roundoff error. A more effective approach to increasing accuracy is to partition the interval $[a, b]$ in N subintervals and approximate $I_{[a,b]}[f]$ by applying a quadrature rule to each subinterval.

Notation: *From now on, x_i, $i = 0, \ldots, N$, denote the boundaries of the subintervals, N denotes the number of subintervals, and n denotes (as before) the maximum polynomial degree used in the integration of subintervals.*

A uniform partition of $[a, b]$ into N subintervals corresponds to

$$x_j = a + jh, \qquad j = 0, \ldots, N, \qquad h = \frac{b - a}{N}.$$

Applying the quadrature rule $Q^{(n)}_{[x_i,x_{i+1}]}[f]$ to each subinterval und summing the obtained values yields the corresponding **composite** quadrature rule:

$$Q^{(n)}_h[f] := \sum_{i=0}^{N-1} Q^{(n)}_{[x_i,x_{i+1}]}[f]. \tag{2.13}$$

For $n = 1$ we obtain the **composite trapezoidal rule**:

$$Q^{(1)}_h[f] = \sum_{i=0}^{N-1} \frac{x_{i+1} - x_i}{2} \left[f(x_i) + f(x_{i+1})\right] = \frac{h}{2} \left[f(a) + 2 \sum_{i=1}^{N-1} f(x_i) + f(b)\right].$$

For $n = 1$ we obtain the **composite Simpson rule**:

$$\begin{aligned} Q_h^{(2)}[f] &= \sum_{i=0}^{N-1} \frac{x_{i+1}-x_i}{6}\,\Big[f(x_i) + 4f\Big(\frac{x_i+x_{i+1}}{2}\Big) + f(x_{i+1})\Big] \\ &= \frac{h}{6}\,\Big[f(a) + 2\sum_{i=1}^{N-1} f(x_i) + 4\sum_{i=0}^{N-1} f\Big(\frac{x_i+x_{i+1}}{2}\Big) + f(b)\Big]. \end{aligned}$$

Theorem 2.6 *For the composite trapezoidal and Simpson rules it holds that*

$$\big|I_{[a,b]}[f] - Q_h^{(1)}[f]\big| \le \frac{h^2}{12}(b-a)\max_{x\in[a,b]}|f''(x)|, \quad f \in C^2[a,b],$$

$$\big|I_{[a,b]}[f] - Q_h^{(2)}[f]\big| \le \frac{h^4}{2880}(b-a)\max_{x\in[a,b]}|f^{(4)}(x)|, \quad f \in C^4[a,b].$$

Proof. Applying Theorem 2.5 to each subinterval yields

$$\begin{aligned} \big|I_{[a,b]}[f] - Q_h^{(1)}[f]\big| &= \Big|\sum_{i=0}^{N-1}\frac{(x_{i+1}-x_i)^3}{12}\,f''(\xi_i)\Big| \\ &\le \sum_{i=0}^{N-1}\frac{h^3}{12}|f''(\xi_i)| \le \frac{h^2}{12}(b-a)\max_{x\in[a,b]}|f''(x)|. \end{aligned}$$

The proof for the Simpson rule proceeds analogously. □

Example 2.7 Figure 2.3 shows the implementations and the errors obtained when applying the composite trapezoidal and Simpson rules to approximating

$$\int_0^{\pi} \sin(x)\,\mathrm{d}x = 2. \tag{2.14}$$

As expected, the error of the (composite) Simpson rule converges much faster to zero than the error of the trapezoidal rule. The asymptotic behavior is clearly $O(h^4)$ and $O(h^2)$, as predicted by Theorem 2.6. As a consequence, the Simpson rule needs much fewer function evaluations to attain the same accuracy.

If f is not smooth, the advantage of the Simpson rule becomes less important. To see this, consider

$$\int_0^1 \sqrt{x}\,\mathrm{d}x = \frac{2}{3}. \tag{2.15}$$

Figure 2.4 shows the error of the composite trapezoidal and Simpson rules. For both rules, the asymptotic behavior of the error is $O(h^p)$ with $p = 3/2$ (but with a smaller constant for the Simpson rule). ◇

Python
```python
import numpy as np
import matplotlib.pyplot as plt

def trapez(fun, a, b, n):
  x = np.linspace(a,b,n+1)
  vecfun = np.vectorize(fun)
  f = vecfun(x)
  f[0] = f[0]/2
  f[-1] = f[-1]/2
  T = (b-a) * sum(f) / n
  return T

nn = 100; err = np.zeros(nn)
for n in range(nn):
  err[n] = np.abs(2 -
  trapez(np.sin, 0, np.pi, n + 1))
plt.loglog(range(1,nn+1), err)
```

Python
```python
def simpson(fun,a,b,n):
  x = np.linspace(a,b,2*n+1)
  vecfun = np.vectorize(fun)
  f = vecfun(x)
  end = len(f) - 1
  f[1:end: 2] = 4*f[1:end:2]
  f[2:end-1:2] = 2*f[2:end-1:2]
  T = (b-a) * sum(f) / n / 6
  return T
```

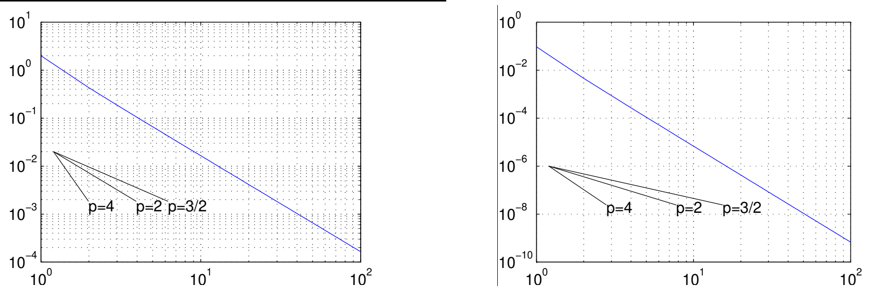


Figure 2.3: Errors of composite Simpson rule (left) and trapezoidal rule (right) vs. $N = O(h^{-1})$ when approximating smooth integral (2.14).

The **composite midpoint rule**,

$$Q_h^{(0)}[f] = \sum_{i=0}^{N-1} (x_{i+1} - x_i) f\Big(\frac{x_i + x_{i+1}}{2}\Big)$$

is of interest for functions with singularities at the interval boundaries; see Section 2.6.2 below.

2.5 Gauss formulae

In the Newton-Cotes formulae, we have chosen the interpolation points to be uniformly distributed. In this section, we modify these points in order to obtain a quadrature rule of higher order.

Remark 2.8 *There is no quadrature rule* (2.11) *with* $n + 1$ *points* $x_0, \dots, x_n$ *of order higher than* $2n + 2$. *If* $Q^{(n)}[p]$ *was such a quadrature rule it would be exact*

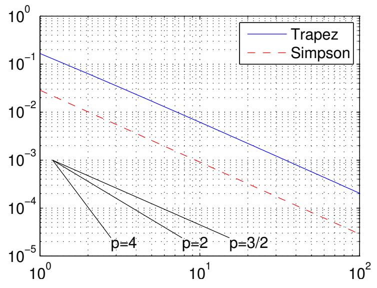


Figure 2.4: Errors of composite Simpson and trapezoidal rules vs. $N = O(h^{-1})$ when approximating non-smooth integral (2.15).

for the polynomial

$$p(x) = \prod_{i=0}^{n} (x - x_i)^2 \in \mathbb{P}_{2n+2}\,,$$

which leads to the contradiction

$$0 < \int_a^b p(x)\,dx = Q^{(n)}[p] = 0\,.$$

In the following, we construct quadrature rules that attain the highest possible order according to Remark 2.8, the so called **Gauss formulae**. As an additional benefit, these formulae always have nonnegative weights and therefore avoid the numerical problems associated with high orders for the Newton-Cotes formulae.

To construct Gauss formulae, we have to choose the quadrature points $x_0, \dots, x_n$ such that $Q^{(n)}$ exactly integrates all polynomials of degree at most $2n+1$. Because of linearity, it suffices to verify this property for any basis $\mathbb{P}_{2n+1}$. When choosing the monomial basis this leads to the $2n+2$ (nonlinear) equations

$$\int_{-1}^{1} x^k \,\mathrm{d}x = \alpha_0 x_0^k + \cdots + \alpha_n x_n^k, \quad k = 0, \dots, 2n+1. \tag{2.16}$$

Here and in the following, we assume for simplicity that $[a, b] = [-1, 1]$. The solution of the nonlinear equations (2.16) by hand or computer algebra becomes quickly infeasible for larger n. There is a much simpler and more elegant approach, which will be discussed in the following. Polynomial division provides the starting point of this approach.

Theorem 2.9 (Polynomial division) *Let $p \in \mathbb{P}_{2n+1}$ and $q \in \mathbb{P}_{n+1}$. There exist unique polynomials $h \in \mathbb{P}_n$ and $r \in \mathbb{P}_n$ such that $p = hq + r$.*

For integrating a polynomial $p \in \mathbb{P}_{2n+1}$ exactly, it must hold that

$$\begin{aligned} 0 &= \int_{-1}^{1} p(x)\,\mathrm{d}x - \sum_{j=0}^{n} \alpha_j p(x_j) \\ &= \int_{-1}^{1} h(x)q(x)\,\mathrm{d}x - \sum_{j=0}^{n} \alpha_j h(x_j)q(x_j) + \left(\int_{-1}^{1} r(x)\,\mathrm{d}x - \sum_{j=0}^{n} \alpha_j r(x_j) \right) \end{aligned} \tag{2.17}$$

Because $r \in \mathbb{P}_n$, the third term (in brackets) is always zero when using an interpolating quadrature rule with $n+1$ points. For addressing the first two terms, we will make a clever choice of q via Legendre polynomials.

Definition 2.10 *The* **Legendre polynomial** *(of order $n+1$) is the polynomial $q_{n+1} \in \mathbb{P}_{n+1}$ satisfying*

$$\int_{-1}^{1} q_{n+1}(x)h(x)\,\mathrm{d}x = 0 \ \forall h \in \mathbb{P}_n, \quad q_{n+1}(1) = 1. \tag{2.18}$$

Defining the L_2 inner product

$$\langle p, q \rangle = \int_{-1}^{1} p(x)q(x)\,\mathrm{d}x$$

on the vector space $\mathbb{P}_{n+1}$, the first condition in (2.18) states that q_{n+1} is orthogonal to the subspace $\mathbb{P}_n$. In turn, this shows that Legendre polynomials can be constructed by applying the Gram-Schmidt procedure to the monomial basis 1, x, …, x^{n+1}. The next result provides a more direct characterization of Legendre polynomials.

Theorem 2.11 *The polynomial q_k defined by*

$$q_k(x) = c_k \frac{\mathrm{d}^k}{\mathrm{d}x^k}\left[(x^2-1)^k\right], \qquad c_k := \frac{1}{2^k k!}, \tag{2.19}$$

is the kth Legendre polynomial.

Proof. First, we note that $q_k \in \mathbb{P}_k$, since it is obtained by computing k derivatives of a polynomial of degree $2k$. Also, it is straightforward to see that $q_k(1) = 1$. To show that q_k is orthogonal to every $g \in \mathbb{P}_{k-1}$ we proceed via integration by parts:

$$\begin{aligned} c_k \int_{-1}^{1} \frac{d^k}{dx^k}\left[(x^2-1)^k\right] g(x)\ \mathrm{d}x = &- c_k \int_{-1}^{1} \frac{d^{k-1}}{dx^{k-1}}\left[(x^2-1)^k\right] \frac{d}{dx} g(x)\ \mathrm{d}x \\ &+ c_k \left[\frac{d^{k-1}}{dx^{k-1}} (x^2-1)^k g(x) \right]_{-1}^{1} \end{aligned}$$

Note that the second term vanishes because $(x^2-1)^k$ has a zero of multiplicity k at ± 1. To treat the first term, we again integrate by parts:

$$-c_k \int_{-1}^{1} \frac{d^{k-1}}{dx^{k-1}} \left[(x^2-1)^k\right] \frac{d}{dx} g(x)\ \mathrm{d}x = c_k \int_{-1}^{1} \frac{d^{k-2}}{dx^{k-2}} \left[(x^2-1)^k\right] \frac{d^2}{dx^2} g(x)\ \mathrm{d}x \\ - c_k \left[\frac{d^{k-2}}{dx^{k-2}} (x^2-1)^k \frac{d}{dx} g(x)\right]_{-1}^{1}.$$

Once again, the second term vanishes. Continuing this procedure and integrating by parts k times, we obtain

$$c_k \int_{-1}^{1} \frac{d^k}{dx^k} \left[(x^2-1)\right]^k g(x)\ \mathrm{d}x = (-1)^k c_k \int_{-1}^{1} \left(x^2-1\right) \frac{d^k}{dx^k} g(x)\ \mathrm{d}x.$$

Since g has degree at most $\mathbb{P}_{k-1}$ its kth derivative vanishes and, hence,

$$c_k \int_{-1}^{1} \frac{d^k}{dx^k} \left[(x^2-1)\right]^k g(x)\ \mathrm{d}x = 0.$$

□

Theorem 2.11 implies a recurrence relation, which is convenient for computing Legendre polynomials.

Theorem 2.12 *The Legendre polynomials $q_0, q_1, \ldots$ satisfy the three-term recurrence relation*

$$q_{n+1}(x) = \frac{2n+1}{n+1} x q_n(x) - \frac{n}{n+1} q_{n-1}(x), \quad q_0(x) = 1, \quad q_1(x) = x.$$

Proof. EFY. □

According to Theorem 2.12, the first five Legendre polynomials are given by

$$\begin{array}{rcl} q_0(x) & = & 1 \\ q_1(x) & = & x \\ q_2(x) & = & \frac{1}{2}(3x^2-1) \\ q_3(x) & = & \frac{1}{2}(5x^3-3x) \\ q_4(x) & = & \frac{1}{8}(35x^4-30x^2+3) \\ q_5(x) & = & \frac{1}{8}(63x^5-70x^3+15x). \end{array}$$

Choosing $q = q_{n+1}$, the first term in (2.17) vanishes. Choosing $x_0, \ldots, x_{n+1}$ as zeros of q_{n+1} then the second term vanishes as well.

Theorem 2.13 *The Legendre polynomial q_{n+1} has $n+1$ simple zeros in $(-1,1)$.*

Proof. Let us define the set

$$N := \{\lambda \in (-1,1) : \lambda \text{ is a zero of } \textit{odd} \text{ multiplicity of } q_{n+1}\}$$

and

$$\begin{aligned} h(x) &:= 1 && \text{for } N = \emptyset, \quad \text{and} \\ h(x) &:= \prod_{i=1}^{m} (x - \lambda_i) && \text{for } N = \{\lambda_1, \dots, \lambda_m\}. \end{aligned}$$

Then $q_{n+1} \cdot h \in \mathbb{P}_{n+m+1}$ is real and all its roots in $(-1,1)$ have even order. In particular, it has *no* change of sign in $(-1,1)$ and therefore

$$(q_{n+1}, h) = \int_{-1}^{1} q_{n+1}(x)\, h(x)\ \mathrm{d}x \neq 0\,.$$

If $m \leq n$ this contradicts $q_{n+1} \perp \mathbb{P}_n$. In turn, $m > n$ and q_{n+1} has at least $n+1$ zeros in $(-1,1)$. By the fundamental lemma of algebra, q_{n+1} has *exactly* $n+1$ zeros in $(-1,1)$. □

The following theorem summarizes our observations.

Theorem 2.14 *Let $x_0, \dots, x_n \in (-1,1)$ be the zeros of the Legendre polynomials q_{n+1} let $\ell_0, \dots, \ell_n$ be the corresponding Lagrange polynomials. Choosing $\alpha_j = \int_{-1}^{1} \ell_j(x)\,\mathrm{d}x$ the* **Gauss formula**

$$Q^{(n)}[f] = \alpha_0 f(x_0) + \cdots + \alpha_n f(x_n)$$

has order $2n+2$.

For $n = 1$ and $n = 2$ we obtain the quadrature rules

$$\begin{aligned} Q^{(1)}[f] &= f(-\sqrt{1/3}) + f(\sqrt{1/3}), \\ Q^{(2)}[f] &= \frac{1}{9}\big\{5f(-\sqrt{3/5}) + 8f(0) + 5f(\sqrt{3/5})\big\}. \end{aligned}$$

Because of

$$0 < \int_{-1}^{1} \ell_i(x)^2\, dx = \sum_{j=0}^{n} \alpha_j \underbrace{\ell_i(x_j)^2}_{\delta_{ij}} = \alpha_i,$$

it follows that the weights are always positive and we avoid the numerical instability associated with high-order Newton-Cotes formulae. For general n, one can obtain the weights from the following linear system of equations:

$$\sum_{j=0}^{n} \alpha_j x_j^i = \int_{-1}^{1} x^i\ \mathrm{d}x = \frac{1}{i+1}(1 - (-1)^{i+1}), \quad i = 0, \dots, n.$$

The computation of the zeros of q_{n+1} is more difficult; the following result yields a simple implementation.[8]

Theorem 2.15 (Golub/Welsh) *The zeros $x_0, \dots, x_n$ of q_{n+1} are the eigenvalues of the matrix*

$$J = \begin{pmatrix} 0 & b_1 & & \\ b_1 & 0 & \ddots & \\ & \ddots & \ddots & b_n \\ & & b_n & 0 \end{pmatrix},$$

where

$$b_j = \frac{j}{\sqrt{4j^2-1}}.$$

Python

```python
import numpy as np
def gaussQuad(n):
    b = np.arange(1,n+1)
    b = b / np.sqrt(4*b*b-1)
    J = np.diag(b,-1) + np.diag(b,1)
    x, ev = np.linalg.eigh(J) # eigh stands for symmetric EVP
    a = np.array(2*(ev[0,:]*ev[0,:]))
    return x,a
```

The following theorem establishes an expression for the error of the Gauss formulae, which highlights (once more) their advantages compared to Newton-Cotes formulae.

Theorem 2.16 (Error of Gauss quadrature) *For $f \in C^{2n+2}[-1,1]$ there exists $\xi \in (-1,1)$ such that*

$$\begin{aligned} R^{(n)}[f] := Q^{(n)}[f] - \int_{-1}^{1} f(x)\,\mathrm{d}x &= \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \int_{-1}^{1} \prod_{j=0}^{n} (x-x_j)^2\,\mathrm{d}x \\ &= \frac{2^{2n+3}[(n+1)!]^4}{(2n+3)[(2n+2)!]^3}\, f^{(2n+2)}(\xi). \end{aligned}$$

Example 2.17 For the trapezoidal rule, Theorem 2.5 yields an error of the form $2/3\, f''(\xi)$ on the interval $[a,b] = [-1,1]$. For the Gauss quadrature for $n = 1$ (which requires the same number of function evaluations), Theorem 2.16 establishes an error of the form $1/135\, f^{(4)}(\xi)$. ◇

By a linear transformation, one obtains Gauss quadrature rules on an arbitrary

[8]See `https://pi.math.cornell.edu/~ajt/papers/QuadratureEssay.pdf` for a nice account of the history of methods for computing Gauss quadrature nodes/weights.

interval $[a,b]$. For $n=1$ and $n=2$ one obtains

$$Q^{(1)}[f] = \frac{b-a}{2}\left[f\big(c-\tilde{h}\sqrt{1/3}\big)+f\big(c+\tilde{h}\sqrt{1/3}\big)\right],$$
$$Q^{(2)}[f] = \frac{b-a}{18}\left[5f\big(c-\tilde{h}\sqrt{3/5}\big)+8f\big(c\big)+5f\big(c+\tilde{h}\sqrt{3/5}\big)\right],$$

with $c=\frac{b+a}{2}$ and $\tilde{h}=\frac{b-a}{2}$. Setting $x_j=a+jh$ for $j=0,\dots,N$ and $h=(b-a)/N$), the corresponding composite rules are given by

$$Q_h^{(1)}[f] = \frac{h}{2}\sum_{j=0}^{N-1}\left[f(x_j+h')+f(x_{j+1}-h')\right]$$

with $h'=(\frac{1}{2}-\frac{1}{2\sqrt{3}})\,h\sim 0.2113249\,h$, and

$$Q_h^{(2)}[f] = \frac{h}{18}\sum_{j=0}^{N-1}\left[5f(x_j+h')+8f\big(x_j+\tfrac{1}{2}\,h\big)+5f(x_{j+1}-h')\right]$$

with $h'=(\frac{1}{2}-\frac{1}{2}\sqrt{\frac{3}{5}})\,h\sim 0.1127012\,h$.

2.6 Miscellaneous*

2.6.1 Periodic functions

The composite trapezoidal rule has excellent convergence properties for (smooth) periodic functions. For example, consider the function

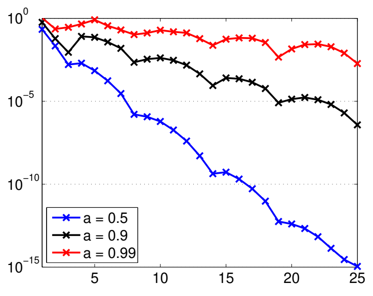
$$f(x)=\frac{1}{\sqrt{1-a\sin(x-1)}}, \tag{2.20}$$

which is periodic on the interval $[0,2\pi]$. Für $a=1$, the function has a singularity at $2\pi/4+1\approx 2.57$.

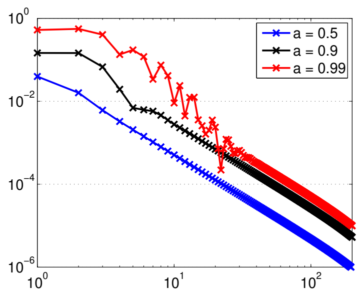
In the left part of Figure 2.5, the interval is chosen such that it matches a period of the function. Note that the x axis is *not* scaled logarithmically; the quadrature error converges *exponentially fast* to 0 as N increases. The speed of convergence clearly depends on a; as a gets very close 1 one suffers from the non-smoothness of the function for $a=1$. In the right part of Figure 2.5 the interval does not match the period of the function. Note that the x axis is scaled logarithmically. In this situation, the quadrature error converges much more slowly and in accordance with the result of Theorem 2.6. On the other hand, choosing a close to 1 has a less dramatic impact on the convergence speed.

The fast convergence of the composite trapezoidal rule for a periodic function is connected to favorable properties of the Fourier expansion, which will be discussed later on.

For a periodic function, the composite trapezoidal rule is the method of choice. It would do more harm than good to choose higher order quadrature rules.



$\int_0^{2\pi} f(x)\,\mathrm{d}x$



$\int_0^{\pi} f(x)\,\mathrm{d}x$

Figure 2.5: Quadrature error vs. N for composite trapezoidal rule applied to integrating f from (2.20) for $a \in \{0.5, 0.9, 0.99\}$ on two different intervals.

2.6.2 Singular integrals

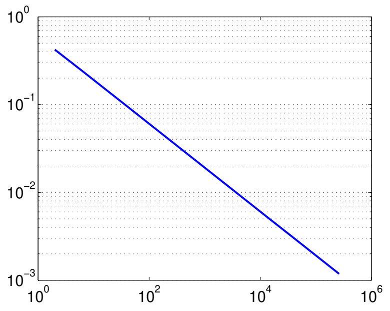
Functions with singularities (leading to singular/improper integrals) can also be integrated numerically, but some care is needed in the choice of quadrature rule. For example, the composite midpoint rule converges for

$$\int_0^1 \frac{1}{\sqrt{x}}\,\mathrm{d}x = 2.$$

Python

```
import numpy as np
import matplotlib.pyplot as plt
def sing(x):
    return 1/np.sqrt(x)
def midpoint(fun,a,b,n):
    h = (b-a)/n;
    x = np.arange(a+h/2,b-h/2+h,h)
    fun_vec = np.vectorize(fun)
    f = fun_vec(x)
    return h * sum(f)
nn = 2**np.arange(1,19,1)
err = []
for n in nn:
    err.append(abs(2 -
    midpoint(sing,0,1,n) ))
plt.loglog(nn,err)
```

The following figure shows the observed error vs. N:



The observed convergence order 1/2 is not very satisfying; halving the error requires to increase the number of points by a factor four. More effective approaches to singularities are variable transformation techniques or adaptive quadrature.

2.6.3 Two-dimensional integrals

In practice, one is often interested in integrals in 2D, 3D or, more generally, on domains in $\mathbb{R}^d$. By domain decomposition and transformation one typically reduces such problems to standard domains like the unit cube. To illustrate the development of quadrature rules for such standard domains we discuss the unit square and the unit triangle.

Unit square. Consider the integral

$$\int_0^1 \int_0^1 f(x,y)\,\mathrm{d}x\,\mathrm{d}y. \tag{2.21}$$

Let

$$Q_m[g] = \sum_{i=0}^{m} \alpha_i g(x_i)$$

be a quadrature rule for $\int_0^1 g(x)\ \mathrm{d}x$. Denoting

$$F(y) = \int_0^1 f(x,y)\ \mathrm{d}x,$$

we obtain from the application of Q_m to the x and y variables the following product quadrature rule $Q^{(m\times m)}_{[0,1]\times[0,1]}[f]$:

$$\begin{aligned}
\int_0^1 \int_0^1 f(x,y)\ \mathrm{d}x\ \mathrm{d}y &= \int_0^1 F(y)\ \mathrm{d}y \approx \sum_{j=0}^{m} \alpha_j F(x_j) \\
&= \sum_{j=0}^{m} \alpha_j \int_0^1 f(x,x_j)\ \mathrm{d}x \approx \sum_{j=0}^{m} \alpha_j \sum_{i=0}^{m} \alpha_i f(x_i,x_j) \\
&= \sum_{i,j=0}^{m} \alpha_i \alpha_j f(x_i,x_j) =: Q_{m\times m}[f].
\end{aligned}$$

For the hypercube in $\mathbb{R}^d$ the approach above would result in m^d terms. Hence, the cost grows very quickly with d. This so called *curse of dimensionality* can sometimes be countered with *Monte Carlo methods* or *sparse grids.*

Unit triangle. The approach for the unit square has no meaningful extension to triangles. Instead one aims at finding quadrature rule which exactly integrate $x^{k_1}y^{k_2}$, $k_1 + k_2 \le M$ for a certain integer M. Typical examples for the unit triangle $\{(x,y) : x + y \le 1\}$:

1. $Q[f] = \frac{1}{2} f(1/3, 1/3)$,
2. $Q[f] = \frac{1}{6}\big[f(0,0) + f(1,0) + f(0,1)\big]$,
3. $Q[f] = \frac{1}{6}\big[f(1/2,0) + f(0,1/2) + f(1/2,1/2)\big]$,

4. $Q[f] = \frac{1}{6}\big[f(1/6, 1/6) + f(2/3, 1/6) + f(1/6, 2/3)\big]$.

One verifies that 1 and 2 exactly integrate $1, x, y$ ($M = 1$), while 3 and 4 exactly integrate $1, x, y, xy, x^2, y^2$ ($M = 2$).