

# 1 Introduction to Numerical Integration & Quadrature

Suppose we want to evaluate the definite integral of a function  $f(x)$  from  $a$  to  $b$ , denoted as  $I(f) = \int_a^b f(x)dx$ . Numerical integration, also known as quadrature, provides methods to approximate this integral.

A common approach to numerical integration is to approximate the function  $f(x)$  with a polynomial interpolant  $p_n(x)$  and then integrate this polynomial exactly. Let  $Q_n(f)$  be the quadrature rule obtained by integrating  $p_n(x)$  from  $a$  to  $b$ :

$$Q_n(f) = \int_a^b p_n(x)dx$$

We hope that  $Q_n(f)$  will be a good approximation to  $I(f)$ .

## 2 Newton-Cotes Quadrature Rules

Newton-Cotes formulas are a family of quadrature rules based on polynomial interpolation at equally spaced points over the interval  $[a, b]$ . For a Newton-Cotes formula of order  $n$ , we use  $n + 1$  equally spaced points  $x_0, x_1, \dots, x_n$  in  $[a, b]$ . Let  $x_k = a + kh$ , where  $h = \frac{b-a}{n}$  and  $k = 0, 1, \dots, n$ . We can express the interpolant  $p_n(x)$  in Lagrange form:

$$p_n(x) = \sum_{k=0}^n f(x_k)L_k(x)$$

where  $L_k(x)$  are the Lagrange polynomials associated with the points  $x_k$ . Then, the Newton-Cotes quadrature rule is:

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

We define the quadrature weights  $w_k$  as:

$$w_k = \int_a^b L_k(x)dx$$

Thus, the Newton-Cotes quadrature rule becomes:

$$Q_n(f) = \sum_{k=0}^n w_k f(x_k)$$

### 2.1 n=1 (Trapezoid Rule)

For  $n = 1$ , we have two quadrature points  $x_0 = a$  and  $x_1 = b$ . The Lagrange polynomials are:

$$L_0(x) = \frac{x - x_1}{x_0 - x_1} = \frac{x - b}{a - b}$$

$$L_1(x) = \frac{x - x_0}{x_1 - x_0} = \frac{x - a}{b - a}$$

The quadrature weights are:

$$w_0 = \int_a^b L_0(x)dx = \int_a^b \frac{x - b}{a - b}dx$$

$$w_1 = \int_a^b L_1(x)dx = \int_a^b \frac{x - a}{b - a}dx$$

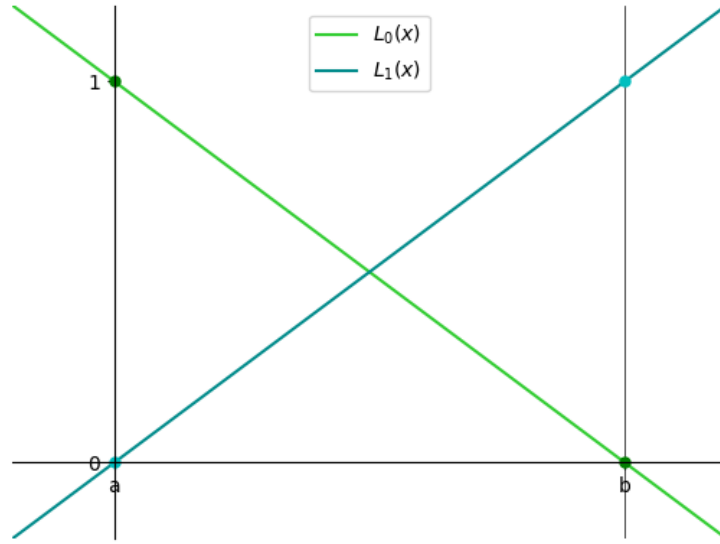


Figure 1: Lagrange polynomials  $L_0(x)$  and  $L_1(x)$  for  $n=1$  Newton-Cotes

Let's calculate  $w_0$ . We use the substitution  $z = b - x$ , so  $dz = -dx$ . When  $x = a$ ,  $z = b - a$ , and when  $x = b$ ,  $z = 0$ .

$$\begin{aligned}
 w_0 &= \int_{b-a}^0 \frac{(b-z)-b}{a-b} (-dz) \\
 &= \int_0^{b-a} \frac{-z}{a-b} dz \\
 &= \int_0^{b-a} \frac{z}{b-a} dz \\
 &= \frac{1}{b-a} \left[ \frac{z^2}{2} \right]_0^{b-a} \\
 &= \frac{1}{b-a} \frac{(b-a)^2}{2} = \frac{b-a}{2}
 \end{aligned}$$

Similarly, for  $w_1$ , using the substitution  $z = x - a$ ,  $dz = dx$ . When  $x = a$ ,  $z = 0$ , and when  $x = b$ ,  $z = b - a$ .

$$\begin{aligned}
 w_1 &= \int_0^{b-a} \frac{z}{b-a} dz \\
 &= \frac{1}{b-a} \left[ \frac{z^2}{2} \right]_0^{b-a} \\
 &= \frac{1}{b-a} \frac{(b-a)^2}{2} = \frac{b-a}{2}
 \end{aligned}$$

Thus, the Trapezoid rule is:

$$Q_1(f) = w_0 f(x_0) + w_1 f(x_1) = \frac{b-a}{2} [f(a) + f(b)]$$

This is the familiar trapezoid rule, which approximates the integral by the area of the trapezoid under the curve.

**Error Estimate for Trapezoid Rule** The error in Newton-Cotes quadrature is given by the integral of the interpolation error:

$$E_n(f) = I(f) - Q_n(f) = \int_a^b [f(x) - p_n(x)] dx$$

We know from interpolation theory that:

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\theta)}{(n+1)!} \prod_{i=0}^n (x - x_i)$$

for some  $\theta \in [a, b]$ . For  $n = 1$ , we have:

$$|E_1(f)| \leq \frac{M_2}{2!} \int_a^b |(x - x_0)(x - x_1)| dx = \frac{M_2}{2} \int_a^b |(x - a)(x - b)| dx$$

where  $M_2 = \max_{x \in [a, b]} |f''(x)|$ . Since  $(x - a)(x - b) \leq 0$  for  $x \in [a, b]$ , we have  $|(x - a)(x - b)| = -(x - a)(x - b) = (a - x)(x - b)$ .

$$\int_a^b (a - x)(x - b) dx = - \int_a^b (x - a)(x - b) dx$$

Let  $z = x - a$ ,  $dz = dx$ . When  $x = a$ ,  $z = 0$ , when  $x = b$ ,  $z = b - a$ .

$$\begin{aligned} - \int_0^{b-a} z(z - (b - a)) dz &= - \int_0^{b-a} (z^2 - z(b - a)) dz \\ &= - \left[ \frac{z^3}{3} - \frac{z^2}{2}(b - a) \right]_0^{b-a} \\ &= - \left[ \frac{(b - a)^3}{3} - \frac{(b - a)^2}{2}(b - a) \right] \\ &= -(b - a)^3 \left[ \frac{1}{3} - \frac{1}{2} \right] \\ &= -(b - a)^3 \left[ \frac{2 - 3}{6} \right] = \frac{(b - a)^3}{6} \end{aligned}$$

Thus, the error bound for the trapezoid rule is:

$$|E_1(f)| \leq \frac{M_2}{2} \frac{(b - a)^3}{6} = \frac{M_2}{12} (b - a)^3$$

## 2.2 n=2 (Simpson's Rule)

For  $n = 2$ , we have three quadrature points  $x_0 = a$ ,  $x_1 = \frac{a+b}{2}$ , and  $x_2 = b$ . The Simpson's rule is given by:

$$Q_2(f) = \frac{b - a}{6} [f(a) + 4f(\frac{a+b}{2}) + f(b)]$$

Simpson's rule generally provides a more accurate approximation than the trapezoid rule because it uses a quadratic polynomial interpolation instead of a linear one.

## 2.3 Runge Phenomenon and Higher-order Newton-Cotes

While we can develop Newton-Cotes formulas for higher values of  $n$ , using high-degree polynomial interpolation with equally spaced points can lead to oscillations, especially near the edges of the interval. This is known as the Runge phenomenon. As  $n$  increases, the Lagrange polynomials  $L_k(x)$  for equally spaced points can become very large in magnitude between the nodes, especially near the endpoints of the interval.

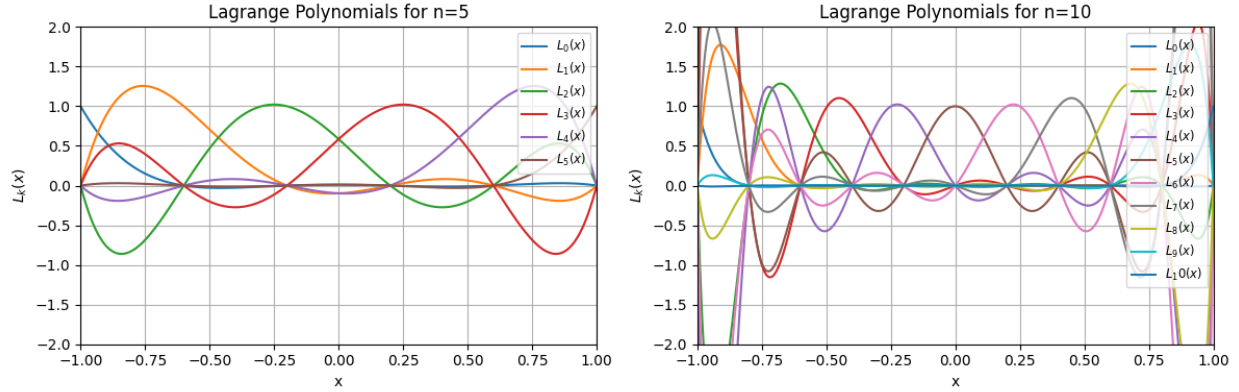


Figure 2: Lagrange Polynomials for  $n=5$  and  $n=10$  showing oscillations and blow-up near the interval ends.

This blow-up of Lagrange polynomials for equally spaced points is directly related to the term  $C_n$  in the error bound for quadrature rules that integrate polynomials of degree  $n$  exactly:

$$|E_n(f)| \leq C_n \min_{p \in P_n} \|f - p\|_\infty$$

where  $C_n = (b - a) + \sum_{k=0}^n |w_k|$  and  $w_k = \int_a^b L_k(x) dx$ . If the quadrature weights  $w_k$  become negative and large in magnitude (which happens for higher-order Newton-Cotes), the constant  $C_n$  can grow rapidly with  $n$ . This growth in  $C_n$ , combined with the Runge phenomenon, makes high-order Newton-Cotes formulas with equally spaced points unreliable for high degrees.

However, the sum of quadrature weights without absolute values is always well-behaved. For the constant function  $f(x) = 1$ , we must have  $\int_a^b 1 dx = \sum_{k=0}^n w_k \cdot 1$ , so  $\sum_{k=0}^n w_k = b - a$ . If all  $w_k > 0$ , then  $C_n = (b - a) + \sum_{k=0}^n w_k = 2(b - a)$ , a constant independent of  $n$ . But for Newton-Cotes with large  $n$ , some  $w_k$  become negative, causing  $\sum_{k=0}^n |w_k|$  to blow up. Newton-Cotes formulas with  $n > 8$  have negative weights and are generally not useful for large  $n$  due to error accumulation.

## 2.4 Addressing Runge Phenomenon

Runge phenomenon arises from using high-degree polynomial interpolation at equally spaced points. Ways to address this include:

- **Piecewise Polynomial Interpolation (Composite Rules):** Instead of using a single high-degree polynomial over the entire interval, we can divide the interval into smaller subintervals and use low-degree polynomials (like linear or quadratic) on each subinterval. This leads to composite quadrature rules.
- **Non-Equally Spaced Points (Gauss Quadrature):** Choosing interpolation points strategically, such as the roots of orthogonal polynomials, can minimize oscillations and significantly improve accuracy. This leads to Gauss quadrature.

## 3 Composite Quadrature Rules

### 3.1 Definition and Motivation

Composite quadrature rules are constructed by dividing the interval  $[a, b]$  into  $m$  subintervals and applying a simple quadrature rule (like trapezoid or Simpson's rule) on each subinterval. This piecewise approach mitigates the Runge phenomenon and improves convergence for general functions.

Divide  $[a, b]$  into  $m$  equally spaced subintervals  $[x_{i-1}, x_i]$  for  $i = 1, 2, \dots, m$ , where  $x_i = a + ih$  and  $h = \frac{b-a}{m}$ . Then,

$$\int_a^b f(x)dx = \sum_{i=1}^m \int_{x_{i-1}}^{x_i} f(x)dx$$

We approximate the integral on each subinterval using a simple quadrature rule.

### 3.2 Composite Trapezoid Rule

Apply the trapezoid rule on each subinterval  $[x_{i-1}, x_i]$ :

$$\int_{x_{i-1}}^{x_i} f(x)dx \approx \frac{h}{2}[f(x_{i-1}) + f(x_i)]$$

Summing over all subintervals, we get the composite trapezoid rule:

$$Q_{1,h}(f) = \sum_{i=1}^m \frac{h}{2}[f(x_{i-1}) + f(x_i)] = h \left[ \frac{1}{2}f(x_0) + \sum_{i=1}^{m-1} f(x_i) + \frac{1}{2}f(x_m) \right]$$

**Error Analysis for Composite Trapezoid Rule** Let  $E_{1,h}(f)$  be the error of the composite trapezoid rule. The error on each subinterval  $[x_{i-1}, x_i]$  is bounded by  $\frac{M_2^{(i)}}{12}h^3$ , where  $M_2^{(i)} = \max_{x \in [x_{i-1}, x_i]} |f''(x)|$ . Then, the error for the composite rule is bounded by the sum of errors on each subinterval:

$$\begin{aligned} |E_{1,h}(f)| &\leq \sum_{i=1}^m \frac{M_2^{(i)}}{12}h^3 \\ &\leq \sum_{i=1}^m \frac{M_2}{12}h^3 = m \frac{M_2}{12}h^3 \end{aligned}$$

where  $M_2 = \max_{x \in [a,b]} |f''(x)|$ . Since  $m = \frac{b-a}{h}$ , we have:

$$|E_{1,h}(f)| \leq \frac{b-a}{h} \frac{M_2}{12}h^3 = \frac{M_2}{12}(b-a)h^2$$

Thus, the error of the composite trapezoid rule is  $O(h^2)$ .

### 3.3 Composite Simpson's Rule

To apply composite Simpson's rule, we divide  $[a, b]$  into  $2m$  intervals (so  $m$  applications of Simpson's rule) with step size  $h = \frac{b-a}{2m}$ . Let  $x_i = a + ih$ . Apply Simpson's rule on each interval  $[x_{2i-2}, x_{2i}]$  for  $i = 1, 2, \dots, m$ :

$$\int_{x_{2i-2}}^{x_{2i}} f(x)dx \approx \frac{2h}{6}[f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})] = \frac{h}{3}[f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})]$$

Summing over all intervals, we get the composite Simpson's rule:

$$\begin{aligned} Q_{2,h}(f) &= \frac{h}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \dots + 4f(x_{2m-1}) + f(x_{2m})] \\ Q_{2,h}(f) &= \frac{h}{3} \left[ f(x_0) + f(x_{2m}) + 4 \sum_{i=1}^m f(x_{2i-1}) + 2 \sum_{i=1}^{m-1} f(x_{2i}) \right] \end{aligned}$$

The expected error for Simpson's rule is  $O(h^4)$ . However, in practice, the observed convergence rate for composite Simpson's rule can be  $O(h^4)$ , not just  $O(h^3)$  as a naive error bound might suggest. This is due to a cancellation of leading-order error terms due to symmetry in Simpson's rule. Although the local error for Simpson's rule is typically  $O(h^5)$ , when summed over the interval, it results in a global error of  $O(h^4)$ . More refined error analysis shows the error bound is proportional to  $h^4 M_4$ , where  $M_4 = \max |f^{(4)}(x)|$ .

### 3.4 Adaptive Quadrature

Adaptive quadrature methods aim to achieve a desired accuracy with minimal function evaluations. The basic principle is to refine the subdivision of the interval only where the function is "difficult" to integrate (i.e., where the error is large).

**Principle** Start with a coarse interval subdivision. Estimate the error on each subinterval. If the estimated error on a subinterval is greater than a given tolerance, subdivide that interval further and refine the quadrature. Repeat until the desired accuracy is achieved over the entire interval.

**Error Estimation** A common way to estimate the error on an interval  $I$  is to compare the result of a quadrature rule  $Q_h(f)$  with a more refined quadrature rule  $Q_{\hat{h}}(f)$  on the same interval. We can approximate the error as:

$$|I(f) - Q_h(f)| \approx |Q_{\hat{h}}(f) - Q_h(f)|$$

assuming  $Q_{\hat{h}}(f)$  is a much more accurate approximation to  $I(f)$  than  $Q_h(f)$ . For instance, we can compare the trapezoid rule with Simpson's rule, or Simpson's rule with a higher-order rule. Alternatively, we can halve the interval size and compare the results.

**Python and MATLAB 'quad' Functions** Both Python (SciPy) and MATLAB provide 'quad' functions for adaptive quadrature. MATLAB's 'quad' function implements an adaptive Simpson's rule. These functions recursively subdivide intervals until a specified error tolerance is met.

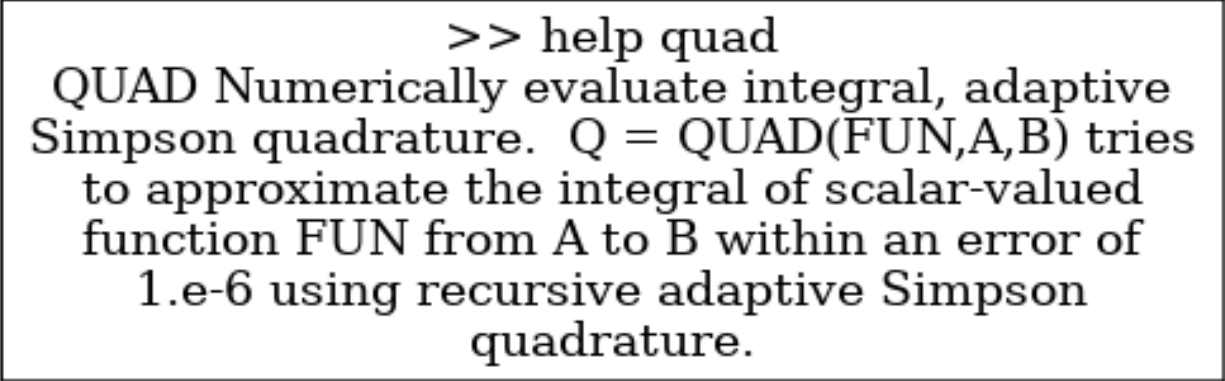
A rectangular box with a thin black border containing MATLAB help text for the 'quad' function. The text is centered and uses a serif font. It starts with a prompt '>> help quad' followed by a description of the QUAD function and its usage: 'QUAD Numerically evaluate integral, adaptive Simpson quadrature. Q = QUAD(FUN,A,B) tries to approximate the integral of scalar-valued function FUN from A to B within an error of 1.e-6 using recursive adaptive Simpson quadrature.'

Figure 3: MATLAB help text for quad function.

## 4 Gauss Quadrature

### 4.1 Motivation for Optimal Quadrature Point Selection

Gauss quadrature aims to maximize the degree of polynomials that can be integrated exactly using a given number of quadrature points. For an  $(n + 1)$ -point quadrature rule, we have  $2n + 2$  degrees of freedom (choosing  $n + 1$  weights and  $n + 1$  points). We can hope to integrate polynomials of degree up to  $2n + 1$  exactly, which is a significant improvement over Newton-Cotes rules that integrate polynomials up to degree  $n$  exactly (in general).

### 4.2 Orthogonal Polynomials (Legendre Polynomials)

Gauss quadrature points are chosen to be the roots of orthogonal polynomials. Legendre polynomials  $P_n(x)$  are a set of orthogonal polynomials on the interval  $[-1, 1]$  with respect to the weight function  $w(x) = 1$ .

**Definition and Properties** The Legendre polynomials  $\{P_n(x)\}_{n=0}^{\infty}$  are orthogonal with respect to the inner product:

$$\langle P_m, P_n \rangle = \int_{-1}^1 P_m(x) P_n(x) dx = \begin{cases} \frac{2}{2n+1} & \text{if } m = n \\ 0 & \text{if } m \neq n \end{cases}$$

**Recurrence Relation** Legendre polynomials satisfy the three-term recurrence relation:

$$P_0(x) = 1$$

$$P_1(x) = x$$

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x) \quad \text{for } n \geq 1$$

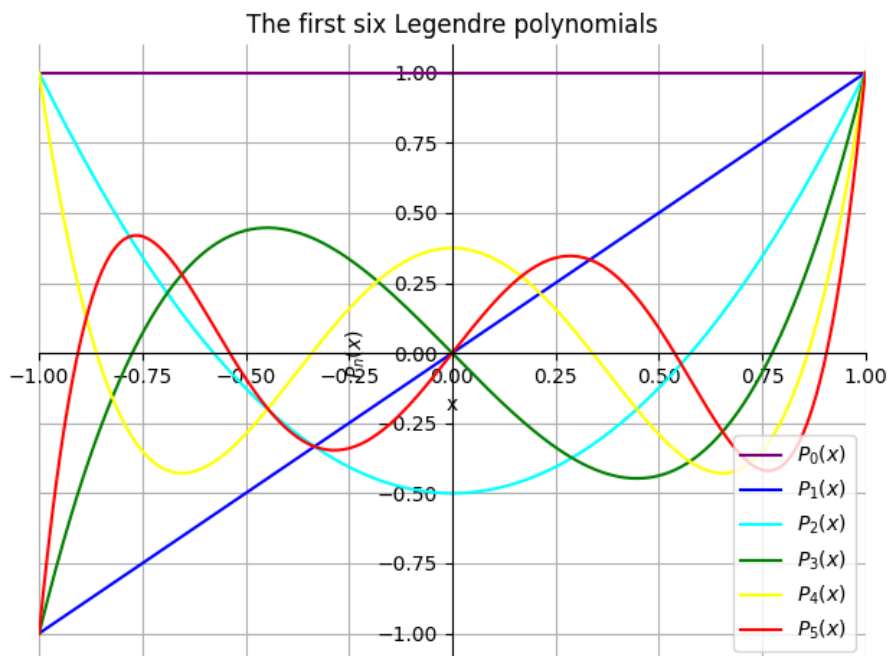


Figure 4: The first six Legendre polynomials.

### 4.3 Gauss Quadrature Points as Roots of Legendre Polynomials

The nodes  $x_0, x_1, \dots, x_n$  for  $(n+1)$ -point Gauss quadrature on  $[-1, 1]$  are chosen to be the roots of the Legendre polynomial  $P_{n+1}(x)$ . The weights  $w_k$  are then chosen such that the quadrature rule is exact for polynomials of degree up to  $2n+1$ . Gauss quadrature points are not equally spaced; they are clustered towards the endpoints of the interval  $[-1, 1]$ . This clustering helps in achieving higher accuracy and mitigating Runge phenomenon.

### 4.4 Why Gauss Quadrature Integrates Polynomials up to Degree $2n+1$ Exactly

Consider a polynomial  $f(x)$  of degree at most  $2n+1$ . We can use polynomial long division to write  $f(x)$  as:

$$f(x) = p(x)P_{n+1}(x) + r(x)$$

where  $p(x)$  and  $r(x)$  are polynomials of degree at most  $n$ . Now consider the weighted integral with weight function  $w(x) = 1$  (for Legendre polynomials):

$$\int_{-1}^1 f(x)w(x)dx = \int_{-1}^1 p(x)P_{n+1}(x)w(x)dx + \int_{-1}^1 r(x)w(x)dx$$

Since  $\deg(p) \leq n$  and  $P_{n+1}(x)$  is orthogonal to all polynomials of degree  $\leq n$ , we have:

$$\int_{-1}^1 p(x)P_{n+1}(x)w(x)dx = 0$$

So,

$$\int_{-1}^1 f(x)w(x)dx = \int_{-1}^1 r(x)w(x)dx$$

Now consider the Gauss quadrature approximation using the roots of  $P_{n+1}(x)$  as nodes.

$$Q_n(f) = \sum_{k=0}^n w_k f(x_k) = \sum_{k=0}^n w_k [p(x_k)P_{n+1}(x_k) + r(x_k)]$$

Since  $x_k$  are roots of  $P_{n+1}(x)$ ,  $P_{n+1}(x_k) = 0$ . Thus,

$$Q_n(f) = \sum_{k=0}^n w_k r(x_k)$$

Since  $r(x)$  is a polynomial of degree at most  $n$ , the  $(n+1)$ -point Newton-Cotes quadrature (and Gauss quadrature which also uses Lagrange interpolation concept to determine weights) integrates  $r(x)$  exactly. Hence,

$$Q_n(f) = \sum_{k=0}^n w_k r(x_k) = \int_{-1}^1 r(x)w(x)dx = \int_{-1}^1 f(x)w(x)dx$$

Therefore, Gauss quadrature integrates polynomials of degree up to  $2n+1$  exactly.

## 4.5 Generalizations of Gauss Quadrature

We can generalize Gauss quadrature by choosing different weight functions  $w(x)$  and corresponding orthogonal polynomials. For example, using the weight function  $w(x) = \frac{1}{\sqrt{1-x^2}}$  on  $[-1, 1]$  leads to Chebyshev polynomials  $T_n(x)$  as orthogonal polynomials. The roots of Chebyshev polynomials can be used to construct Gauss-Chebyshev quadrature, which is particularly useful for integrals with singularities of the form  $\frac{1}{\sqrt{1-x^2}}$  near the endpoints.

Chebyshev roots are closer to the ends than Legendre roots, leading to better sampling of functions near  $\pm 1$ , as expected based on the weight function  $w(x) = \frac{1}{\sqrt{1-x^2}}$  which emphasizes the endpoints.

## 5 Conclusion

Numerical integration techniques offer various ways to approximate definite integrals. Newton-Cotes rules are simple to derive but suffer from Runge phenomenon for higher orders. Composite rules improve accuracy and stability by using piecewise low-degree polynomials. Adaptive quadrature refines the integration adaptively based on error estimation. Gauss quadrature provides optimal accuracy for a given number of points by strategically selecting quadrature points as roots of orthogonal polynomials, achieving exact integration for polynomials of very high degree. The choice of method depends on the function's properties, desired accuracy, and computational cost.



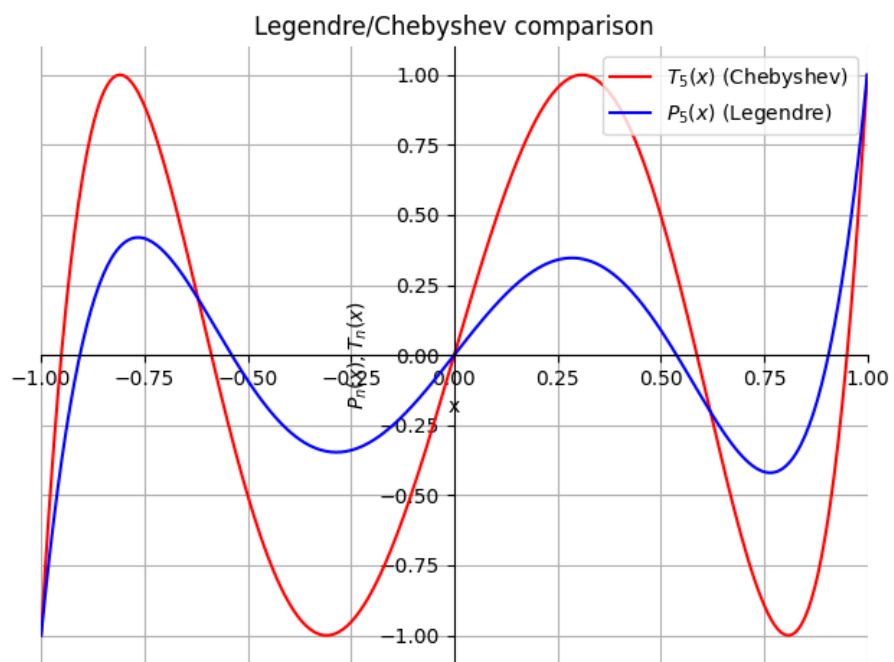


Figure 5: Comparison of Chebyshev and Legendre Polynomials of degree 5.