Simulation and High-Performance Computing

Part 3: Higher-order Methods

Steffen Börm

Christian-Albrechts-Universität zu Kiel

September 29th, 2020

Higher-order methods

Observation:

- First-order convergence: Error proportional to δ.
  Examples: Implicit and explicit Euler methods.
- Second-order convergence: Error proportional to δ^2.
  Examples: Runge's method, Crank-Nicolson, leapfrog method.

Goal: n-th order convergence, error proportional to δ^n.

Higher-order methods

Observation:

- First-order convergence: Error proportional to δ.
  Examples: Implicit and explicit Euler methods.
- Second-order convergence: Error proportional to δ^2.
  Examples: Runge's method, Crank-Nicolson, leapfrog method.

Goal: n-th order convergence, error proportional to δ^n.

Idea: Euler methods are exact for linear solutions, Crank-Nicolson and the leapfrog method are exact for quadratic solutions.
$\rightarrow$ Look for methods that are exact for polynomials of high degree m.

$$p(t) = a_0 + a_1 t + \ldots + a_m t^m.$$

Polynomial approximation

Goal: Approximate a given function f by a polynomial p of degree m.

Taylor polynomial around a center t_0 given by

$$p(t) = \sum_{k=0}^{m} (t - t_0)^k \frac{f^{(k)}(t_0)}{k!}.$$

Polynomial approximation

Goal: Approximate a given function f by a polynomial p of degree m.

Taylor polynomial around a center t_0 given by

$$p(t) = \sum_{k=0}^{m} (t - t_0)^k \frac{f^{(k)}(t_0)}{k!}.$$

Problem: We need the first k derivatives to compute p.

Polynomial approximation

Goal: Approximate a given function f by a polynomial p of degree m.

Taylor polynomial around a center t_0 given by

$$p(t) = \sum_{k=0}^{m} (t - t_0)^k \frac{f^{(k)}(t_0)}{k!}.$$

Problem: We need the first k derivatives to compute p.

Interpolation: Given $t_0, \ldots, t_m$, find p of degree m with

$$p(t_j) = f(t_j) \qquad \text{for all } j \in [0 : m].$$

Polynomial approximation

Goal: Approximate a given function f by a polynomial p of degree m.

Taylor polynomial around a center t_0 given by

$$p(t) = \sum_{k=0}^{m} (t - t_0)^k \frac{f^{(k)}(t_0)}{k!}.$$

Problem: We need the first k derivatives to compute p.

Interpolation: Given $t_0, \ldots, t_m$, find p of degree m with

$$p(t_j) = f(t_j) \qquad \text{for all } j \in [0 : m].$$

Problem: p is given implicitly, i.e., as the solution of equations.

Lagrange polynomials

Idea: If we can find polynomials $(\ell_k)_{k=0}^m$ satisfying

$$\ell_k(t_j) = \begin{cases} 1 & \text{if } k = j, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } j, k \in [0 : m],$$

we can represent p explicitly as

$$p(t) = \sum_{k=0}^{m} f(t_k)\ell_k(t) \qquad \text{since} \qquad p(t_j) = \sum_{k=0}^{m} f(t_k)\ell_k(t_j) = f(t_j).$$

Lagrange polynomials

Idea: If we can find polynomials $(\ell_k)_{k=0}^m$ satisfying

$$\ell_k(t_j) = \begin{cases} 1 & \text{if } k = j, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } j, k \in [0 : m],$$

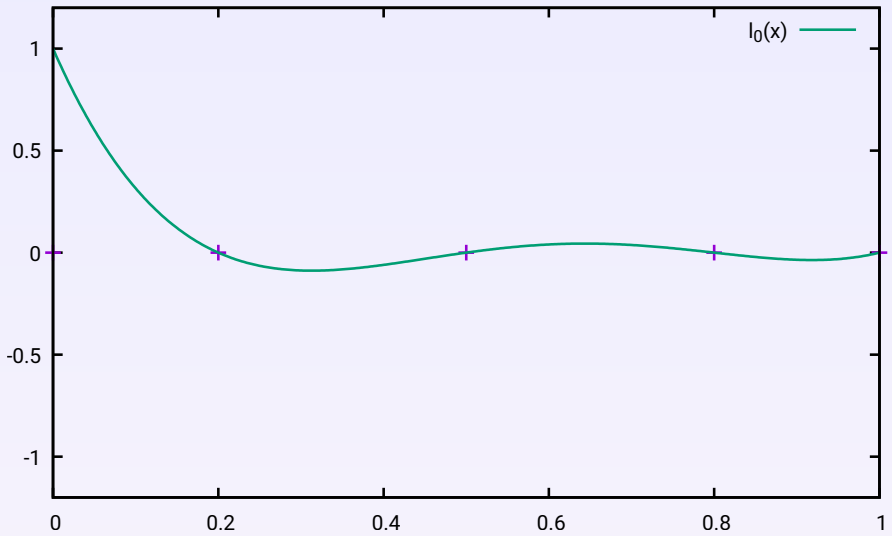
we can represent p explicitly as

$$p(t) = \sum_{k=0}^{m} f(t_k)\ell_k(t) \qquad \text{since} \qquad p(t_j) = \sum_{k=0}^{m} f(t_k)\ell_k(t_j) = f(t_j).$$

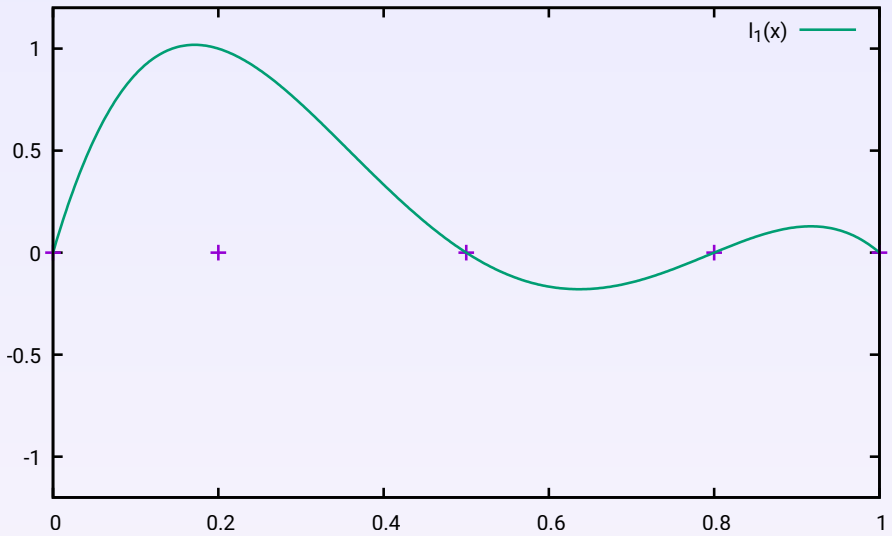
Construction: Multiply linear factors to create the required zeros.

$$\ell_k(t) = \prod_{\substack{i=0 \\ i \neq k}}^{m} \frac{t - t_i}{t_k - t_i}.$$

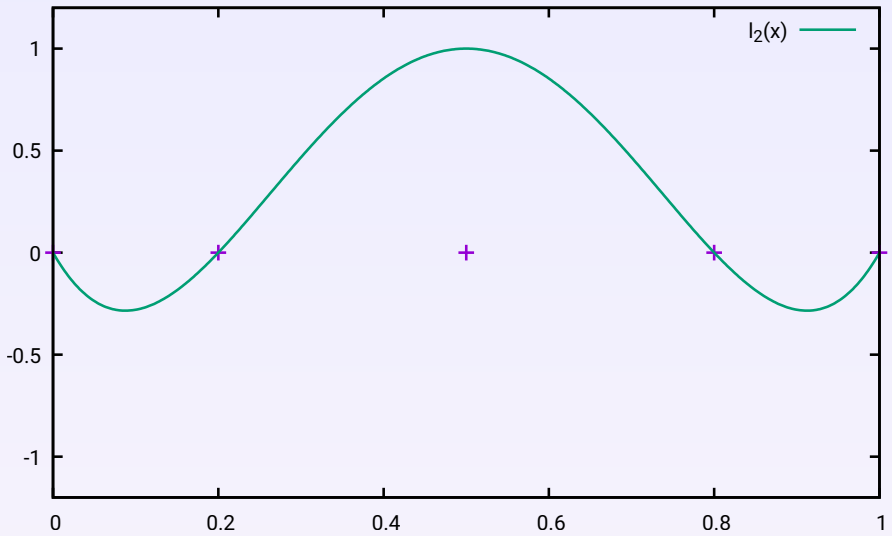
Lagrange representation



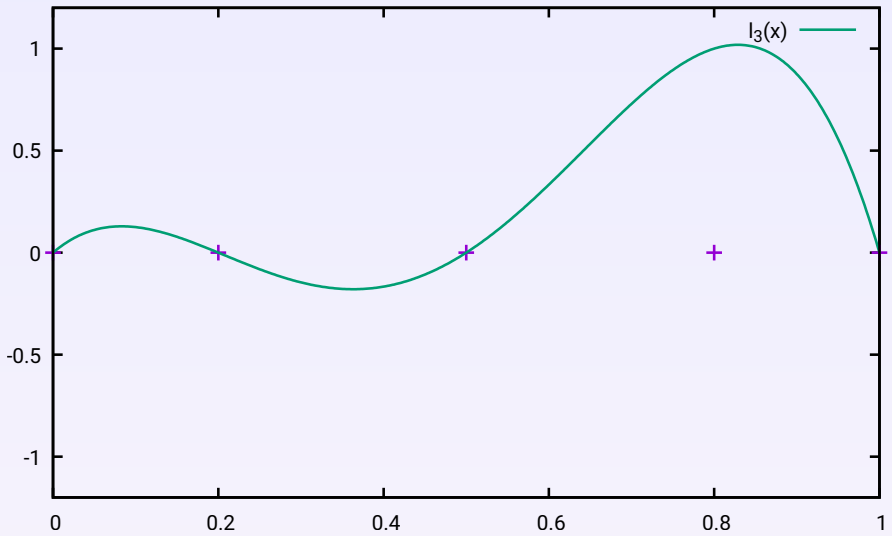
Lagrange representation



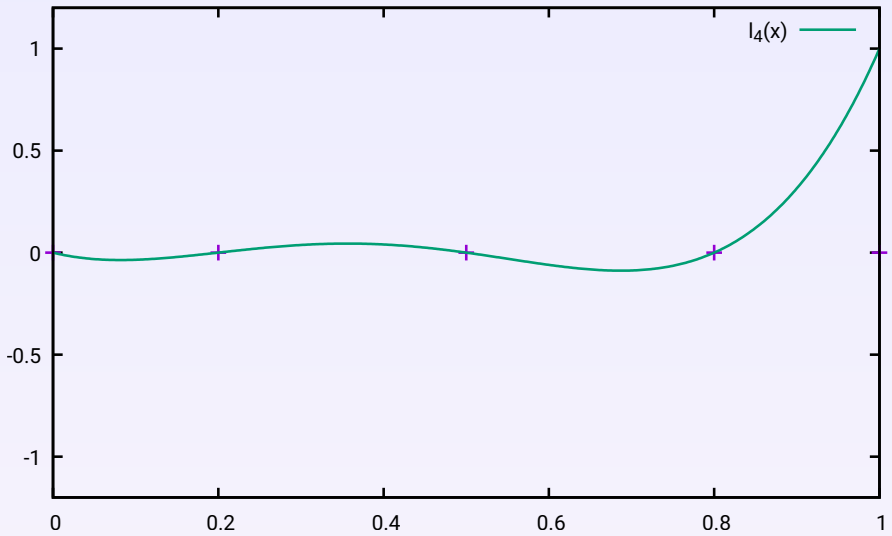
Lagrange representation



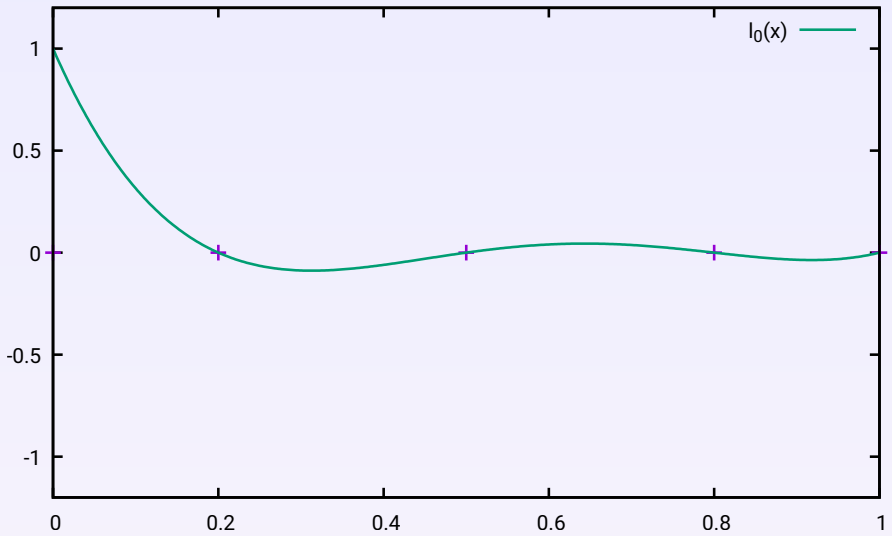
Lagrange representation

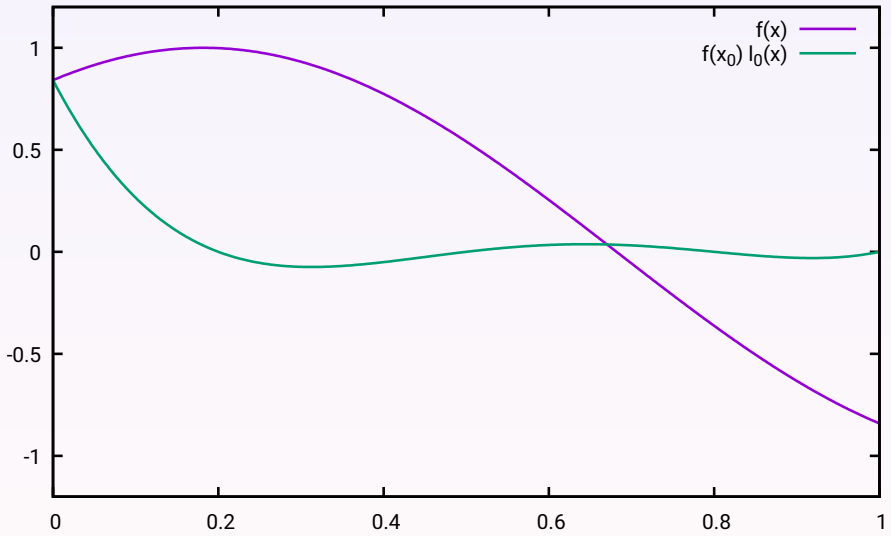


Lagrange representation

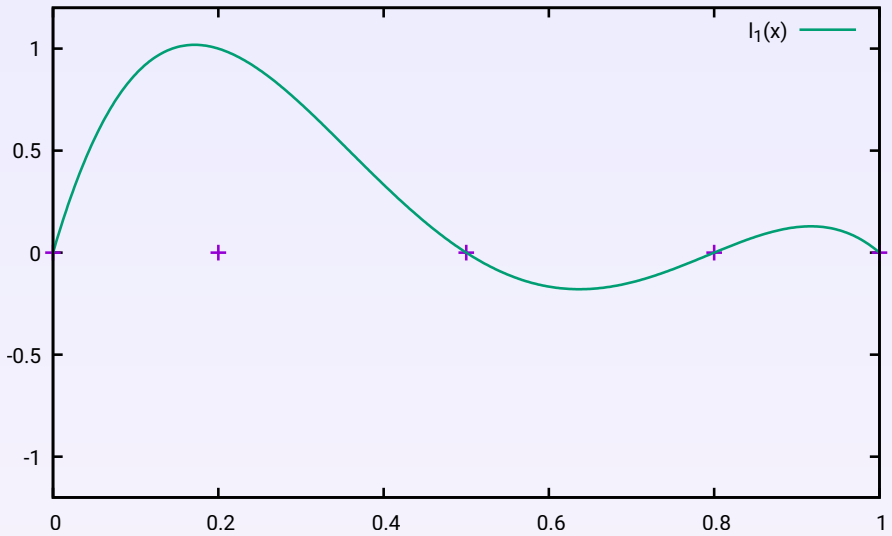


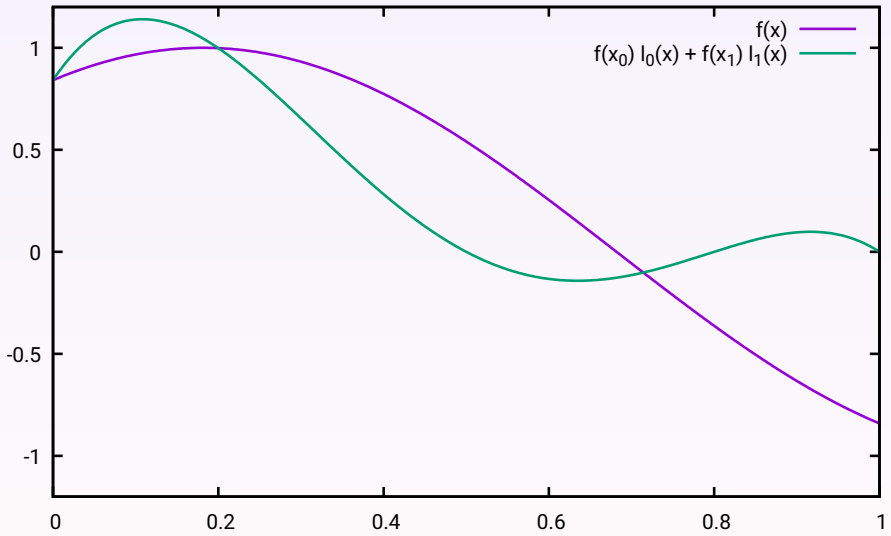
Lagrange representation



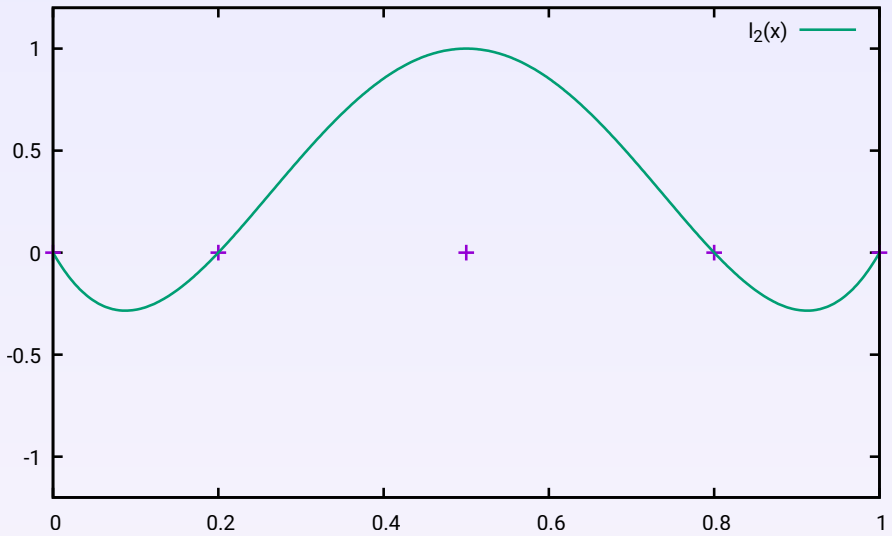


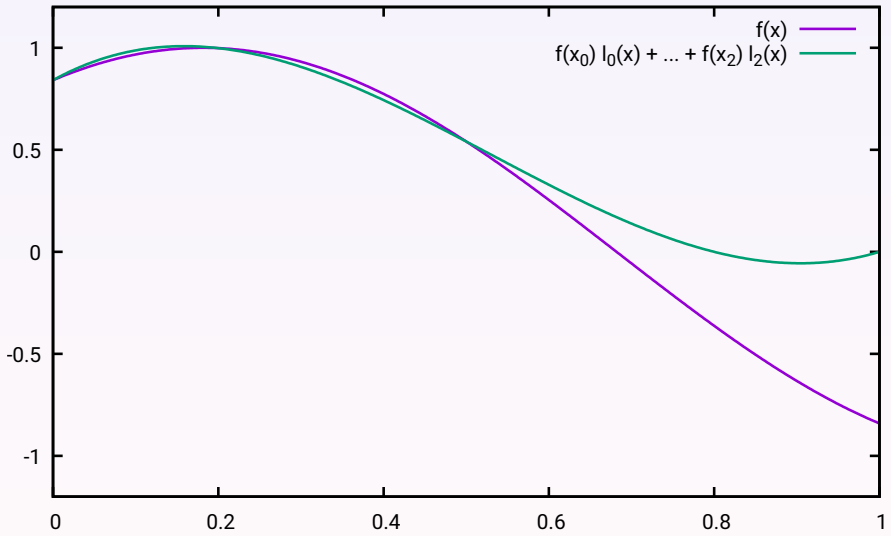
Lagrange representation



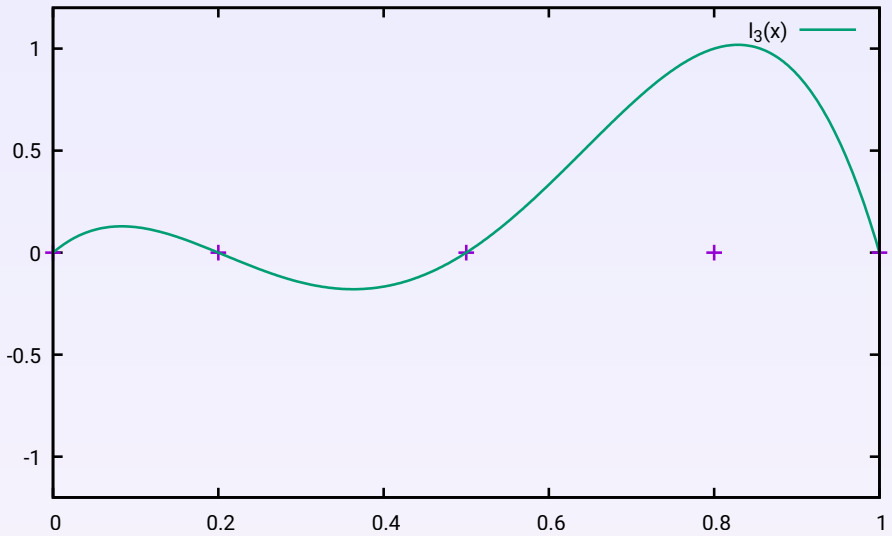


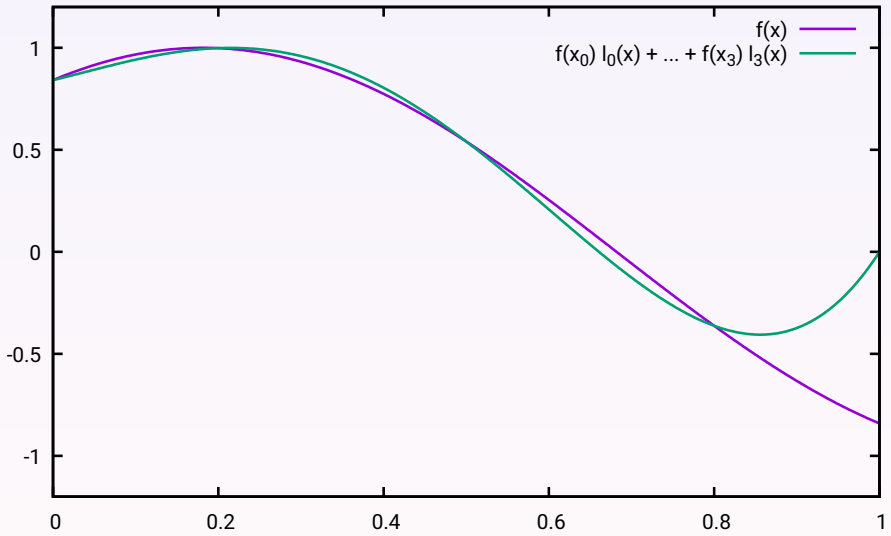
Lagrange representation



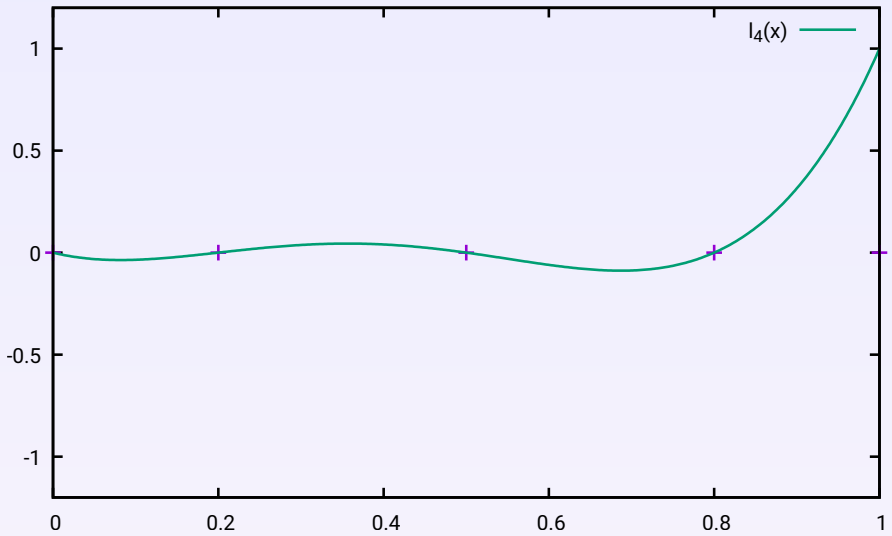


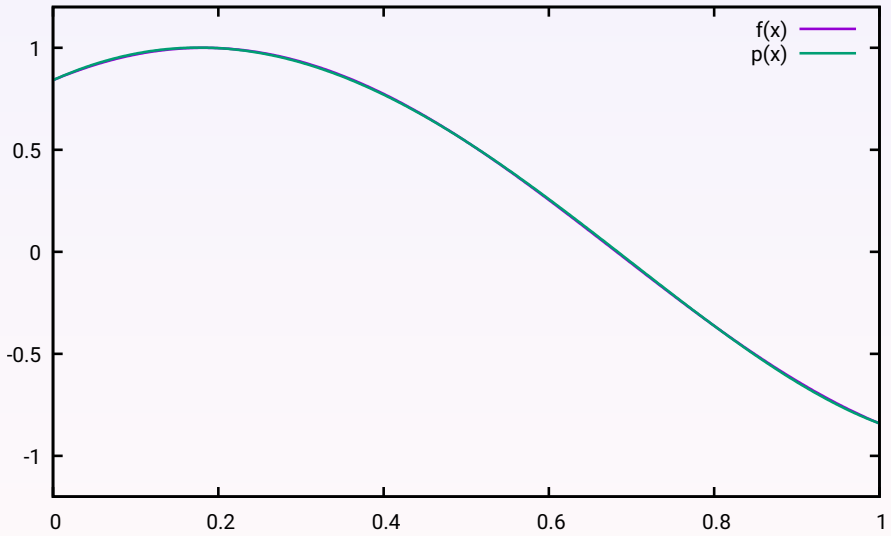
Lagrange representation





Lagrange representation





Numerical differentiation

Goal: Given a function f, evaluate $f'(0)$.

Approach: Approximate f by a polynomial p and evaluate $p'(0)$.

$$f'(0) \approx p'(0) = \sum_{k=0}^{m} f(t_k) \underbrace{\ell_k'(0)}_{=:w_k} = \sum_{k=0}^{m} f(t_k) w_k.$$

Weights $w_0, \ldots, w_m$ can be computed by solving a linear system.
Monomials $p_i(t) = t^i$, $i \in [0 : m]$, have to be differentiated exactly.

$$\sum_{k=0}^{m} t_k^i w_k = \sum_{k=0}^{m} p_i(t_k) w_k = p_i'(0) = \begin{cases} 1 & \text{if } i = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Example: Fourth-order numerical differentiation

Approach: Interpolation points $-3, -1, 1, 3$, weights have to solve

$$\begin{aligned}
w_0 + w_1 + w_2 + w_3 &= 0, && (p_0(t) = 1)\\
-3w_0 - w_1 + w_2 + 3w_3 &= 1, && (p_1(t) = t)\\
9w_0 + w_1 + w_2 + 9w_3 &= 0, && (p_2(t) = t^2)\\
-27w_0 - w_1 + w_2 + 27w_3 &= 0. && (p_3(t) = t^3)
\end{aligned}$$

Example: Fourth-order numerical differentiation

Approach: Interpolation points $-3, -1, 1, 3$, weights have to solve

$$\begin{aligned} w_0 + w_1 + w_2 + w_3 &= 0, & (p_0(t) &= 1) \\ -3w_0 - w_1 + w_2 + 3w_3 &= 1, & (p_1(t) &= t) \\ 9w_0 + w_1 + w_2 + 9w_3 &= 0, & (p_2(t) &= t^2) \\ -27w_0 - w_1 + w_2 + 27w_3 &= 0. & (p_3(t) &= t^3) \end{aligned}$$

The first and third equation yield $w_3 = -w_0$ and $w_2 = -w_1$.
The fourth equation yields $w_1 = -27w_0$, and the second $w_0 = \frac{1}{48}$.

Example: Fourth-order numerical differentiation

Approach: Interpolation points $-3, -1, 1, 3$, weights have to solve

$$\begin{aligned} w_0 + w_1 + w_2 + w_3 &= 0, & (p_0(t) &= 1) \\ -3w_0 - w_1 + w_2 + 3w_3 &= 1, & (p_1(t) &= t) \\ 9w_0 + w_1 + w_2 + 9w_3 &= 0, & (p_2(t) &= t^2) \\ -27w_0 - w_1 + w_2 + 27w_3 &= 0. & (p_3(t) &= t^3) \end{aligned}$$

The first and third equation yield $w_3 = -w_0$ and $w_2 = -w_1$.
The fourth equation yields $w_1 = -27w_0$, and the second $w_0 = \frac{1}{48}$.

Result: Another central difference quotient.

$$f'(0) \approx \frac{f(-3) - 27\,f(-1) + 27\,f(1) - f(3)}{48}.$$

Experiment: Numerical differentiation

Idea: Control accuracy by scaling, $\hat{f}(t) = f(ht)$,

$$f'(0) = \frac{\hat{f}'(0)}{h} = \frac{f(-3h) - 27\,f(-h) + 27\,f(h) - f(3h)}{48h}.$$

Experiment: Numerical differentiation

Idea: Control accuracy by scaling, $\hat{f}(t) = f(ht)$,

$$f'(0) = \frac{\hat{f}'(0)}{h} = \frac{f(-3h) - 27\,f(-h) + 27\,f(h) - f(3h)}{48h}.$$

Experiment: Approximate derivative of $f(t) = e^t$.

h	error	factor
1	-9.5_{-2}	
1/2	-5.0_{-3}	19.2
1/4	-3.0_{-4}	16.7
1/8	-1.8_{-5}	16.2
1/16	-1.1_{-6}	16.0
1/32	-7.2_{-8}	16.0
1/64	-4.5_{-9}	16.0

Example: Second derivative

Approach: Interpolation points $-1, 0, 1$, weights have to solve

$$\begin{aligned} w_0 + w_1 + w_2 &= 0, & (p_0(t) &= 1) \\ -w_0 + w_2 &= 0, & (p_1(t) &= t) \\ w_0 + w_2 &= 2. & (p_2(t) &= t^2) \end{aligned}$$

Example: Second derivative

Approach: Interpolation points $-1, 0, 1$, weights have to solve

$$\begin{aligned} w_0 + w_1 + w_2 &= 0, & (p_0(t) &= 1) \\ -w_0 + w_2 &= 0, & (p_1(t) &= t) \\ w_0 + w_2 &= 2. & (p_2(t) &= t^2) \end{aligned}$$

The second equation yields $w_2 = w_0$.
The third yields $w_0 = 1$, and the first $w_1 = -2$.

Example: Second derivative

Approach: Interpolation points $-1, 0, 1$, weights have to solve

$$\begin{aligned} w_0 + w_1 + w_2 &= 0, & (p_0(t) &= 1) \\ -w_0 + w_2 &= 0, & (p_1(t) &= t) \\ w_0 + w_2 &= 2. & (p_2(t) &= t^2) \end{aligned}$$

The second equation yields $w_2 = w_0$.
The third yields $w_0 = 1$, and the first $w_1 = -2$.

Result: Central difference quotient for the second derivative.

$$f''(0) \approx f(-1) - 2\,f(0) + f(1).$$

Experiment: Second derivative

Once again: Control accuracy by scaling, $\hat{f}(t) = f(ht)$,

$$f''(0) = \frac{\hat{f}''(0)}{h^2} = \frac{f(-h) - 2\,f(0) + f(h)}{h^2}.$$

Experiment: Second derivative

Once again: Control accuracy by scaling, $\hat{f}(t) = f(ht)$,

$$f''(0) = \frac{\hat{f}''(0)}{h^2} = \frac{f(-h) - 2\,f(0) + f(h)}{h^2}.$$

Experiment: Approximate second derivative of $f(t) = e^t$.

h	error	factor
1	8.6_{-2}	
1/2	2.1_{-2}	4.1
1/4	5.2_{-3}	4.0
1/8	1.3_{-3}	4.0
1/16	3.3_{-4}	4.0
1/32	8.1_{-5}	4.0
1/64	2.0_{-5}	4.0

Extrapolation

Goal: Given a function f, evaluate $f(0) = \lim_{t \to 0} f(t)$.

Approach: Approximate f by a polynomial p and evaluate $p(0)$.

$$f(0) \approx p(0) = \sum_{k=0}^{m} f(t_k) \ell_k(0) = \sum_{k=0}^{m} f(t_k) w_k.$$

Weights $w_0, \ldots, w_k$ can again be computed by solving a linear system. Monomials $p_i(t) = t^i$, $i \in [0 : m]$, have to be evaluated exactly.

$$\sum_{k=0}^{m} t_k^i w_k = \sum_{k=0}^{m} p_i(t_k) w_k = p_i(0) = \begin{cases} 1 & \text{if } i = 0, \\ 0 & \text{otherwise.} \end{cases}$$

Example: Fourth-order extrapolation

Approach: Interpolation points $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$, weights have to solve

$$\begin{aligned} w_0 + w_1 + w_2 + w_3 &= 1, & (p_0(t) &= 1) \\ w_0 + \tfrac{1}{2} w_1 + \tfrac{1}{4} w_2 + \tfrac{1}{8} w_3 &= 0, & (p_1(t) &= t) \\ w_0 + \tfrac{1}{4} w_1 + \tfrac{1}{16} w_2 + \tfrac{1}{64} w_3 &= 0, & (p_2(t) &= t^2) \\ w_0 + \tfrac{1}{8} w_1 + \tfrac{1}{64} w_2 + \tfrac{1}{512} w_3 &= 0. & (p_3(t) &= t^3) \end{aligned}$$

The solution is $w_0 = -\frac{1}{21}$, $w_1 = \frac{2}{3}$, $w_2 = -\frac{8}{3}$, $w_3 = \frac{64}{21}$.

Result: Difference quotient for approximating the limit $t \to 0$.

$$\lim_{t \to 0} f(t) \approx \frac{-f(1) + 14\, f(\tfrac{1}{2}) - 56\, f(\tfrac{1}{4}) + 64\, f(\tfrac{1}{8})}{21}$$

Experiment: Extrapolation

Once again: Control accuracy by scaling, $\hat{f}(t) = f(ht)$,

$$\lim_{t \to 0} f(t) = \lim_{t \to 0} \hat{f}(t) \approx \frac{-f(h) + 14\, f(\frac{h}{2}) - 56\, f(\frac{h}{4}) + 64\, f(\frac{h}{8})}{21}.$$

Experiment: Extrapolation

Once again: Control accuracy by scaling, $\hat{f}(t) = f(ht)$,

$$\lim_{t\to 0} f(t) = \lim_{t\to 0} \hat{f}(t) \approx \frac{-f(h) + 14\, f(\frac{h}{2}) - 56\, f(\frac{h}{4}) + 64\, f(\frac{h}{8})}{21}.$$

Experiment: Approximate $\lim_{t\to 0} \frac{\exp(t)-1}{t}$.

h	error	factor
1	-1.8_{-4}	
1/2	-9.5_{-6}	18.8
1/4	-5.5_{-7}	17.3
1/8	-3.3_{-8}	16.6
1/16	-2.0_{-9}	16.3
1/32	-1.3_{-10}	16.1
1/64	-7.7_{-12}	16.2

Numerical integration

Goal: Given a function f, evaluate $\int_{-1}^{1} f(t)\,dt$.

Approach: Approximate f by a polynomial p and integrate p.

$$\int_{-1}^{1} f(t)\,dt \approx \int_{-1}^{1} p(t)\,dt = \sum_{k=0}^{m} f(t_k) \int_{-1}^{1} \ell_k(t)\,dt = \sum_{k=0}^{m} f(t_k) w_k.$$

Weights $w_0, \ldots, w_k$ can again be computed by solving a linear system. Monomials $p_i(t) = t^i$, $i \in [0 : m]$, have to be integrated exactly.

$$\sum_{k=0}^{m} t_k^i w_k = \sum_{k=0}^{m} p_i(t_k) w_k = \int_{-1}^{1} p_i(t)\,dt = \begin{cases} \frac{2}{i+1} & \text{if } i \text{ even,} \\ 0 & \text{otherwise.} \end{cases}$$

Examples: Trapezoidal and Simpson's rule

Trapezoidal quadrature rule: Interpolation points $-1, 1$, weights solve

$$\begin{aligned} w_0 + w_1 &= 2, && (p_0(t) = 1) \\ -w_0 + w_1 &= 0. && (p_1(t) = t) \end{aligned}$$

The solution is $w_0 = w_1 = 1$.

Examples: Trapezoidal and Simpson's rule

Trapezoidal quadrature rule: Interpolation points $-1, 1$, weights solve

$$\begin{aligned} w_0 + w_1 &= 2, & (p_0(t) &= 1) \\ -w_0 + w_1 &= 0. & (p_1(t) &= t) \end{aligned}$$

The solution is $w_0 = w_1 = 1$.

Simpson's quadrature rule: Interpolation points $-1, 0, 1$, weights solve

$$\begin{aligned} w_0 + w_1 + w_2 &= 2, & (p_0(t) &= 1) \\ -w_0 + w_2 &= 0, & (p_1(t) &= t) \\ w_0 + w_2 &= \tfrac{2}{3}. & (p_2(t) &= t^2) \end{aligned}$$

The solution is $w_0 = w_2 = \frac{1}{3}$, $w_1 = \frac{4}{3}$.

Composite integration

Problem: Scaling to $\hat{f}(t) = f(ht)$ only yields

$$\int_{-h}^{h} f(t)\,dt = h \int_{-1}^{1} \hat{f}(t)\,dt \approx h \sum_{k=0}^{m} f(ht_k) w_k,$$

i.e., the interval is scaled, as well.

Solution: Split the original interval $[a, b]$ into subintervals, apply quadrature rule to subintervals.

Experiment: Composite trapezoidal rule

Scaling and subdividing yields

$$\int_a^b f(t)\,dt \approx \frac{b-a}{2n}\left(f(a) + 2\sum_{i=1}^{n-1} f\left(a + \tfrac{b-a}{n} i\right) + f(b)\right).$$

Experiment: Composite trapezoidal rule

Scaling and subdividing yields

$$\int_a^b f(t)\,dt \approx \frac{b-a}{2n}\left(f(a)+2\sum_{i=1}^{n-1} f\left(a+\tfrac{b-a}{n}i\right)+f(b)\right).$$

Experiment: Approximate $\int_0^2 e^t\,dt$.

n	error	factor
1	2	
2	5.2_{-1}	3.8
4	1.3_{-1}	4.0
8	3.3_{-2}	3.9
16	8.3_{-3}	4.0
32	2.1_{-3}	3.9
64	5.2_{-4}	4.0

Romberg quadrature

Idea: Combine extrapolation and the trapezoidal quadrature rule.

Trapezoidal rule for $h > 0$ with $n := \frac{b-a}{h} \in \mathbb{N}$ given by

$$T(h) = \frac{h}{2}\left(f(a) + 2\sum_{i=1}^{n-1} f(a+hi) + f(b)\right).$$

Euler-Maclaurin summation yields $c_1, c_2, \ldots$ such that

$$T(h) = \int_a^b f(t)\,dt + c_1 h^2 + c_2 h^4 + \ldots$$

Romberg quadrature

Idea: Combine extrapolation and the trapezoidal quadrature rule.

Trapezoidal rule for $h > 0$ with $n := \frac{b-a}{h} \in \mathbb{N}$ given by

$$T(h) = \frac{h}{2}\left(f(a) + 2\sum_{i=1}^{n-1} f(a+hi) + f(b)\right).$$

Euler-Maclaurin summation yields $c_1, c_2, \ldots$ such that

$$T(h) = \int_a^b f(t)\,dt + c_1 h^2 + c_2 h^4 + \ldots$$

Idea: Use extrapolation to find limit $s \to 0$ of

$$g(s) := T(\sqrt{s}) = \int_a^b f(t)\,dt + c_1 s + c_2 s^2 + \ldots$$

Extrapolation

Approach: Approximate limit $s \to 0$ of $g(s) = T(\sqrt{s})$ using the interpolation points $s_0 := \frac{(b-a)^2}{4}$, $s_k := s_0 4^{-k}$, $k \in \mathbb{N}$.

$$g(s_k) = T(\sqrt{s_k}) = T\left(\tfrac{b-a}{2^k}\right) = \frac{h_k}{2}\left(f(a) + 2\sum_{i=1}^{2^{k+1}-1} f(a + h_k i) + f(b)\right)$$

with $h_0 = \frac{b-a}{2}$, $h_k = h_0 2^{-k}$.

Implementation: We can use $g(s_k)$ to compute $g(s_{k+1})$ more efficiently.

Observation: We do not need the entire interpolating polynomial p, we only need its value in $s = 0$.

Neville-Aitken method

Goal: Evaluate an interpolating polynomial efficiently in 0.

Idea: Define polynomials $p_{i,j}$, $j \geq i$, of degree $j - i$ such that

$$p_{i,j}(s_k) = g(s_k) \qquad \text{for all } k \in [i : j].$$

Neville-Aitken method

Goal: Evaluate an interpolating polynomial efficiently in 0.

Idea: Define polynomials $p_{i,j}$, $j \geq i$, of degree $j - i$ such that

$$p_{i,j}(s_k) = g(s_k) \qquad \text{for all } k \in [i : j].$$

Aitken recurrence:

$$p_{i,j}(s) = \begin{cases} g(s_i) & \text{if } i = j, \\ \frac{s_i - s}{s_i - s_j} p_{i+1,j}(s) + \frac{s - s_j}{s_i - s_j} p_{i,j-1}(s) & \text{otherwise.} \end{cases}$$

Neville-Aitken method

Goal: Evaluate an interpolating polynomial efficiently in 0.

Idea: Define polynomials $p_{i,j}$, $j \geq i$, of degree $j - i$ such that

$$p_{i,j}(s_k) = g(s_k) \qquad \text{for all } k \in [i : j].$$

Aitken recurrence:

$$p_{i,j}(s) = \begin{cases} g(s_i) & \text{if } i = j, \\ \frac{s_i - s}{s_i - s_j} p_{i+1,j}(s) + \frac{s - s_j}{s_i - s_j} p_{i,j-1}(s) & \text{otherwise.} \end{cases}$$

In our case: $s_k = s_0\, 4^{-k}$ and therefore

$$\frac{s_i}{s_i - s_j} = \frac{4^{-i}}{4^{-i} - 4^{-j}} = \frac{4^{j-i}}{4^{j-i} - 1}, \qquad \frac{s_j}{s_i - s_j} = \frac{4^{-j}}{4^{-i} - 4^{-j}} = \frac{1}{4^{j-i} - 1}.$$

Romberg algorithm

Approach: In each step, compute $g(s_k)$ and update $p_{kk}(0), \ldots, p_{0k}(0)$.

$$g(s_0) = p_{0,0}(0)$$

Romberg algorithm

Approach: In each step, compute $g(s_k)$ and update $p_{kk}(0), \ldots, p_{0k}(0)$.

$$g(s_0) = p_{0,0}(0)$$
$$g(s_1) = p_{1,1}(0)$$

Romberg algorithm

Approach: In each step, compute $g(s_k)$ and update $p_{kk}(0), \ldots, p_{0k}(0)$.

$$g(s_0) = p_{0,0}(0)$$
$$g(s_1) = p_{1,1}(0) \quad p_{0,1}(0)$$

Romberg algorithm

Approach: In each step, compute $g(s_k)$ and update $p_{kk}(0), \ldots, p_{0k}(0)$.

$$
\begin{array}{lll}
g(s_0) = p_{0,0}(0) & & \\
g(s_1) = p_{1,1}(0) & p_{0,1}(0) & \\
g(s_2) = p_{2,2}(0) & &
\end{array}
$$

Romberg algorithm

Approach: In each step, compute $g(s_k)$ and update $p_{kk}(0), \ldots, p_{0k}(0)$.

$$
\begin{array}{lll}
g(s_0) = p_{0,0}(0) & & \\
g(s_1) = p_{1,1}(0) & p_{0,1}(0) & \\
g(s_2) = p_{2,2}(0) & p_{1,2}(0) &
\end{array}
$$

Romberg algorithm

Approach: In each step, compute $g(s_k)$ and update $p_{kk}(0), \ldots, p_{0k}(0)$.

$$
\begin{aligned}
g(s_0) &= p_{0,0}(0) \\
g(s_1) &= p_{1,1}(0) \quad p_{0,1}(0) \\
g(s_2) &= p_{2,2}(0) \quad p_{1,2}(0) \quad p_{0,2}(0)
\end{aligned}
$$

Romberg algorithm

Approach: In each step, compute $g(s_k)$ and update $p_{kk}(0), \ldots, p_{0k}(0)$.

$$
\begin{aligned}
g(s_0) &= p_{0,0}(0) \\
g(s_1) &= p_{1,1}(0) \quad p_{0,1}(0) \\
g(s_2) &= p_{2,2}(0) \quad p_{1,2}(0) \quad p_{0,2}(0) \\
g(s_3) &= p_{3,3}(0)
\end{aligned}
$$

Romberg algorithm

Approach: In each step, compute $g(s_k)$ and update $p_{kk}(0), \ldots, p_{0k}(0)$.

$$
\begin{array}{llll}
g(s_0) = p_{0,0}(0) & & & \\
g(s_1) = p_{1,1}(0) & p_{0,1}(0) & & \\
g(s_2) = p_{2,2}(0) & p_{1,2}(0) & p_{0,2}(0) & \\
g(s_3) = p_{3,3}(0) & p_{2,3}(0) & &
\end{array}
$$

Romberg algorithm

Approach: In each step, compute $g(s_k)$ and update $p_{kk}(0), \ldots, p_{0k}(0)$.

$$
\begin{array}{lllll}
g(s_0) = p_{0,0}(0) & & & \\
g(s_1) = p_{1,1}(0) & p_{0,1}(0) & & \\
g(s_2) = p_{2,2}(0) & p_{1,2}(0) & p_{0,2}(0) & \\
g(s_3) = p_{3,3}(0) & p_{2,3}(0) & p_{1,3}(0) &
\end{array}
$$

Romberg algorithm

Approach: In each step, compute $g(s_k)$ and update $p_{kk}(0), \ldots, p_{0k}(0)$.

$$
\begin{array}{llll}
g(s_0) = p_{0,0}(0) & & & \\
g(s_1) = p_{1,1}(0) & p_{0,1}(0) & & \\
g(s_2) = p_{2,2}(0) & p_{1,2}(0) & p_{0,2}(0) & \\
g(s_3) = p_{3,3}(0) & p_{2,3}(0) & p_{1,3}(0) & p_{0,3}(0)
\end{array}
$$

Romberg algorithm

Approach: In each step, compute $g(s_k)$ and update $p_{kk}(0), \ldots, p_{0k}(0)$.

$$
\begin{array}{llll}
g(s_0) = p_{0,0}(0) & & & \\
g(s_1) = p_{1,1}(0) & p_{0,1}(0) & & \\
g(s_2) = p_{2,2}(0) & p_{1,2}(0) & p_{0,2}(0) & \\
g(s_3) = p_{3,3}(0) & p_{2,3}(0) & p_{1,3}(0) & p_{0,3}(0)
\end{array}
$$

Result The polynomials $p_{0,0}$, $p_{0,1}$, $p_{0,2}$, $p_{0,3}$ are the required interpolating polynomials, and there values in $s = 0$ approximate the integral.

Experiment: Romberg quadrature

Experiment: Approximate $\int_0^2 e^t \, dt$.

	Trapezoidal		Romberg	
n	error	factor	error	factor
2	5.2_{-1}		5.2_{-1}	
4	1.3_{-1}	4.0	2.2_{-3}	23.6
8	3.3_{-2}	3.9	3.2_{-6}	687.5
16	8.3_{-3}	4.0	1.2_{-9}	2666.7
32	2.1_{-3}	3.9	1.2_{-13}	10000.0
64	5.2_{-4}	4.0	"0"	

Observation: Extrapolation significantly improves the accuracy in this case.

Summary

Higher-order methods are based on polynomials of higher degree.

Interpolation constructs these polynomials based only on point values.

Numerical differentiation: Use values of f to approximate its derivative.

Extrapolation: Use values of f to approximate its limit.

Numerical integration: Use values of f to approximate its integral.

Combining differentiation or integration with extrapolation can yield very high accuracies, e.g., in the Romberg quadrature method.