Applied Math

Interpolation

Federico Pichi

11 November 2025

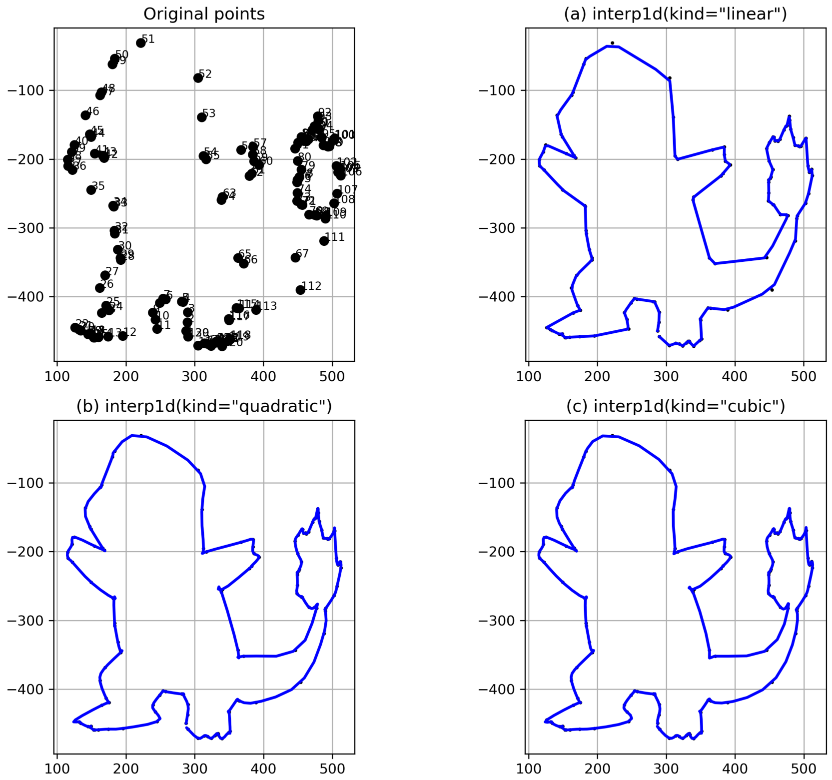
Outline

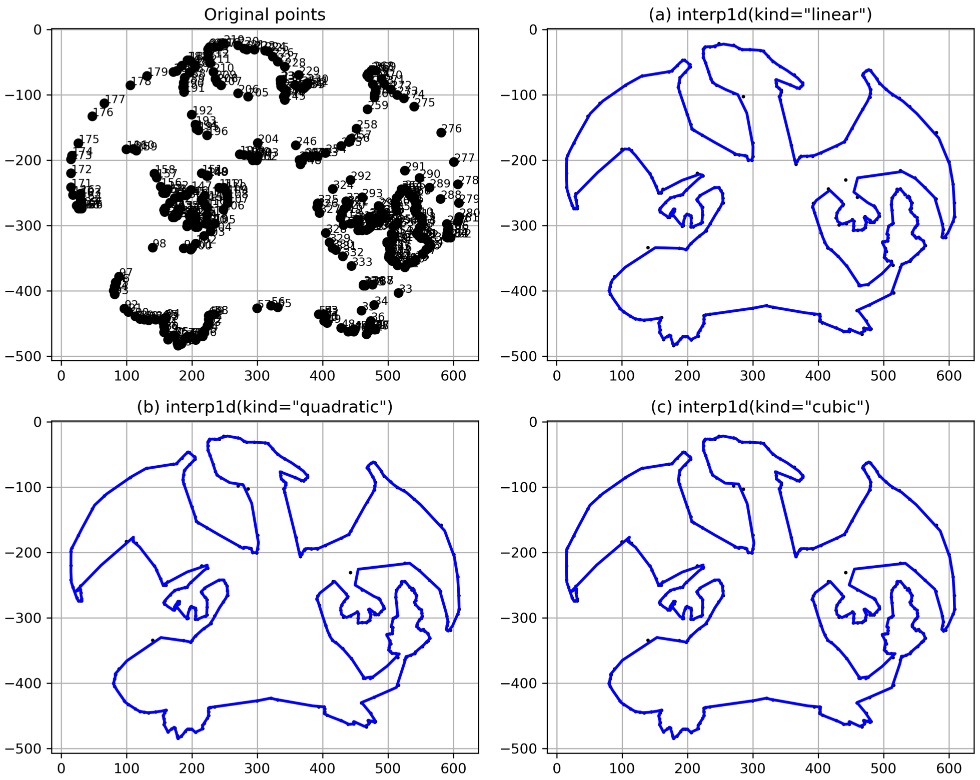
- Introduction
- Interpolation problem
- Polynomial interpolants and uniqueness
  - Monomial
  - Lagrange
  - Newton
- Interpolation error
- Convergence
- Runge phenomenon
- Stability and Lebesgue constant
- Chebyshev-Gauss-Lobatto nodes

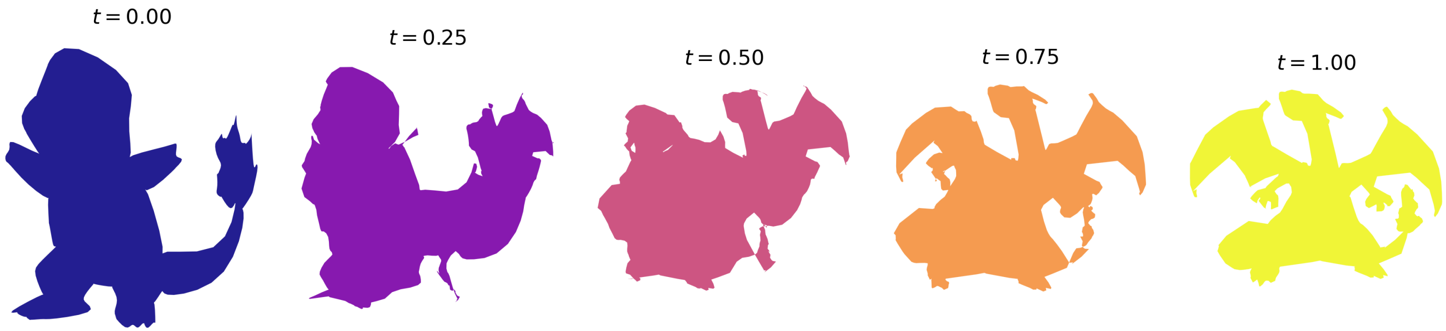
Motivations

- **Data Reconstruction**: estimate missing or unknown values between known data points.
  - Experimental or measured data are available only at discrete points
- **Function Approximation**: approximate complex or unknown functions with simpler ones
  - Polynomials enables easier computation, differentiation, or integration.
- **Computational Efficiency**: replace expensive analytical or numerical evaluations
  - Exploiting fast approximate evaluations using an interpolant in simulations.
- **Data Smoothing and Visualization**: generate smooth curves or surfaces through discrete datasets
  - Qualitative analysis and graphical representation.
- **Modeling and Simulation**: provide continuous input data to numerical solvers for PDEs and ODEs
  - Represent the solution, interpolate boundary or initial conditions.

Real motivations

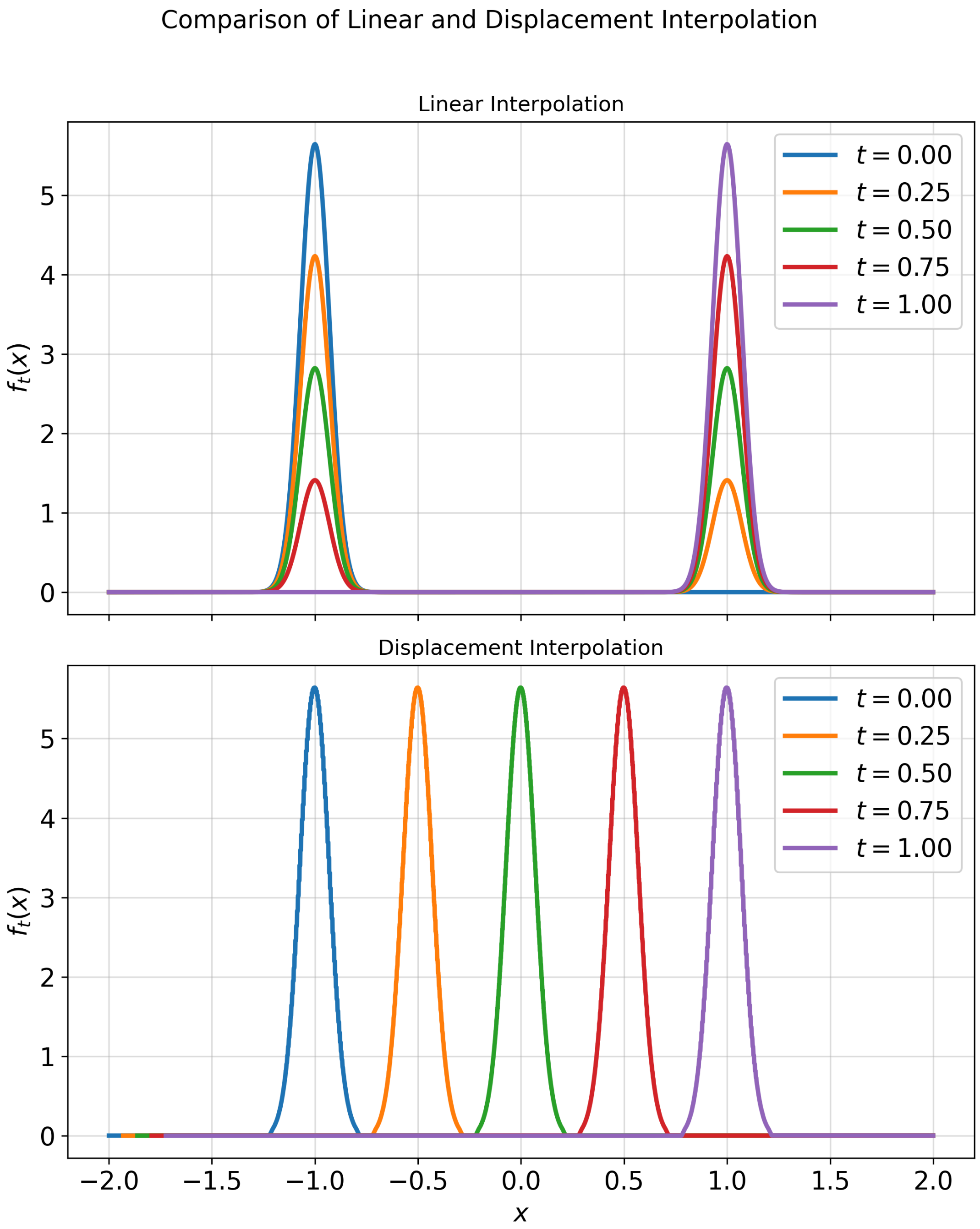






Methodologies and Challenges

- Univariate and multivariate functions
- Polynomial interpolation (Monomial, Lagrange, Newton, and Hermite)
- Piecewise interpolation (splines)
- Trigonometric interpolation (periodicity)
- Scattered data interpolation (Radial Basis Functions for irregularly spaced data)
- High-dimensional and sparse interpolation (sparse grids and tensor-product)
- Interpolating moving features with different support (displacement interpolation)



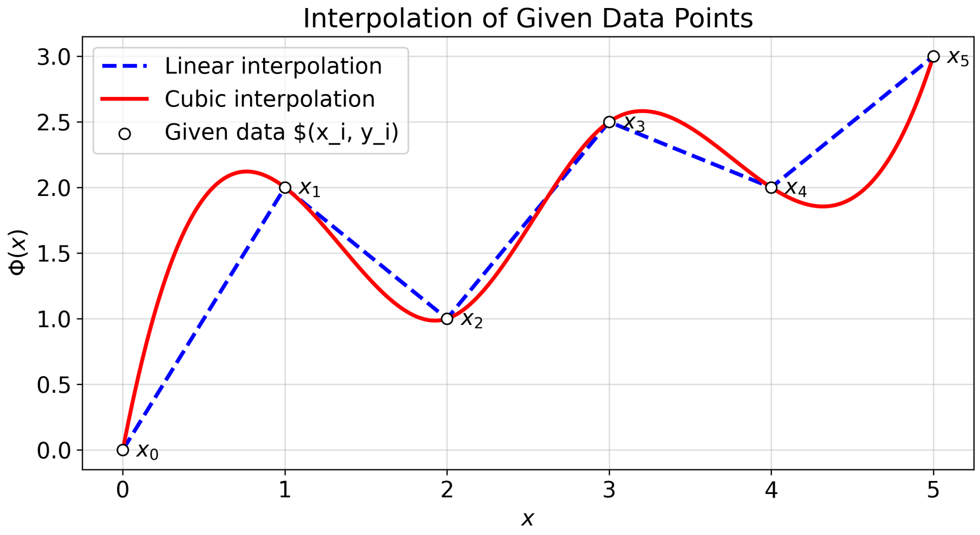
Problem statement

Interpolation. Given $n+1$ pairs (x_i, y_i), we look for a function $\Phi = \Phi(x)$ such that

$$\Phi(x_i) = y_i, \quad \text{for } i = 0, \ldots, n$$

where y_i are some given *values* representing the exact value for a function f at x_i or experimental data. Φ is the *interpolant* that approximate such function/distribution interpolating the y_i at the nodes x_i.

- *Polynomial Interpolation*: Φ is algebraic or piecewise polynomial.
- Additional data might be prescribed (slope), and constraints might be imposed (smooth, monotone)



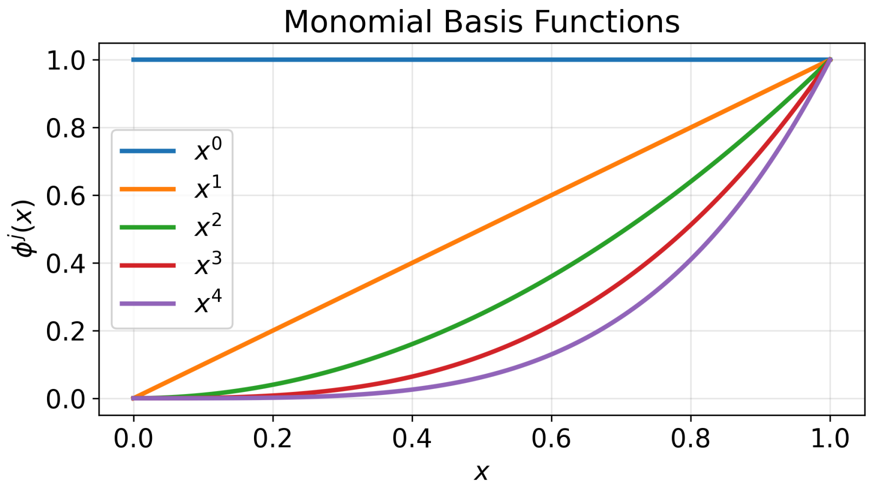
Interpolants

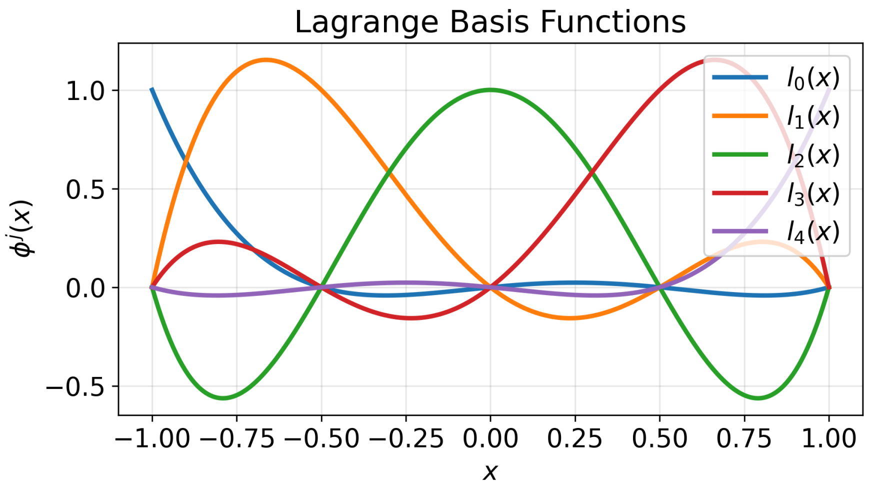
- **Monomial** interpolation

$$\Phi(x) = \sum_{j=0}^{m} b_j x^j$$

- **Lagrange** polynomial interpolation

$$\Phi(x) = \sum_{j=0}^{m} y_j l_j(x), \quad \text{where} \quad l_j(x) = \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}, \quad \text{for } j = 0, \ldots, n$$





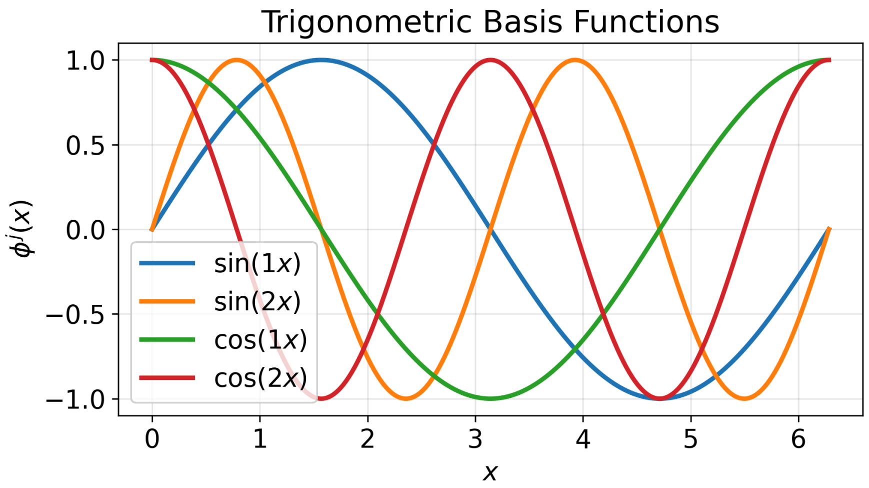
Interpolants

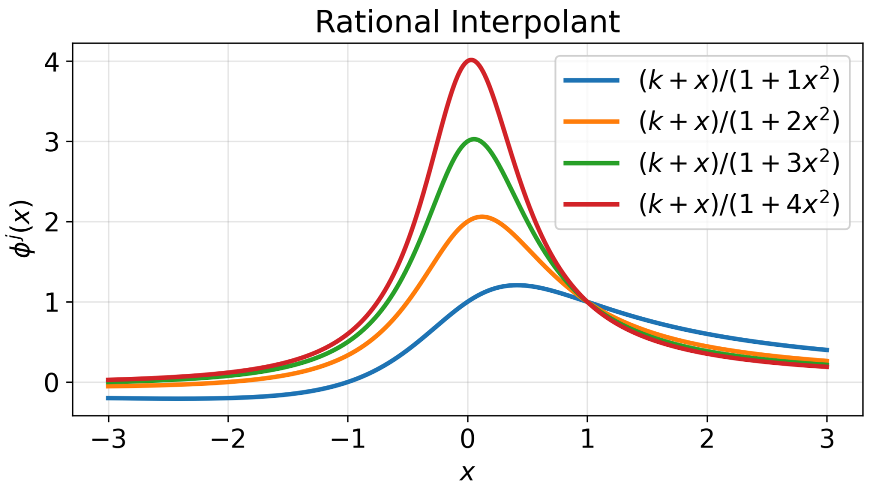
- **Trigonometric** interpolation (M is an integer equal to $m/2$ if m is even, $(m+1)/2$ if m is odd)

$$\Phi(x) = \sum_{j=-M}^{M} b_j e^{ijx}, \quad \text{with} \quad e^{ijx} = \cos(jx) + i\sin(jx)$$

- **Rational** interpolation

$$\Phi(x) = \frac{\sum_{j=0}^{k} b_j x^j}{\sum_{j=0}^{m} b_{k+1+j} x^j}$$





Polynomial interpolation

Interpolating data points $\{(x_i, y_i)\}_{i=0}^{n}$ via a linear combination of nonlinear **basis functions** $\{\phi^j(x)\}_{j=0}^{m}$:

$$\Pi_m(x) = \sum_{j=0}^{m} b_j \phi^j(x)$$

- x_i are called *interpolation nodes*, and when $n \neq m$ the problem is over or under-determined
  - $m > n$, interpolant usually doesn't exist
  - $m < n$, interpolant is not unique
  - $m = n$, data can be fit exactly

Theorem 1 [Existence and uniqueness]. Given $n+1$ distinct nodes $\{x_i\}_{i=0}^{n}$ and $n+1$ corresponding values $\{y_i\}_{i=0}^{n}$, there exists a unique polynomial $\Pi_n \in \mathbb{P}_n$ (of degree $m = n$) such that

$$\Pi_n(x_i) = y_i \quad \text{for } i = 0, \ldots, n.$$

Polynomial interpolation

Proof. Existence by construction. Denoting by $\{l_j\}_{j=0}^n$ a basis for $\mathbb{P}_n$, then Π_n admits the representation

$$\Pi_n(x) = \sum_{j=0}^n b_j l_j(x), \quad \text{such that} \quad \Pi_n(x_i) = \sum_{j=0}^n b_j l_j(x_i) = y_i, \quad \text{for } i = 0, \ldots, n.$$

Lagrange polynomials form a basis for $\mathbb{P}_n$, and it holds $l_j(x_i) = \delta_{ij}$, so that $b_j = y_j$.
As a consequence, the interpolating polynomial exists and has the following form

$$\Pi_n(x) = \sum_{j=0}^n y_j l_j(x), \quad \text{where} \quad l_j(x) = \prod_{\substack{k=0 \\ k \neq j}}^n \frac{x - x_k}{x_j - x_k}, \quad \text{for } j = 0, \ldots, n.$$

To prove uniqueness, suppose that another interpolating polynomial Ψ_m of degree $m \leq n$ exists, such that $\Psi_m(x_i) = y_i$ for $i = 0, \ldots, n$. Then, the difference polynomial $D = \Pi_n - \Psi_m \in \mathbb{P}_n$ vanishes at $n + 1$ distinct points x_i and thus coincides with the null polynomial, i.e. $D \equiv 0$. Therefore, $\Psi_m = \Pi_n$.

Find interpolation coefficients

Given data points $\{(x_i, y_i)\}_{i=0}^{n}$ and an interpolant of the type

$$\Pi_n(x) = \sum_{j=0}^{n} b_j \phi^j(x),$$

exploiting the **interpolation condition** $\Pi_n(x_i) = y_i$ for $i = 0, \ldots, n$ one obtains the **linear system**

$$Ab = \begin{bmatrix} \phi^0(x_0) & \phi^1(x_0) & \cdots & \phi^n(x_0) \\ \phi^0(x_1) & \phi^1(x_1) & \cdots & \phi^n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi^0(x_n) & \phi^1(x_n) & \cdots & \phi^n(x_n) \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} = y$$

Considerations for choosing basis functions

- The system $Ab = y$ has to be solved effectively and efficiently.
- Sensitivity of coefficients b to perturbations in data depends on $\textsf{cond}(A) = \|A^{-1}\|\|A\|$ so that

$$\frac{\|\Delta b\|}{\|b\|} \lesssim \textsf{cond}(A)\left(\frac{\|\Delta y\|}{\|y\|} + \frac{\|\Delta A\|}{\|A\|}\right)$$

- Most common type of interpolation uses polynomials: easy to evaluate, integrate, differentiate.
- Many interpolation problems follow the same procedure: form A and solve for b.

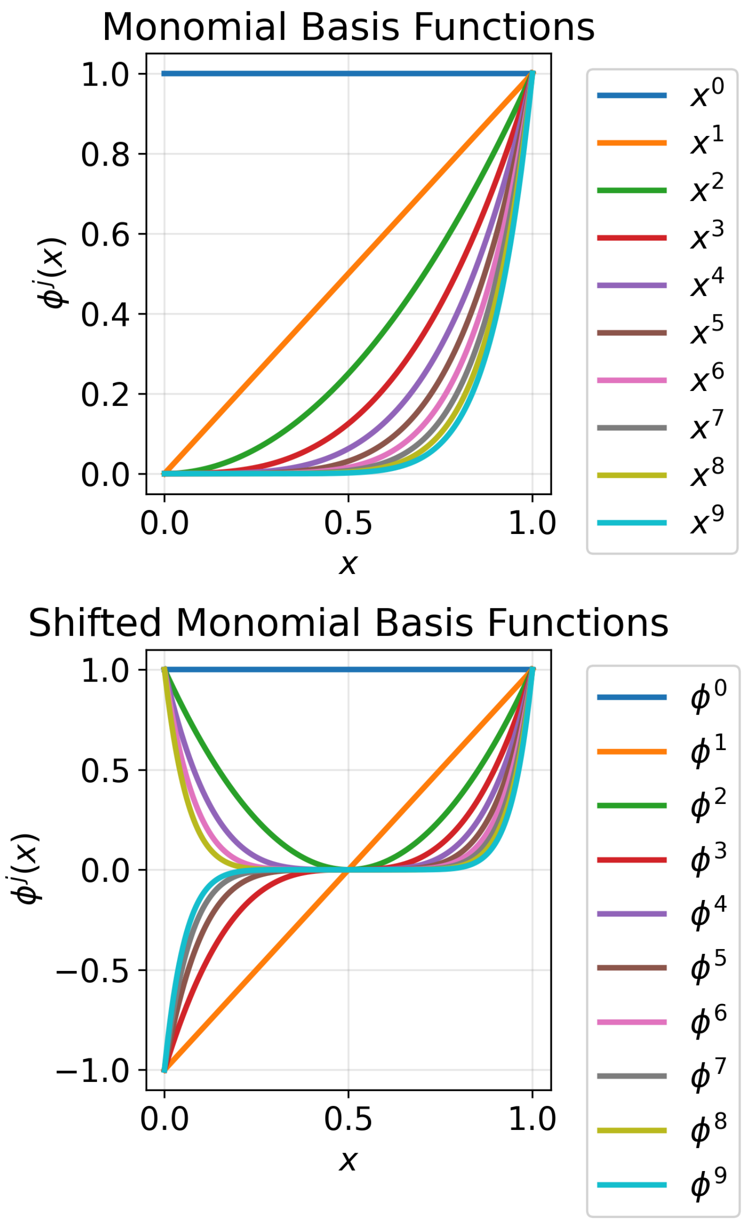
Monomial basis

- Entries of (dense matrix) A are $a_{ij} = \phi^j(x_i) = (x_i)^j$, and $\phi^j(x)$ is similar to $\phi^{j-1}(x)$ as j increases

$$A = \begin{bmatrix} 1 & x_0 & \cdots & (x_0)^n \\ 1 & x_1 & \cdots & (x_1)^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & (x_n)^n \end{bmatrix}$$

- $\mathcal{O}(n^2)$ to construct and $\mathcal{O}(n^3)$ to solve
- $\mathcal{O}(n)$ to evaluate with Horners's rule
- Matrix of this form is called the *Vandermonde* matrix
- Poorly conditioned as n gets larger and/or x-range is wide.
- Improving $\text{cond}(A)$ by scaling and/or shifting basis as

$$\phi^j(x) = \left(\frac{x-c}{d}\right)^j \quad \text{where } c = (x_n + x_0)/2,\ d = (x_n - x_0)/2.$$



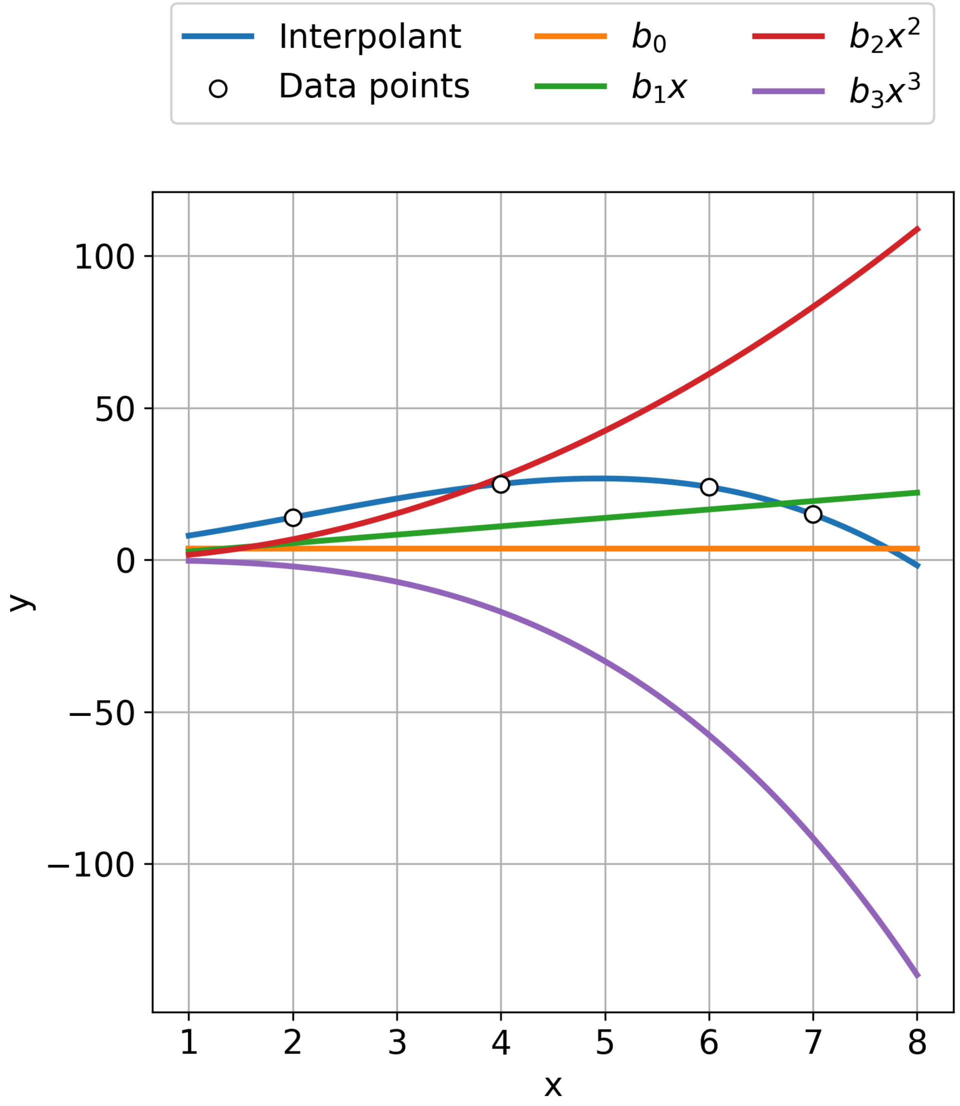
Monomial basis: an example

Construct a monomial basis interpolant for
$\{(x_i, y_i)\}_{i=0}^{3} = \{(2, 14), (6, 24), (4, 25), (7, 15)\}$

- Four basis functions for polynomial degree 3:
  $\{\phi_j(x)\}_{i=0}^{3} = \{1, x, x^2, x^3\}$
- The interpolant is of the form be
  $\Pi_3(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3$
- Construct the linear system:

$$A = \begin{bmatrix} 1 & 2 & 4 & 8 \\ 1 & 6 & 36 & 216 \\ 1 & 4 & 16 & 64 \\ 1 & 7 & 49 & 343 \end{bmatrix}, \quad b = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}, \quad y = \begin{bmatrix} 14 \\ 24 \\ 25 \\ 15 \end{bmatrix}$$

- Solving $Ab = y$ (with **cond**$(A) \approx 10^3$) we get
  $b = [3.8 \quad 2.767 \quad 1.7 \quad -0.267]^T$



Evaluating polynomials

In monomial basis, interpolant can be written as

$$\Pi_n(x) = b_0 + b_1 x + \cdots + b_n x^n,$$

requiring n additions and $2n-1$ multiplications for its evaluation.
By exploiting Horner's nested evaluation scheme one can efficiently rewrite the interpolant as

$$\Pi_n(x) = b_0 + x(b_1 + x(\cdots(b_{n-1} + b_n x)\cdots)),$$

which requires only n additions and n multiplications.

- For example

$$1 - 4x + 5x^2 - 2x^3 + 3x^4 = 1 + x(-4 + x(5 + x(-2 + 3x))).$$

Manipulations with interpolating polynomials (differentiation, integration) are easy with this representation.

Lagrange basis

Lagrange interpolant defined as

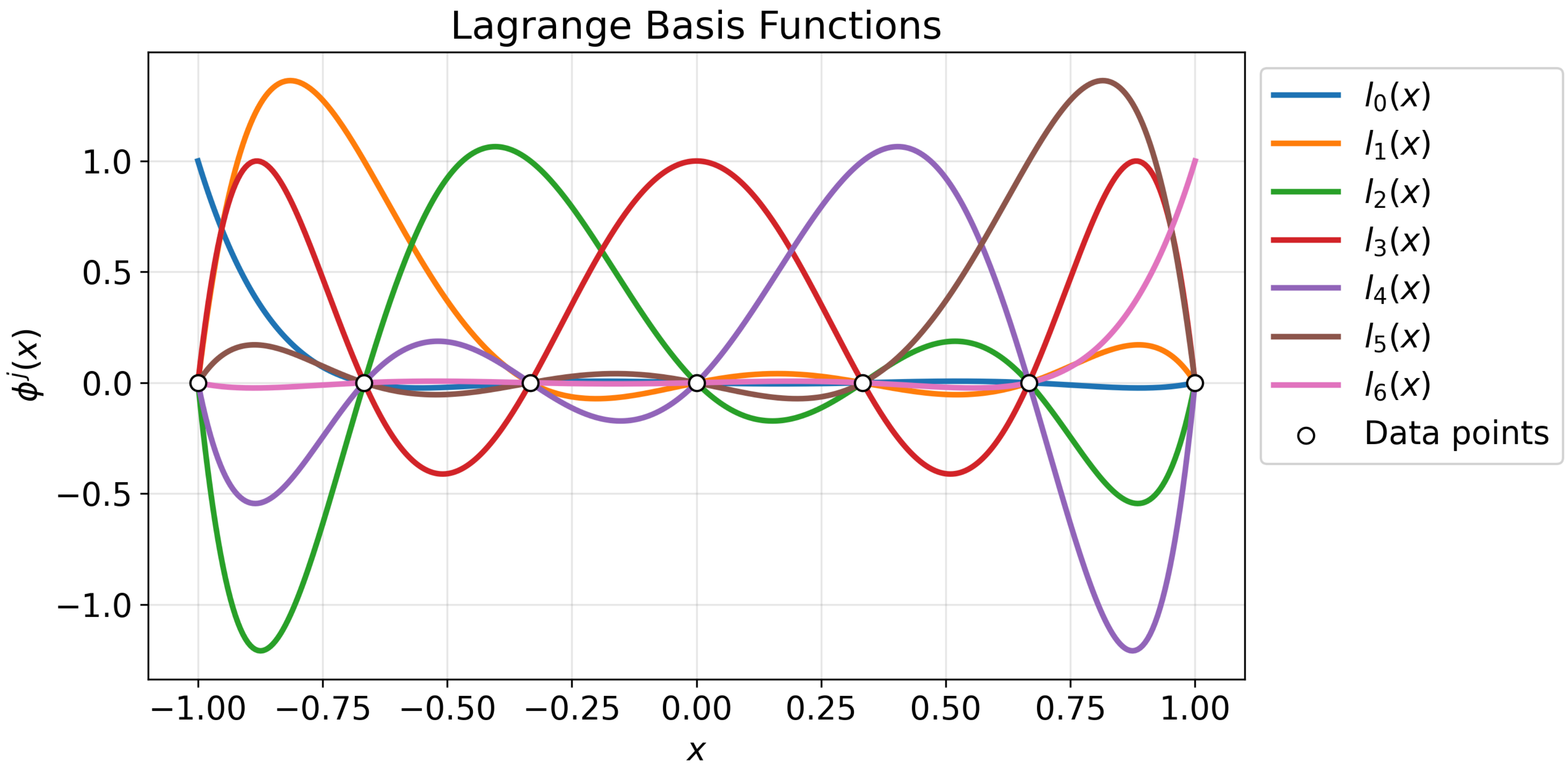
$$\Pi_n(x) = \sum_{j=0}^{n} y_j l_j(x), \quad \text{where} \quad l_j(x) = \prod_{\substack{k=0 \\ k\neq j}}^{n} \frac{x - x_k}{x_j - x_k}, \quad \text{for } j = 0, \ldots, n$$

$\leadsto a_{ij} = l_j(x_i) = \delta_{ij}$, i.e. $A =$ **Id** (best-conditioned matrix), and no system needs to be solved.

- $l_j(x)$ are called **characteristic polynomials** and are defined from interpolation nodes.
- The coefficients b for the Lagrange basis are directly given by the data values y.
- Alternative representation via **nodal polynomial** ω_{n+1} and barycentric weights $q_j^{-1} = \omega'_{n+1}(x_j)$

$$\Pi_n(x) = \omega_{n+1}(x) \sum_{j=0}^{n} \frac{q_j}{(x - x_j)} y_j \quad \text{where} \quad \omega_{n+1}(x) = \prod_{j=0}^{n} (x - x_j), \quad q_j = \prod_{\substack{k=0 \\ k\neq j}}^{n} \frac{1}{x_j - x_k}$$

Lagrange basis



Lagrange basis: an example

Construct a monomial basis interpolant for

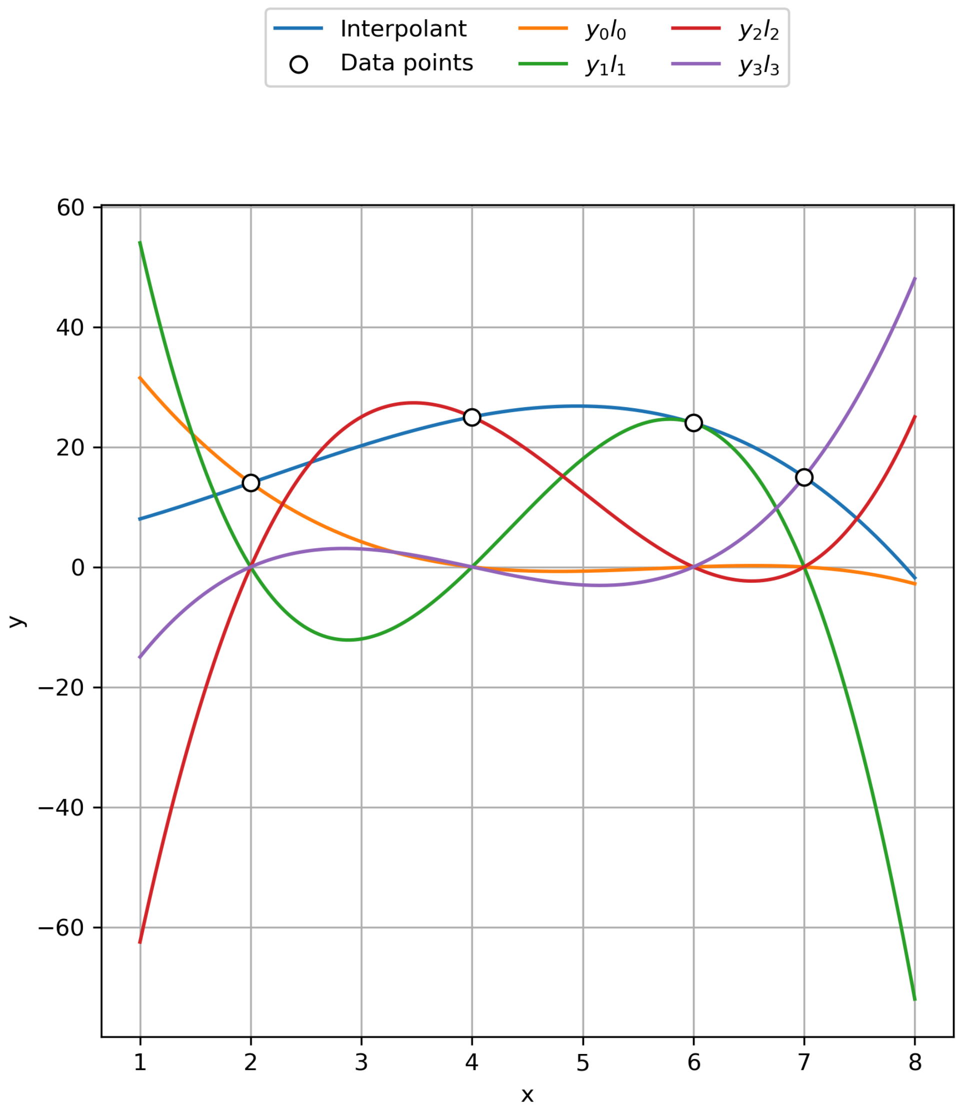
$$\{(x_i, y_i)\}_{i=0}^{3} = \{(2,14),(6,24),(4,25),(7,15)\}$$

- Four basis functions for polynomial degree 3:

$$l_0(x) = \frac{(x-6)(x-4)(x-7)}{(2-6)(2-4)(2-7)}, \quad l_1(x) = \frac{(x-2)(x-4)(x-7)}{(6-2)(6-4)(6-7)}$$
$$l_2(x) = \frac{(x-2)(x-6)(x-7)}{(4-2)(4-6)(4-7)}, \quad l_3(x) = \frac{(x-2)(x-6)(x-4)}{(7-2)(7-6)(7-4)}$$

- The interpolant is of the form be

$$\Pi_3(x) = -\frac{7}{20}(x-6)(x-4)(x-7) - 3(x-2)(x-4)(x-7) + \frac{25}{12}(x-2)(x-6)(x-7) + (x-2)(x-6)(x-4)$$



Newton basis

- Different representation for interpolating polynomials, since Lagrange is not the most convenient one
- Add a new data point without changing the entire interpolant (before we had to recompute q_j)

Goal. Given $n+1$ pairs $\{(x_i, y_i)\}_{i=0}^n$ we want to represent Π_n (with $\Pi_n(x_i) = y_i$ for $i = 0, \ldots, n$) as the sum of Π_{n-1} (with $\Pi_{n-1}(x_i) = y_i$ for $i = 0, \ldots, n-1$) and a polynomial $r_n \in \mathbb{P}_n$ of degree n which depends on the nodes x_i and on only one unknown coefficient:

$$\Pi_n(x) = \Pi_{n-1}(x) + r_n(x)$$

Since $r_n(x_i) = \Pi_n(x_i) - \Pi_{n-1}(x_i) = 0$, for $i = 0, \ldots, n-1$ then the polynomial is given by $\rightsquigarrow r_n(x) = b_n(x - x_0) \cdots (x - x_{n-1}) = b_n \omega_n(x)$ and the coefficients have the closed form

$$b_n \doteq [y_0, \ldots, y_n] = \frac{y_n - \Pi_{n-1}(x_n)}{\omega_n(x_n)}$$

and are called the n-th **Newton divided difference**.

Newton basis

Remarks.

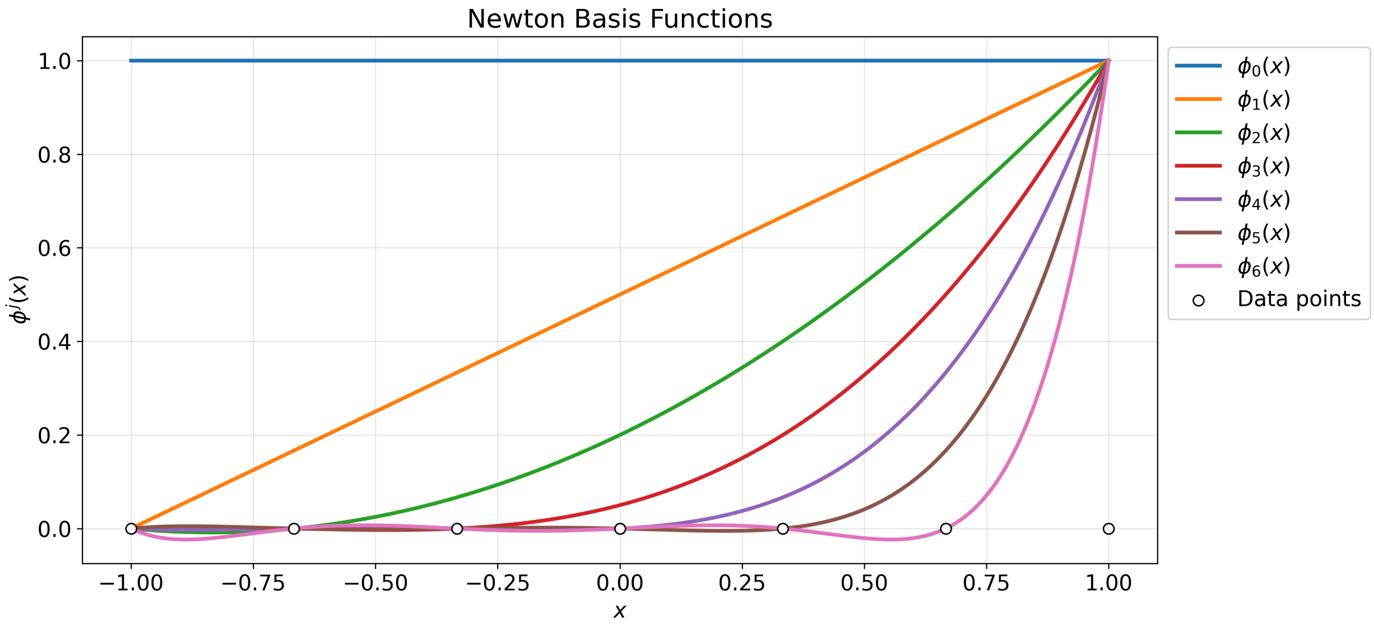
- Uniqueness of the interpolating polynomial ensures that the above expression yields the same interpolating polynomial generated by the Lagrange form.
- Newton basis functions are linearly independent because $\phi^j(x)$ has exactly degree $j-1$.
- New basis function should not interfere with prior interpolation: $\phi^j(x_i) = 0$ for $i < j$.
- Old basis functions don't need info from new data: $\phi^j(x)$ is independent of (x_i, y_i) for $i > j$.

Newton basis

Newton interpolant

$$\Pi_n(x) = \sum_{j=0}^{n} b_j \phi^j(x), \quad \text{where} \quad \phi^j(x) = \prod_{i=0}^{j-1}(x - x_i), \quad \text{for } j \geq 1 \text{ and } \phi^0(x) \equiv 1$$

- This leads to a special matrix A to be inverted: $a_{ij} = \phi^j(x_i) = 0$ for $i < j \rightsquigarrow A$ is **lower triangular**.
- Solution of the triangular system $Ab = y$ is computed via forward-substitution in $\mathcal{O}(n^2)$ operations.



Newton basis: an example

Construct a monomial basis interpolant for

$$\{(x_i, y_i)\}_{i=0}^{3} = \{(2,14),(6,24),(4,25),(7,15)\}$$

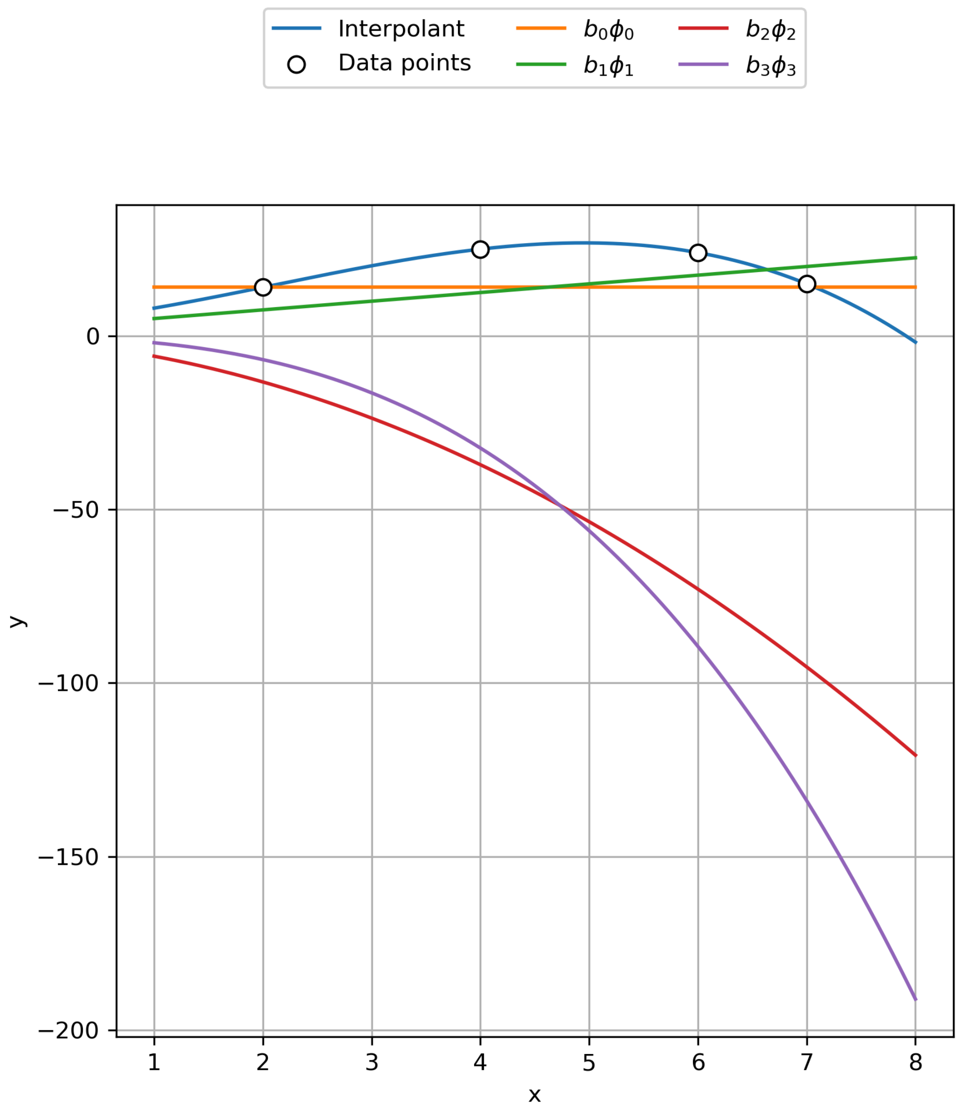
- Four basis functions for polynomial degree 3:

$$\phi^0(x) = 1, \qquad \phi^1(x) = (x-2)$$
$$\phi^2(x) = (x-2)(x-6), \quad \phi^3(x) = (x-2)(x-6)(x-4)$$

Construct the linear (triangular) system

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 \\ 1 & 2 & -4 & 0 \\ 1 & 5 & 5 & 15 \end{bmatrix}, \quad b = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}, \quad y = \begin{bmatrix} 14 \\ 24 \\ 25 \\ 15 \end{bmatrix}$$

- Solving $Ab = y$ (with $\mathbf{cond}(A) = 17$) we get $b = [14 \quad 2.5 \quad -1.5 \quad -0.26677]^T$



Interpolation error

Goal. Estimate the interpolation error that is made when replacing a given function f with its interpolating polynomial $\Pi_n f$ at the nodes $x_0, x_1, \ldots, x_n$.

Theorem 2 [Cauchy]. Let $f \in C^{n+1}(I)$ and suppose that $x_0, \ldots, x_n \in I$ are distinct, and that the polynomial $\Pi_n f$ satisfies

$$\Pi_n f(x_i) = f(x_i), \quad i = 0, \ldots, n.$$

Then

$$E_n[f](x) \doteq f(x) - \Pi_n f(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n}(x - x_i), \quad \text{where} \quad \omega_{n+1}(x) = \prod_{i=0}^{n}(x - x_i),$$

where $\xi \in I$, with I being the smallest open interval containing $x_0, \ldots, x_n$ and x.

Interpolation error

Proof. At the interpolation nodes the results holds trivially.
Let us introduce $\forall x \in I$ the following auxiliary function

$$G(t) = E_n[f](t) - \frac{\omega_{n+1}(t)}{\omega_{n+1}(x)} E_n[f](x)$$

Since $f \in C^{n+1}(I)$ and ω_{n+1} is a polinomial, then $G \in C^{n+1}(I)$ has at least $n+2$ zeros in I:
$\rightsquigarrow$ the $n+1$ interpolation points $x_0, x_1, \ldots, x_n$ and the point $t = x$.

Thanks to the mean value theorem, $G^{(1)}$ has at least $n+1$ distinct zeros and, by recursion, $G^{(j)}$ admits at least $n+2-j$ distinct zeros $\rightsquigarrow$ $G^{(n+1)}$ has at least one zero, which we denote by ξ.
On the other hand, since $E_n^{(n+1)}[f](t) = f^{(n+1)}(t)$ and $\omega_{n+1}^{(n+1)}(x) = (n+1)!$ we get

$$G^{(n+1)}(t) = f^{(n+1)}(t) - \frac{(n+1)!}{\omega_{n+1}(x)} E_n[f](x),$$

which, evaluated at $t = \xi$, gives the desired expression for $E_n[f](x)$.

Interpolation error

It is observed that while ξ exists, it is not generally possible to determine its exact value.

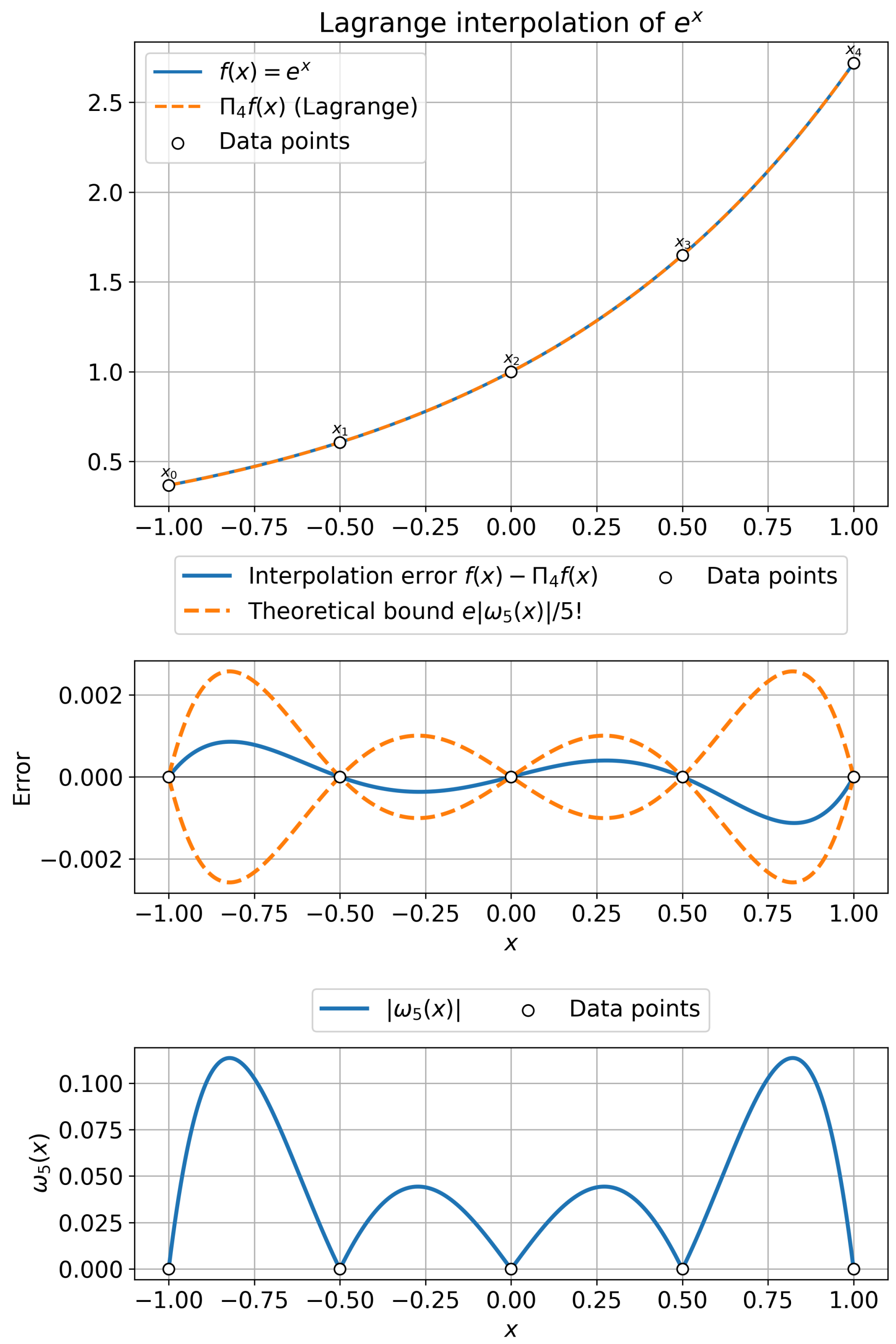
However, if $M = \max_{s \in I} \left| f^{(n+1)}(s) \right|$ is known, it is still possible to estimate the error as follows:

$$|E_n[f](x)| = \left| f^{(n+1)}(\xi) \frac{\prod_{i=0}^{n}(x - x_i)}{(n+1)!} \right| \leq M \frac{\left|\prod_{i=0}^{n}(x - x_i)\right|}{(n+1)!}.$$

Example. Let $f(x) = \exp(x)$ and $\Pi_4 f$ interpolates $f(x)$ at 5 equispaced points $\{x_i = -1 + \frac{i}{2}\}_{i=0}^{4}$.

- f exponential and increasing $\rightsquigarrow$ $M = \|f^{(5)}\|_\infty = \max_{x \in [-1,1]} |\exp(x)| = e$
- $\left|\prod_{i=0}^{n}(x - x_i)\right| \leq 0.12$

$$|E_4[f](x)| = \left| \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n}(x - x_i) \right| \leq \frac{e \cdot 0.12}{120} \approx 0.0027.$$



Convergence of polynomial interpolation

Remark. Given $x_0, h > 0$ and the $n+1$ equispaced nodes $x_i = x_{i-1} + h$ for $i = 1, \ldots, n$, one has

$$\left| \prod_{i=0}^{n} (x - x_i) \right| \leq n! \frac{h^{n+1}}{4}, \quad \text{so that} \quad \max_{x \in I} |E_n[f](x)| \leq \frac{h^{n+1}}{4(n+1)} \max_{x \in I} \left| f^{(n+1)}(x) \right|$$

- We cannot deduce that $\|E_n[f](x)\|_\infty \xrightarrow{n\to\infty} 0$, only if $\left| f^{(n+1)}(x) \right|$ doesn't grow too rapidly with n.

Theorem 3. For any distribution of nodes, there exists at least one function $f \in C(I)$, I bounded, s.t.

$$\|E_n[f]\|_\infty \overset{n\to\infty}{\not\to} 0.$$

Theorem 4. For every function $f \in C(I)$, I bounded, there exists at least one distribution of nodes s.t.

$$\|E_n[f]\|_\infty \overset{n\to\infty}{\to} 0.$$

Convergence of polynomial interpolation

We introduce a lower triangular matrix X of *infinite size*, called the **interpolation matrix** on $I = [a, b]$
$\rightsquigarrow$ for any $n \geq 0$, the $n+1$-th row of X contains $n+1$ distinct interpolation nodes, that for a given f, uniquely define an interpolating polynomial $\Pi_n f$ of degree n.

Denoted the **interpolation error** as $E_{n,\infty}(X) = \|f - \Pi_n f\|_\infty$ and by $\Pi_n^* f$ the **best approximation polynomial** such that $E_n^* = \|f - \Pi_n^* f\|_\infty \leq \|f - r_n\|_\infty, \ \forall r_n \in \mathbb{P}_n$

Theorem 5. Given $f \in C^0(I)$, X interpolation matrix in I, and $l_j^{(n)} \in \mathbb{P}_n$ the j-th characteristic polynomial associated with the $n+1$-th row of X, i.e. $l_j^{(n)}(x_{nk}) = \delta_{jk}$, for $j, k \geq 0$ then

$$E_{n,\infty}(X) \leq E_n^*(1 + \Lambda_n(X)), \quad \text{with} \quad \Lambda_n(X) = \left\| \sum_{j=0}^{n} |l_j^{(n)}| \right\|_\infty$$

where $\Lambda_n(X)$ denotes the **Lebesgue constant** of X.

$\rightsquigarrow$ Notice that E_n^* does not depend on X, so the effects of X on $E_{n,\infty}(X)$ is only contained in $\Lambda_n(X)$.

Runge counterexample

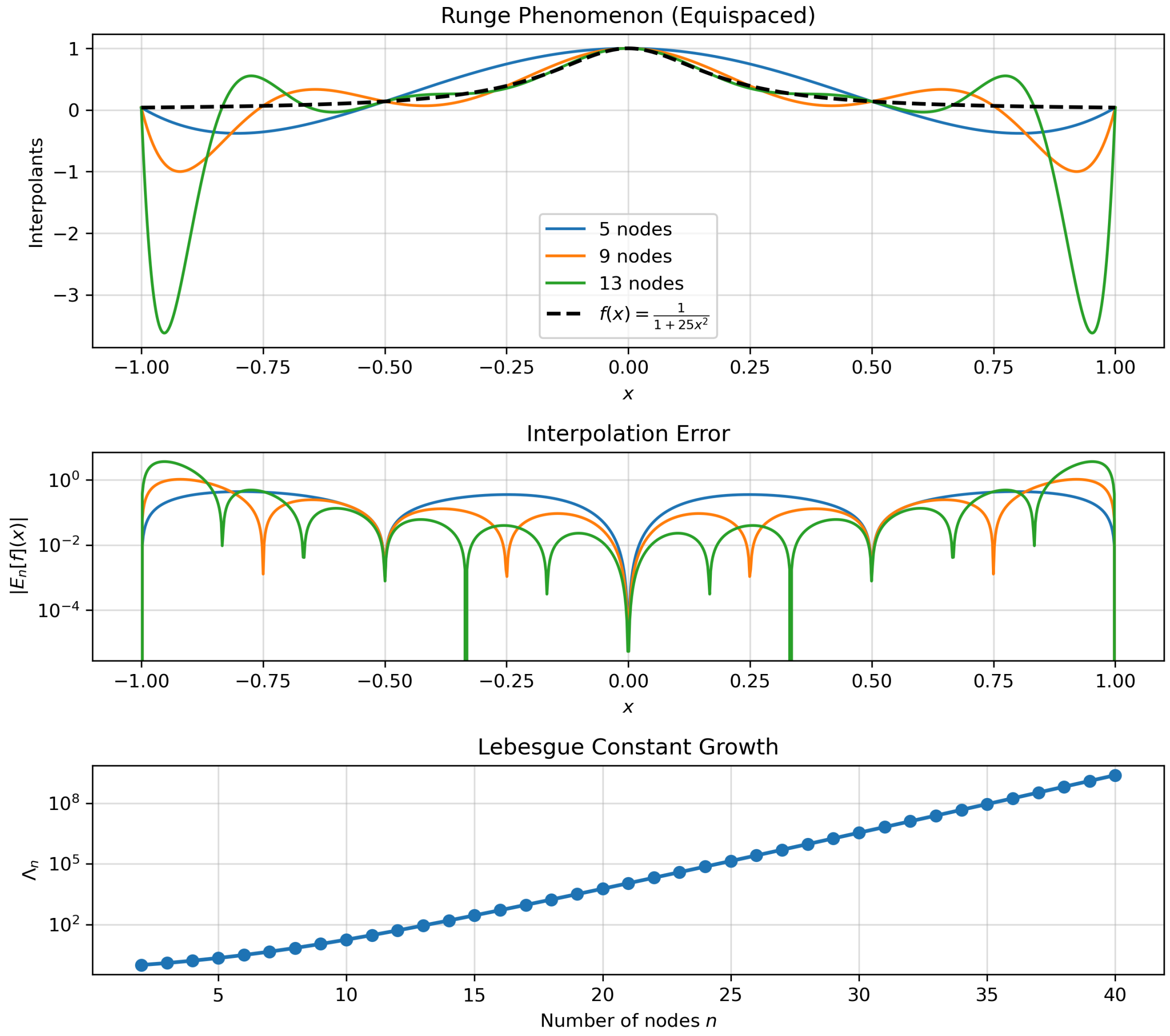
Let us consider the Lagrange interpolation with equispaced nodes $\{x_i\}_{i=0}^n \in [-1, 1]$ for

$$f(x) = \frac{1}{1+25x^2}.$$

$\rightsquigarrow \Pi_n f$ does not converge uniformly to f

In particular, for each choice of X there exist a constant $C > 0$ such that

$$\Lambda_n(X) \geq \left[\frac{2}{\pi}\log(n+1) - C\right] \xrightarrow{n\to\infty} \infty$$



Stability of Polynomial Interpolation

Remarks.

- Interpolating polynomials of high degree is expensive to determine and evaluate
- Coefficients of polynomial may be poorly determined due to ill-conditioning of linear system
- High-degree polynomials necessarily have lots of "oscillations" which may bear no relation to data

Perturbations. Let us consider a set of function values $y_i = \tilde{f}(x_i)$ which is a perturbation (round-off, experiment, noise) of the data $f(x_i)$ relative to the nodes $\{x_i\}_{i=0}^n \in I$.
Denoting by $\Pi_n \tilde{f}$ the Lagrange interpolant for $\{(x_i, \tilde{f}_i)\}_{i=0}^n$ it holds

$$\begin{aligned}\|\Pi_n f(x) - \Pi_n \tilde{f}(x)\|_\infty &= \max_{x \in I} \left| \sum_{j=0}^n (f(x_j) - \tilde{f}(x_j)) l_j(x) \right| \\ &\leq \Lambda_n(X) \max_{j=0,\ldots,n} |f(x_j) - \tilde{f}(x_j)|\end{aligned}$$

$\rightsquigarrow$ small data perturbations give rise to small changes on the interpolant only for small Lebesgue constant.

Lebesgue constant

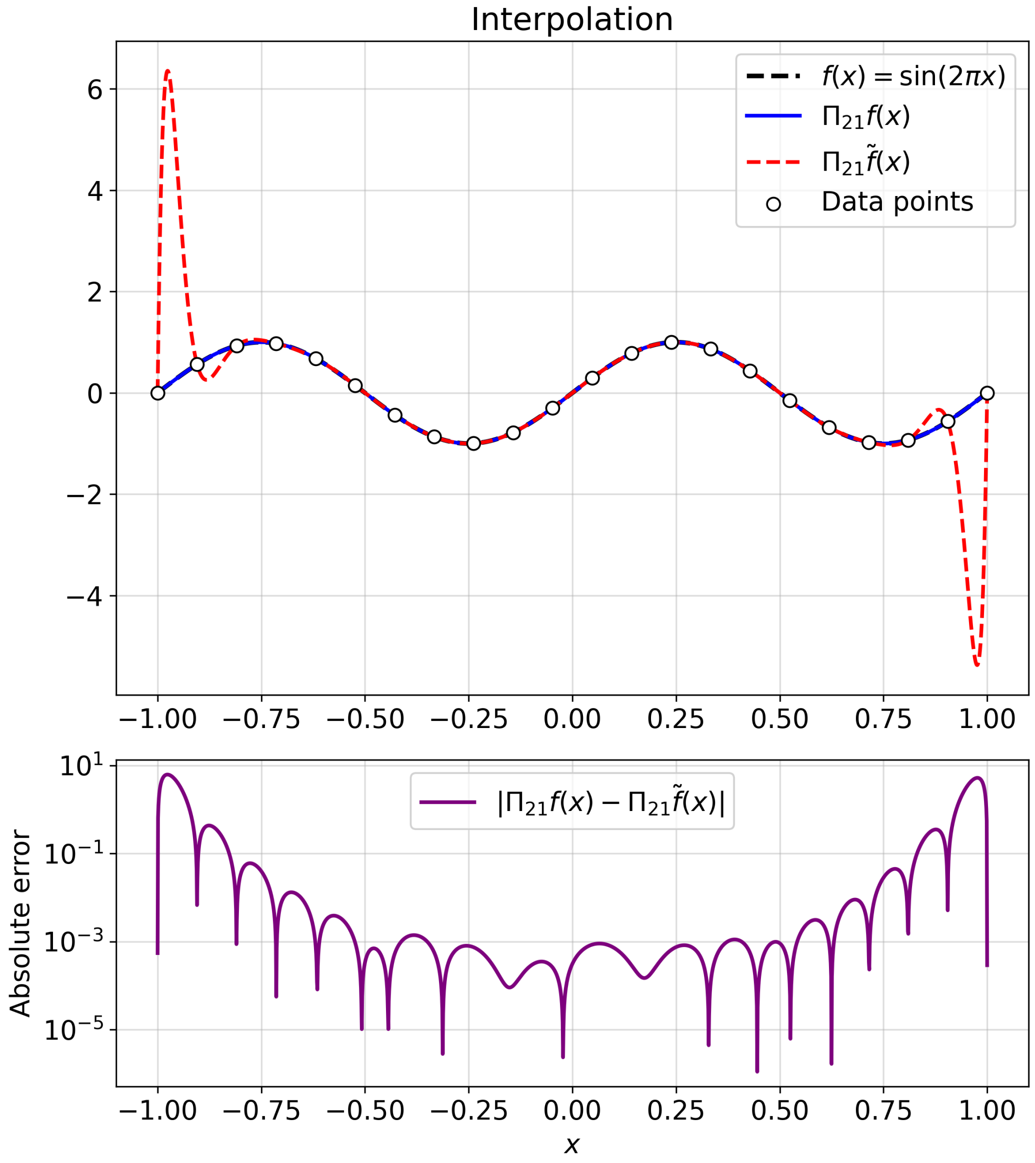
This constant plays the role of the **condition number** for the interpolation problem.

For Lagrange interpolation on $n+1$ equispaced nodes in $I=[-1,1]$, it can be proved that

$$\Lambda_n(X) \sim \frac{2^{n+1}}{n \log n}$$

For large values of n this can become unstable.

Example. Interpolate $f(x)=\sin(2\pi x)$ in $I=[-1,1]$ with $n+1=22$ equispaced nodes, such that $\max_{x_i} |f(x_i)-\tilde{f}(x_i)| = 9.5\times 10^{-4}$.



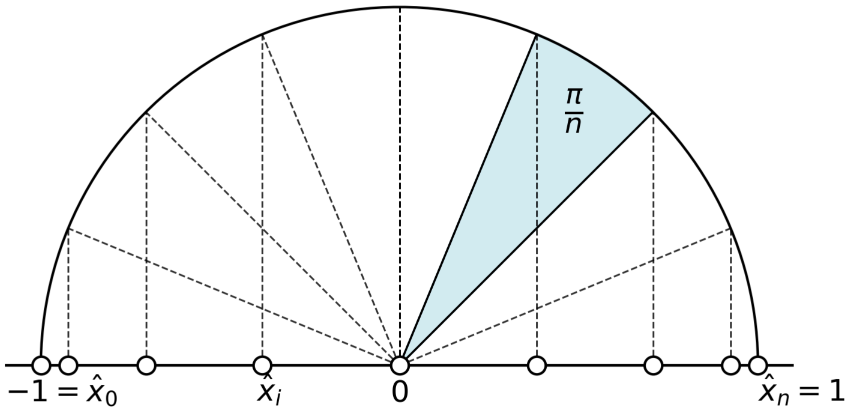
Chebyshev-Gauss-Lobatto points

Runge's phenomenon can be avoided if a suitable distribution of nodes is used. In particular, in an arbitrary interval $[a, b]$, we can consider the so called **Chebyshev-Gauss-Lobatto** nodes

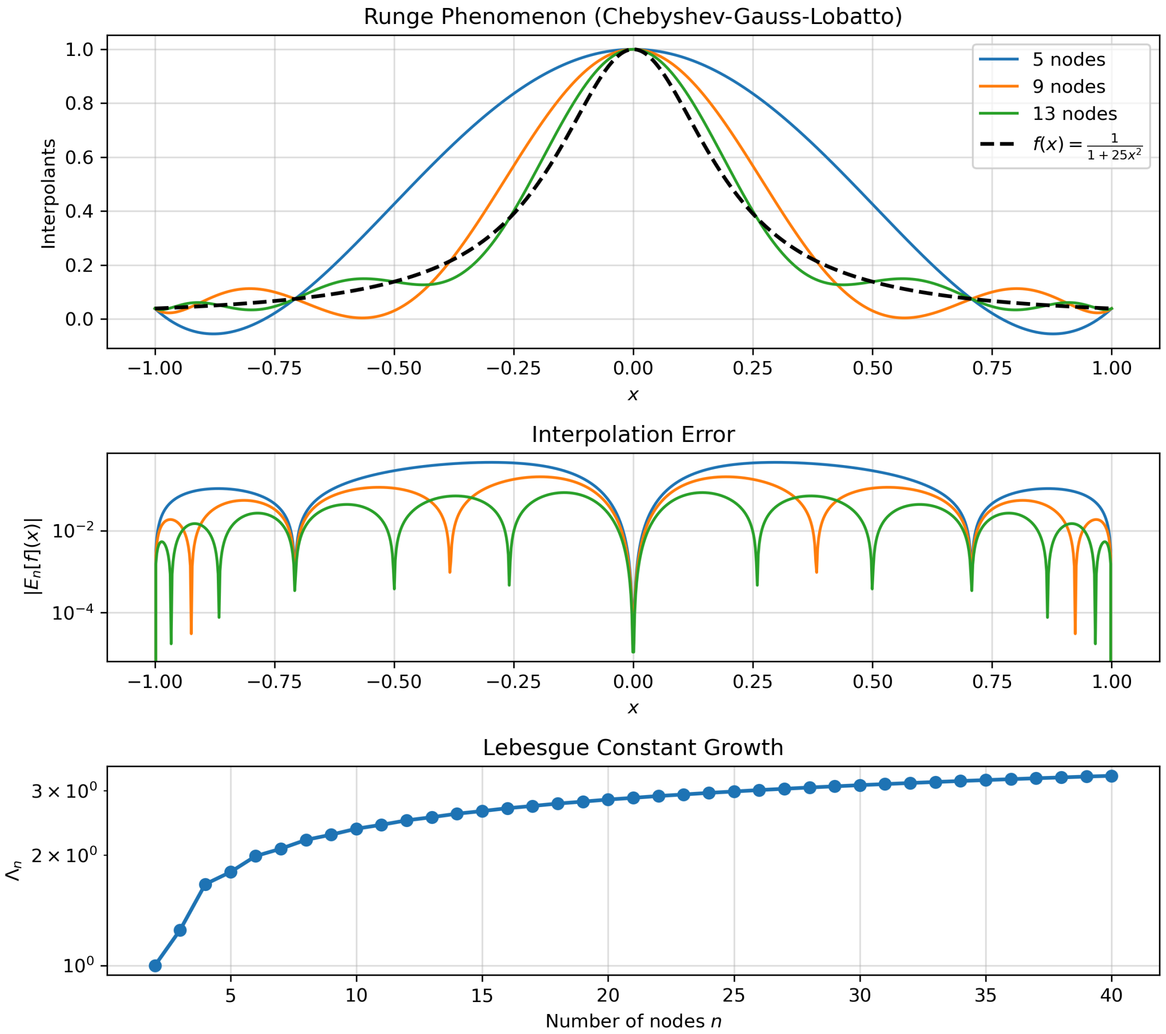
$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\hat{x}_i, \quad \text{where } \hat{x}_i = -\cos\left(\frac{\pi i}{n}\right), \ \text{ for } i = 0, \ldots, n$$

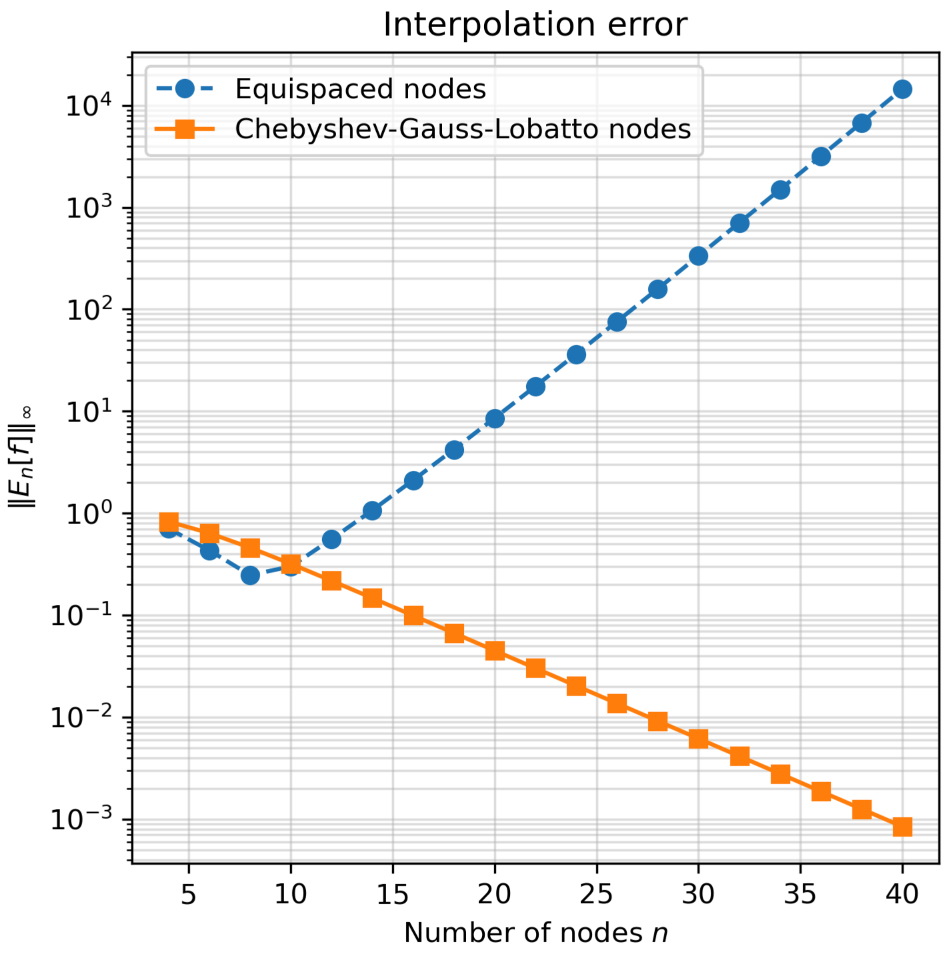
⇝ If f is a continuous and differentiable function in I, then $\Pi_n f(x) \overset{n\to\infty}{\longrightarrow} f(x)$ for all $x \in I$.

They are the abscissas of equispaced nodes on the unit semi-circumference, lie inside $[a, b]$ and are clustered near the endpoints of the interval.



Chebyshev-Gauss-Lobatto points





Barycentric Lagrange Interpolation

The main drawbacks of the Lagrange form of the interpolation are

- each evaluation of Π_n requires $\mathcal{O}(n^2)$ additions and multiplications
- adding a new data pair (x_{n+1}, y_{n+1}) requires a new computation from scratch
- the computations can be numerically unstable.

Barycentric Lagrange interpolation is an alternative to Lagrange formula so that it can be evaluated and updated in $\mathcal{O}(n)$ operations. Starting from the characteristic polynomial $l_j(x)$ one can write

$$l_j(x) = \prod_{k \neq j}^{n} \frac{x - x_k}{x_j - x_k} = \underbrace{\left[\prod_{k=0}^{n} (x - x_k)\right]}_{\omega_{n+1}(x)} \frac{q_j}{x - x_j} \quad \text{where } \; q_j = \left[\prod_{k \neq j}^{n} (x_j - x_k)\right]^{-1}$$

Then we can write the *first form* of the *barycentric interpolation* formula with q_j *barycentric weights* as

$$\Pi_n(x) = \omega_{n+1}(x) \sum_{j=0}^{n} \frac{q_j}{(x - x_j)} y_j$$

Barycentric Lagrange Interpolation

- Computing $n+1$ cofficients costs $\mathcal{O}(n^2)$, while evaluating Π_n costs $\mathcal{O}(n)$ operations for each x.
- If a new pair (x_{n+1}, y_{n+1}) is added, the following $\mathcal{O}(n)$ operations are required:
  - for $j = 0, \dots, n$ divide each q_j by $x_j - x_{n+1}$ for a cost of $n+1$ operations;
  - compute the weight q_{n+1} as before for a cost of $n+1$ additional operations.

In practice, the *second form* of the *barycentric interpolation* formula is used.
Suppose we interpolate the constant values $y_i = 1$ for $i = 0, \dots, n$, then it holds:

$$1 \equiv \Pi_n(x) = \omega_{n+1}(x) \sum_{j=0}^{n} \frac{q_j}{(x - x_j)}$$

thus, dividing first form by this term and cancelling the common factor (nodal polynomial) one obtains

$$\Pi_n(x) = \frac{\sum_{j=0}^{n} \frac{q_j}{(x-x_j)} y_j}{\sum_{j=0}^{n} \frac{q_j}{(x-x_j)}}$$

Barycentric Lagrange Interpolation

The second form of the barycentric Lagrange interpolation formula has a special symmetry: the weights q_j appear both in the denominator and in the numerator (weighted by data y_j).

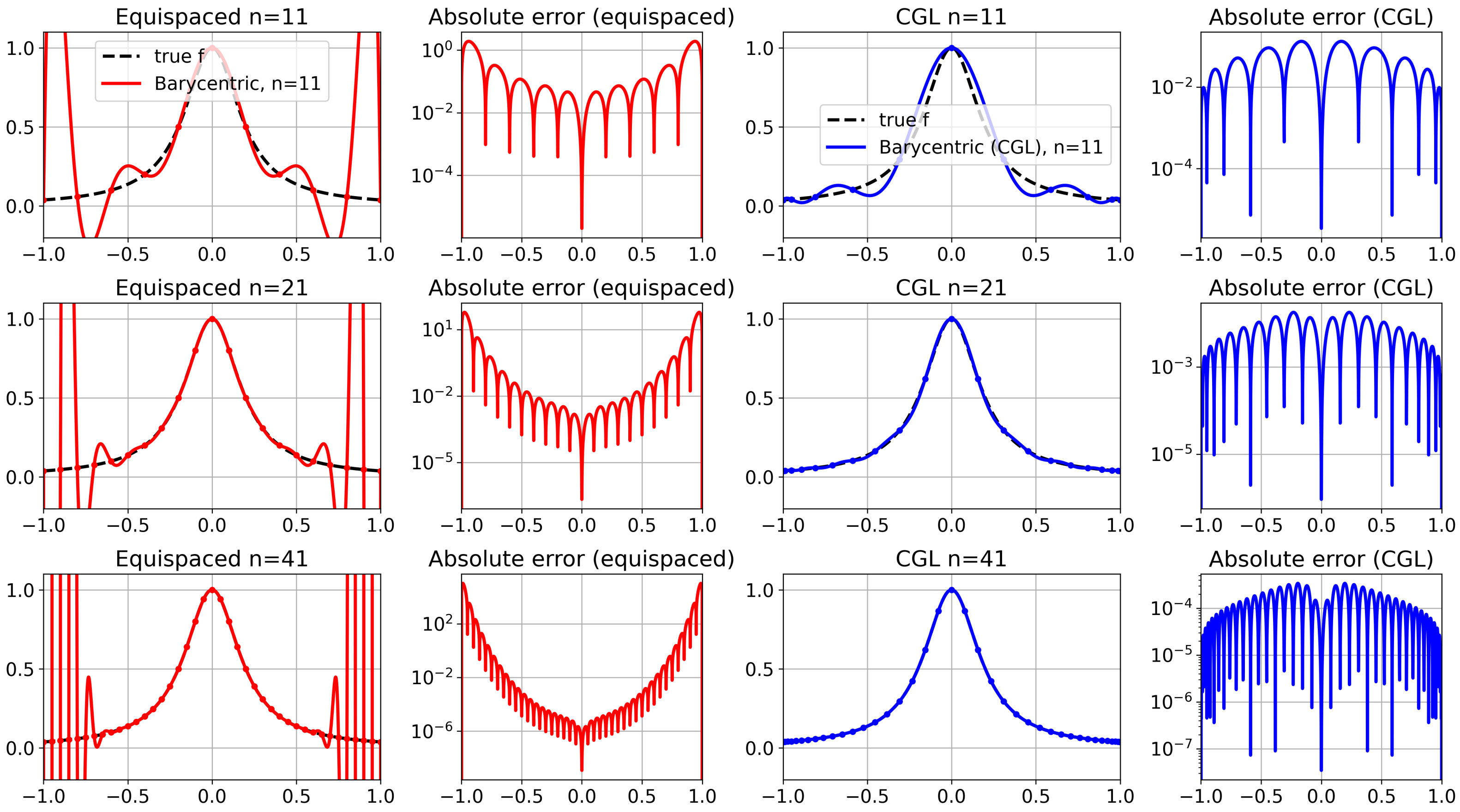
- Possible common factors in all the weights q_j may be cancelled without affecting the value of $\Pi_n(x)$.
- More stable than the Newton formula if one avoids division by zero in the expression of weights q_j.
- When $x \approx x_j$ the quantity $\frac{q_j}{x-x_j}$ will be very large and we would expect a risk of inaccuracy.
- However, the same inaccuracy appears in both numerator and denominator and they cancel out.

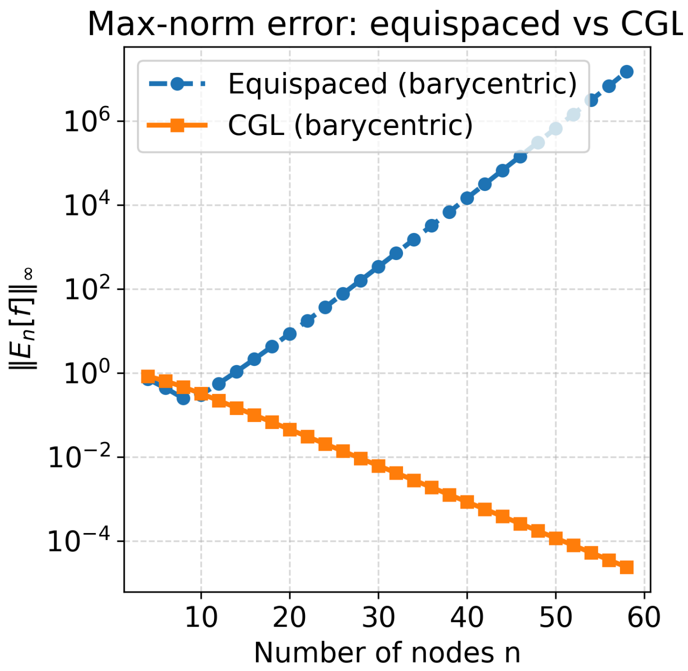
$\rightsquigarrow$ For equidistant nodes with spacing $h = 2/n$ on the interval $[-1, 1]$ one has $q_j = (-1)^j \binom{n}{j}$, while for a generic interval $[a, b]$ it should be multiplied by $2^n(b-a)^{-n}$.

If the barycentric weights vary widely (equispaced case) the interpolation problem must be ill-conditioned:

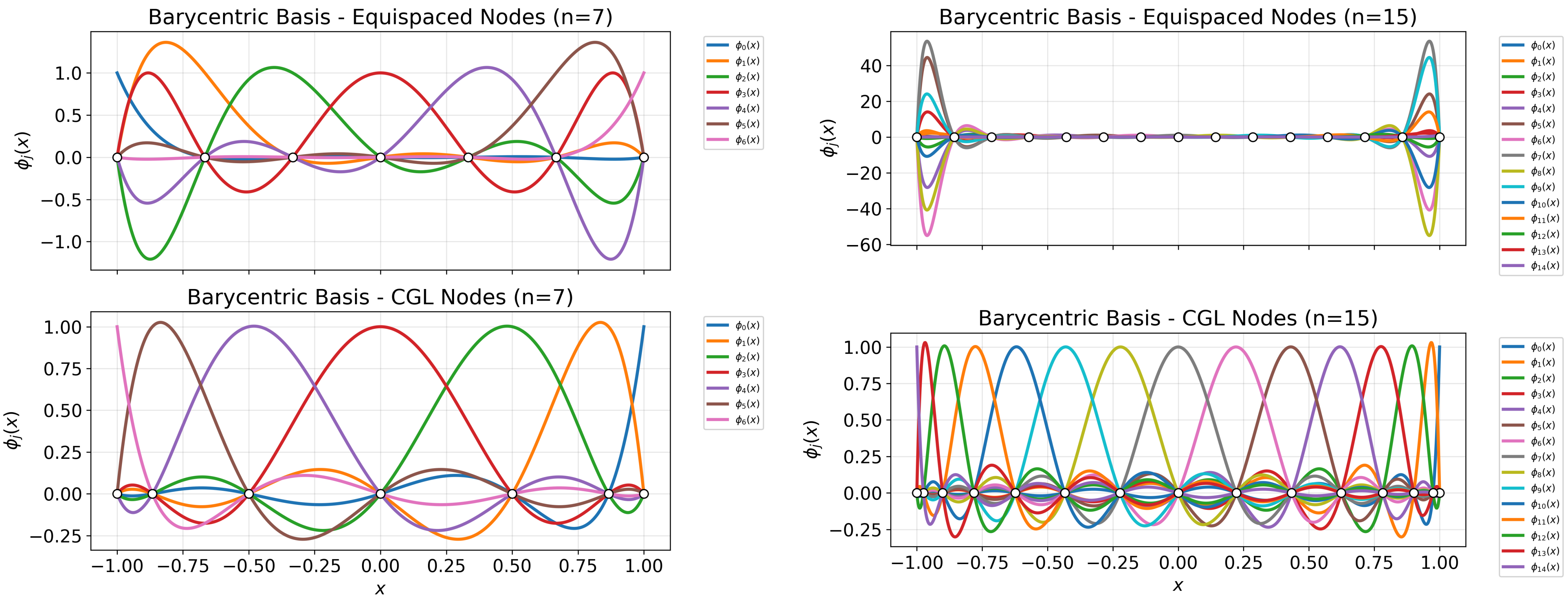
$$\Lambda_n(X) = \frac{1}{2n^2} \frac{\max_{j=0,\ldots,n} |q_j|}{\min_{j=0,\ldots,n} |q_j|}.$$

Barycentric Lagrange Interpolation (Runge)





Barycentric Lagrange Interpolation



Trigonometric interpolation

Approximate a periodic function $f : [0, 2\pi] \to \mathbb{C}$ with $f(0) = f(2\pi)$, by a **trigonometric polynomial** $\tilde{f}$ interpolating f at the equispaced $n+1$ nodes $x_j = \frac{2\pi j}{n+1}$, such that $f(x_j) = \tilde{f}(x_j)$, for $j = 0, \ldots, n$.

The **trigonometric interpolant** $\tilde{f}$ is a linear combination of sines and cosines (with $M = \frac{n}{2}$ and n even)

$$\tilde{f} = \frac{a_0}{2} + \sum_{k=1}^{M} [a_k \cos(kx) + b_k \sin(kx)] = \sum_{k=-M}^{M} c_k e^{ikx},$$

whose complex coefficients $a_k = c_k + c_{-k}$, $b_k = i(c_k - c_{-k})$, and c_k for $k = 0, \ldots, M$, are unknown.

Remark. If n is odd then the sum is with $k = -(M+1), \ldots, M+1$ and $M = \frac{n-1}{2}$, but being $n+2$ unknowns one need to impose $c_{-(M+1)} = c_{(M+1)}$, and

$$\tilde{f} = \sum_{k=-M}^{M} c_k e^{ikx} + 2c_{(M+1)} \cos((M+1)x)$$

$\rightsquigarrow$ This is called *discrete Fourier series* of f.

Trigonometric interpolation

To compute the coefficients c_k, with $k = -M, \ldots, M$, we impose the *interpolation condition* at $x_j = jh$ with $h = \frac{2\pi}{n+1}$, for $j = 0, \ldots, n$ such that (with $\mu = 0$ if n even, and $\mu = 1$ if n odd)

$$\sum_{k=-M}^{M} c_k e^{ikjh} + 2\mu c_{(M+1)} \cos((M+1)jh) = f(jh),$$

that multiplied by $e^{-imx_j} = e^{-imjh}$ where $m = -M, \ldots M + \mu$, and summed over j gives

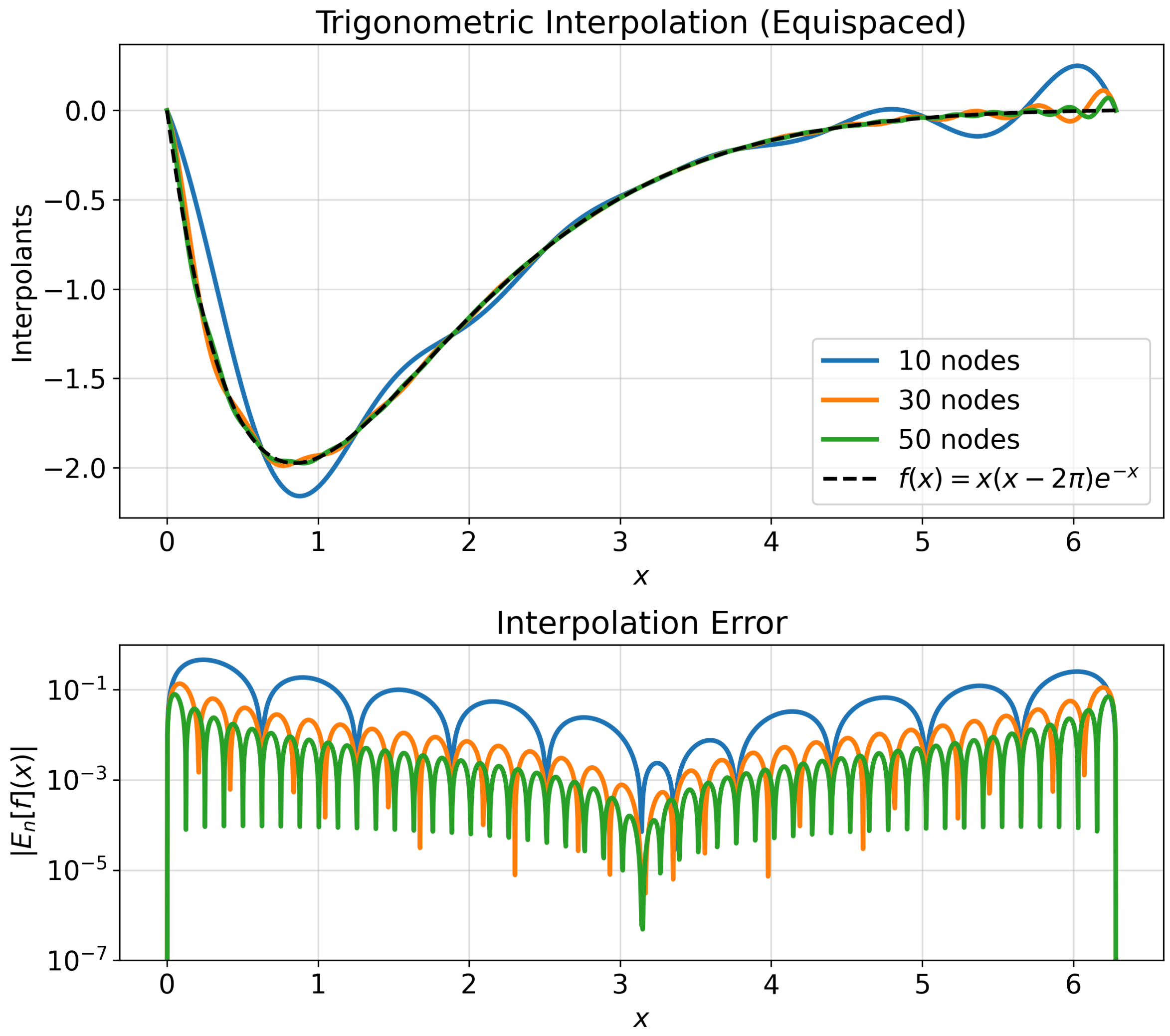
$$\sum_{j=0}^{n} \sum_{k=-M}^{M} c_k e^{ikjh} e^{-imjh} + 2\mu c_{(M+1)} \sum_{j=0}^{n} \cos((M+1)jh) e^{-imjh} = \sum_{j=0}^{n} f(jh) e^{-imjh},$$

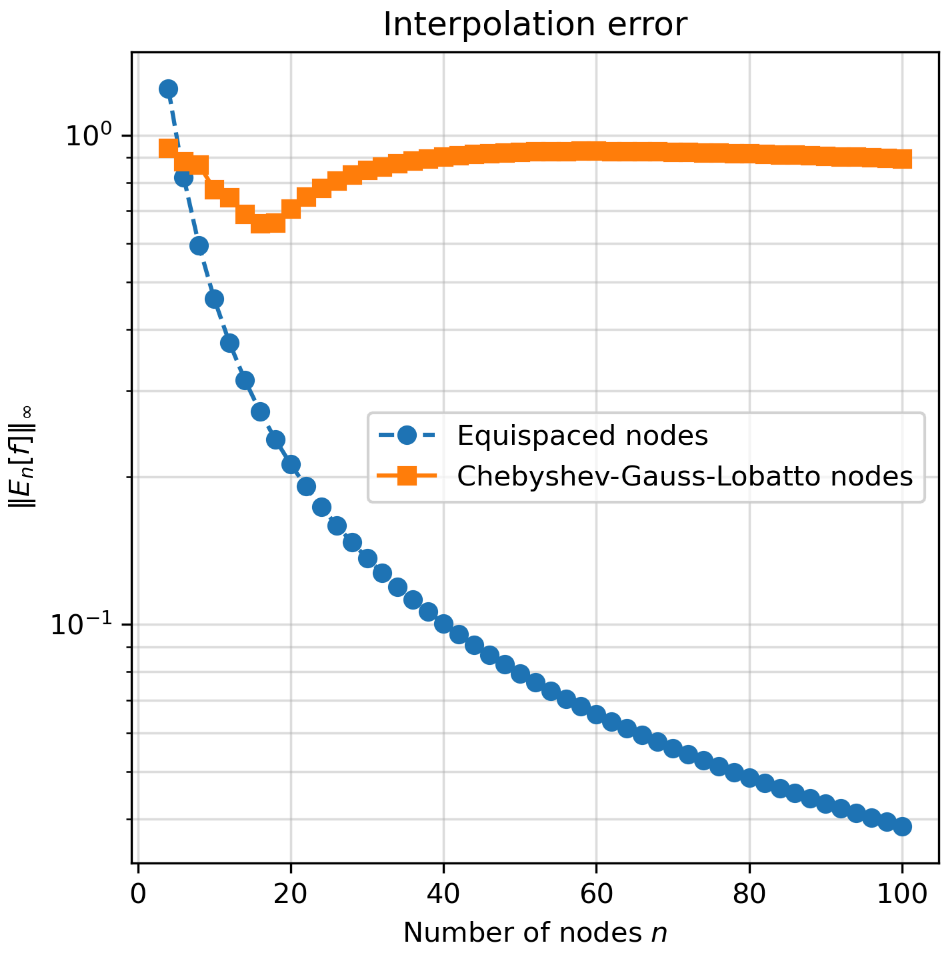
that exploiting trigonometric identities provides the explicit expression for the coefficients of $\tilde{f}$:

$$c_k = \frac{1}{n+1} \sum_{j=0}^{n} f(x_j) e^{-ikjh}, \quad \text{for} \quad k = -M, \ldots, M, \quad \text{and} \quad c_{\pm(M+1)} = \frac{1}{2(n+1)} \sum_{j=0}^{n} (-1)^j f(x_j), \quad \text{if } n \text{ is odd.}$$

If f is real valued, then $c_{\pm(M+1)}$ are real and $c_{-k} = \overline{c_k}$, for $k = -M, \ldots, M$, and $\tilde{f}$ is also real valued.
Fast Fourier Transform (FFT) allows fast computation of all coefficients c_k with a cost of $\mathcal{O}(n \log_2 n)$.

Trigonometric interpolation





Trigonometric interpolation

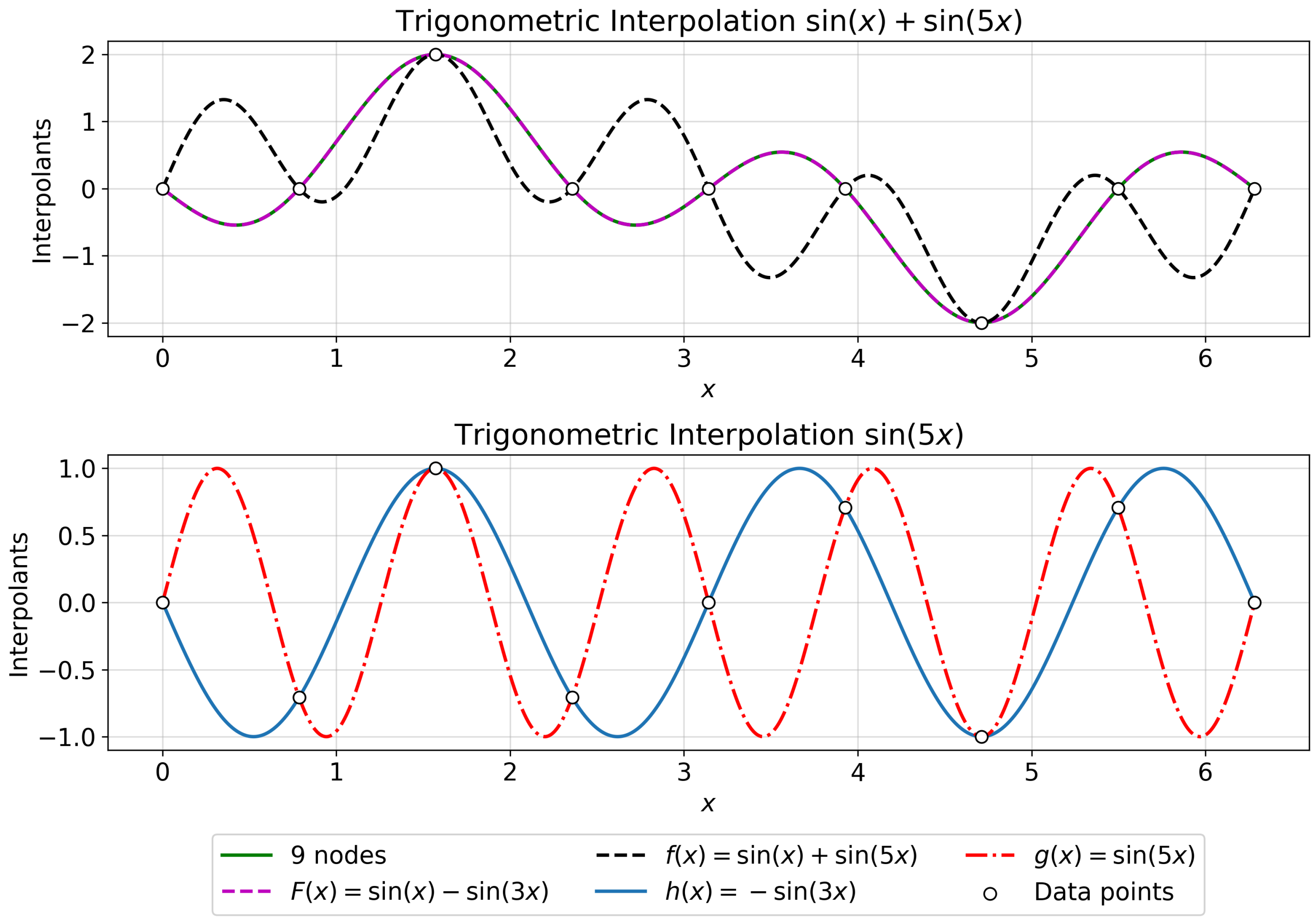
Approximate $f(x) = \sin(x) + \sin(5x)$ using $n = 9$ equispaced nodes in the interval $[0, 2\pi]$.

$\rightsquigarrow$ in some intervals the trigonometric interpolant even shows a phase inversion.

For all nodes x_j, $g(x) = \sin(5x)$ is the same as $h(x) = -\sin(3x)$.

This **aliasing** problem let us approximate the function $F(x) = \sin(x) - \sin(3x)$.

$\rightsquigarrow$ the number of nodes is not enough to resolve the highest frequencies.



Piecewise Polynomial Interpolation

- For equispaced points, uniform convergence of Π_n to f is not guaranteed as $n \to \infty$.
- For smooth function f, the interpolant at Chebyshev nodes provides an accurate approximation.
- What to do when f is nonsmooth or when f is only known through its values at a set of given points?
- Exploiting equispaced nodes is clearly computationally convenient, and if sufficiently small interpolation intervals are considered, Lagrange interpolation of low degree is very accurate.

$\rightsquigarrow$ Introduce a partition $\mathcal{T}_h$ of $[a, b]$ into K subintervals $I_j = [x_j, x_{j+1}]$ of length h_j, with $h = \max_{0 \le j \le K-1} h_j$, such that $[a, b] = \cup_{j=0}^{K-1} I_j$ and then to employ Lagrange interpolation on each I_j using $k+1$ equispaced nodes $\{x_j^{(i)}, 0 \le i \le k\}$ with a small k.

Piecewise Polynomial Interpolation

For $k \geq 1$, we introduce on $\mathcal{T}_h$ the **piecewise polynomial space**

$$X_h^k = \{v \in C^0([a,b]) : v|_{I_j} \in \mathbb{P}_k(I_j), \forall I_j \in \mathcal{T}_h\}$$

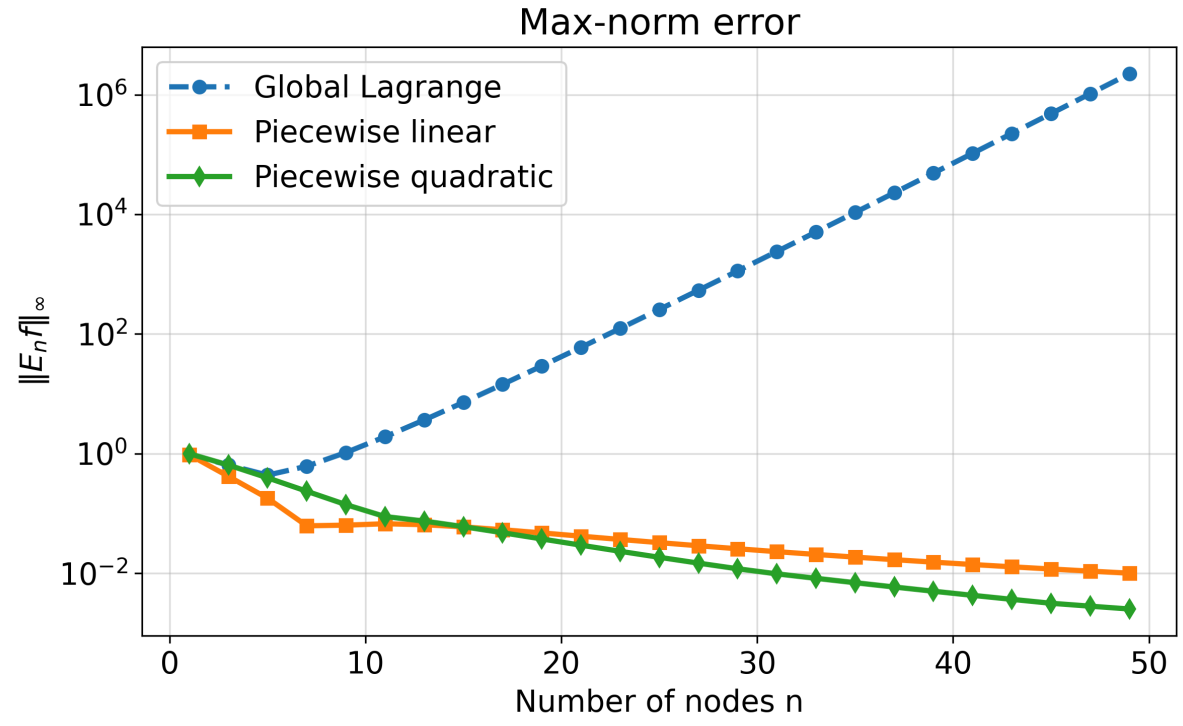
which is the space of the continuous functions over $[a,b]$ whose restrictions on each I_j are polynomials of degree $\leq k$. Then, for any continuous function f in $[a,b]$, the piecewise interpolation polynomial $\Pi_h^k f$ coincides on each I_j with the interpolating polynomial of $f|_{I_j}$ at the $k+1$ nodes $x_j^{(i)}$, $0 \leq i \leq k$.

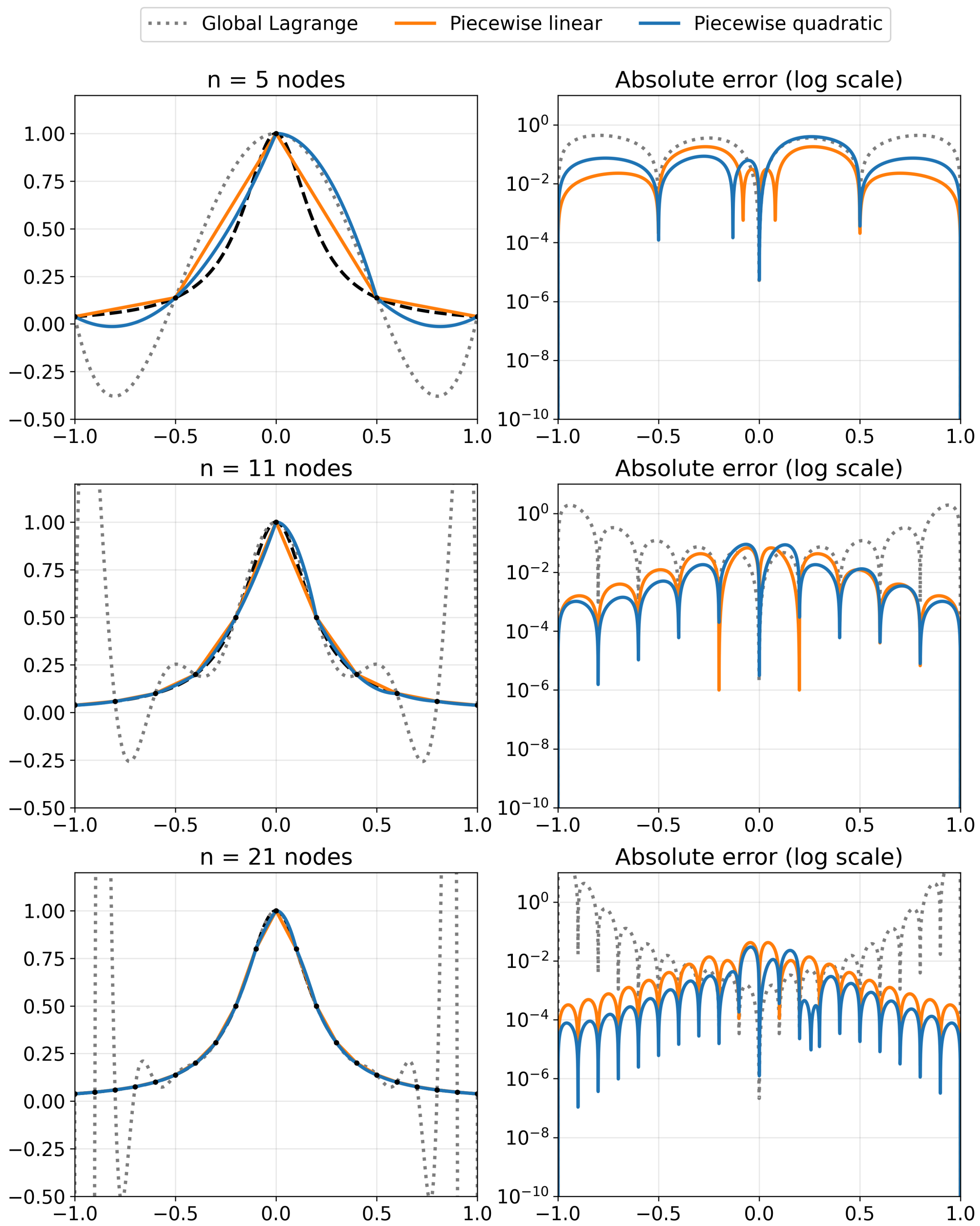
If $f \in C^{k+1}([a,b])$, we use the interpolation error formula within each interval to obtain the following error estimate:

$$\|E_n[f](x)\|_\infty = \|f(x) - \Pi_h^k f(x)\|_\infty \leq ch^{k+1}\|f^{(k+1)}\|_\infty$$

Note that a small interpolation error can be obtained even for low k provided that h is sufficiently "small".

Piecewise Polynomial Interpolation





Hermite Interpolation

Generalize Lagrange interpolation when also derivatives values of a function f are available at nodes x_i.

Given $(x_i, y_i^{(k)} = f^{(k)}(x_i))$, with $i = 0, \ldots, n$, $m_i \in \mathbb{N}$, $k = 0, \ldots, m_i$, and $N = \sum_{i=0}^{n}(m_i + 1)$, then there exists a unique polynomial $H_{N-1} \in \mathbb{P}_{N-1}$, called **Hermite interpolation polynomial**

$$H_{N-1}(x) = \sum_{i=0}^{n} \sum_{k=0}^{m_i} y_i^{(k)} L_{ik}(x), \quad \text{s.t.} \quad H_{N-1}^{(k)}(x_i) = y_i^{(k)}, \quad i = 0, \ldots, n, \;\; k = 0, \ldots, m_i,$$

The functions $L_{ik} \in \mathbb{P}_{N-1}$ are the **Hermite characteristic polynomials**

$$L_{ij}(x) = l_{ij}(x) - \sum_{k=j+1}^{m_i} l_{ij}^{(k)}(x_i) L_{ik}(x), \quad j = m_i - 1, m_i - 2, \ldots, 0,$$

where $L_{im_i}(x) = l_{im_i}(x)$, for $i = 0, \ldots, n$, and

$$l_{ij}(x) = \frac{(x - x_i)^j}{j!} \prod_{\substack{k=0 \\ k \neq i}}^{n} \left(\frac{x - x_k}{x_i - x_k} \right)^{m_k + 1}, \; i = 0, \ldots, n, \; j = 0, \ldots, m_i.$$

Hermite Interpolation

Thus, they are defined through the relations

$$\frac{d^p}{dx^p}(L_{ik})(x_j) = \begin{cases} 1 & \text{if } i = j \text{ and } k = p, \\ 0 & \text{otherwise.} \end{cases}$$

As for the interpolation error, the following estimate holds

$$f(x) - H_{N-1}(x) = f^{(N)}(\xi)\Omega_N(x)/N!, \quad \forall x \in \mathbb{R},$$

where $\xi \in I(x; x_0, \ldots, x_n)$ and Ω_N is the polynomial of degree N defined by

$$\Omega_N(x) = (x - x_0)^{m_0+1}(x - x_1)^{m_1+1} \cdots (x - x_n)^{m_n+1}.$$

Example. If $m_i = 1$ for $i = 0, \ldots, n$, then $N = 2n + 2$ the so-called **osculating polynomial** is given by

$$\sum_{i=0}^{n} y_i A_i(x) + y_i^{(1)} B_i(x),$$

where $A_i(x) = (1 - 2(x - x_i){l_i}'(x_i))l_i(x)^2$ and $B_i(x) = (x - x_i)l_i(x)^2$, for $i = 0, \ldots, n$.

Hermite Interpolation