

# Applied Math

## Interpolation

Federico Pichi

11 November 2025



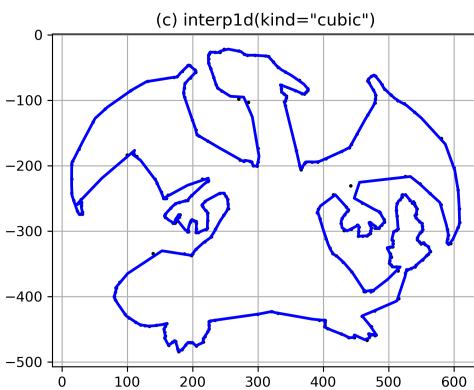
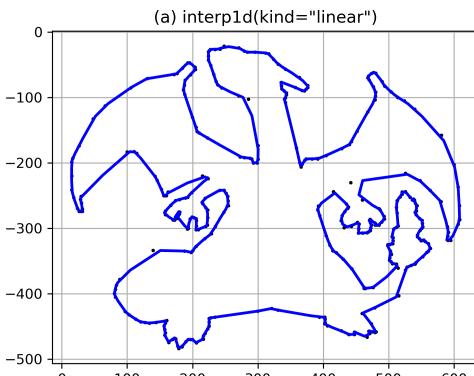
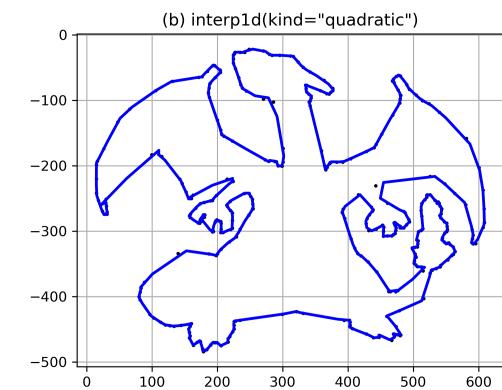
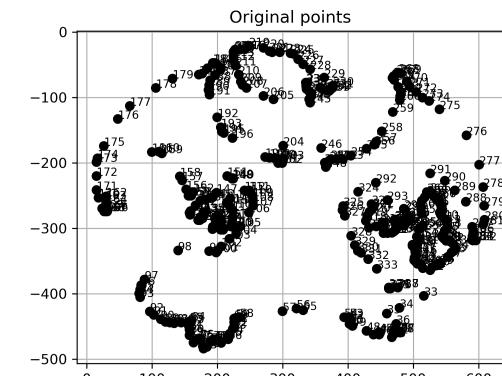
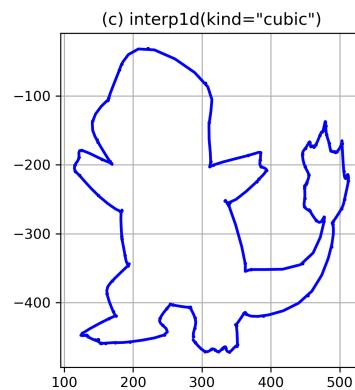
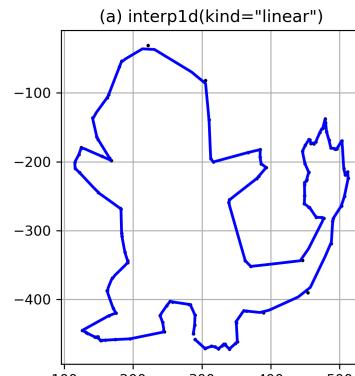
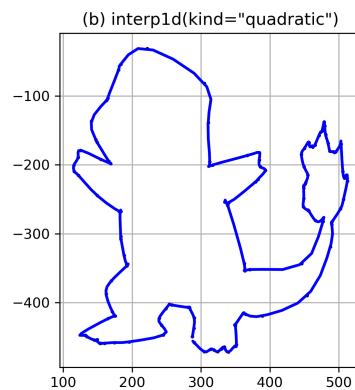
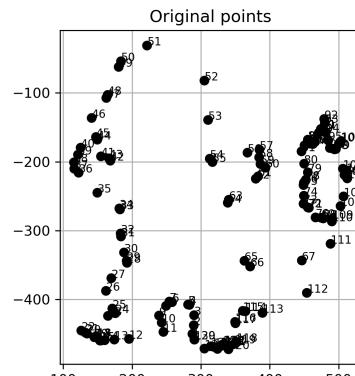
# Outline

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

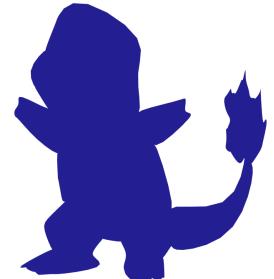
# Motivations

- **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.

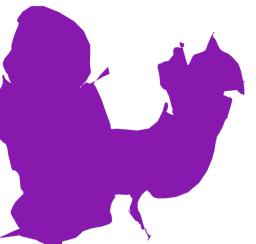
# Real motivations



$t = 0.00$



$t = 0.25$



$t = 0.50$



$t = 0.75$

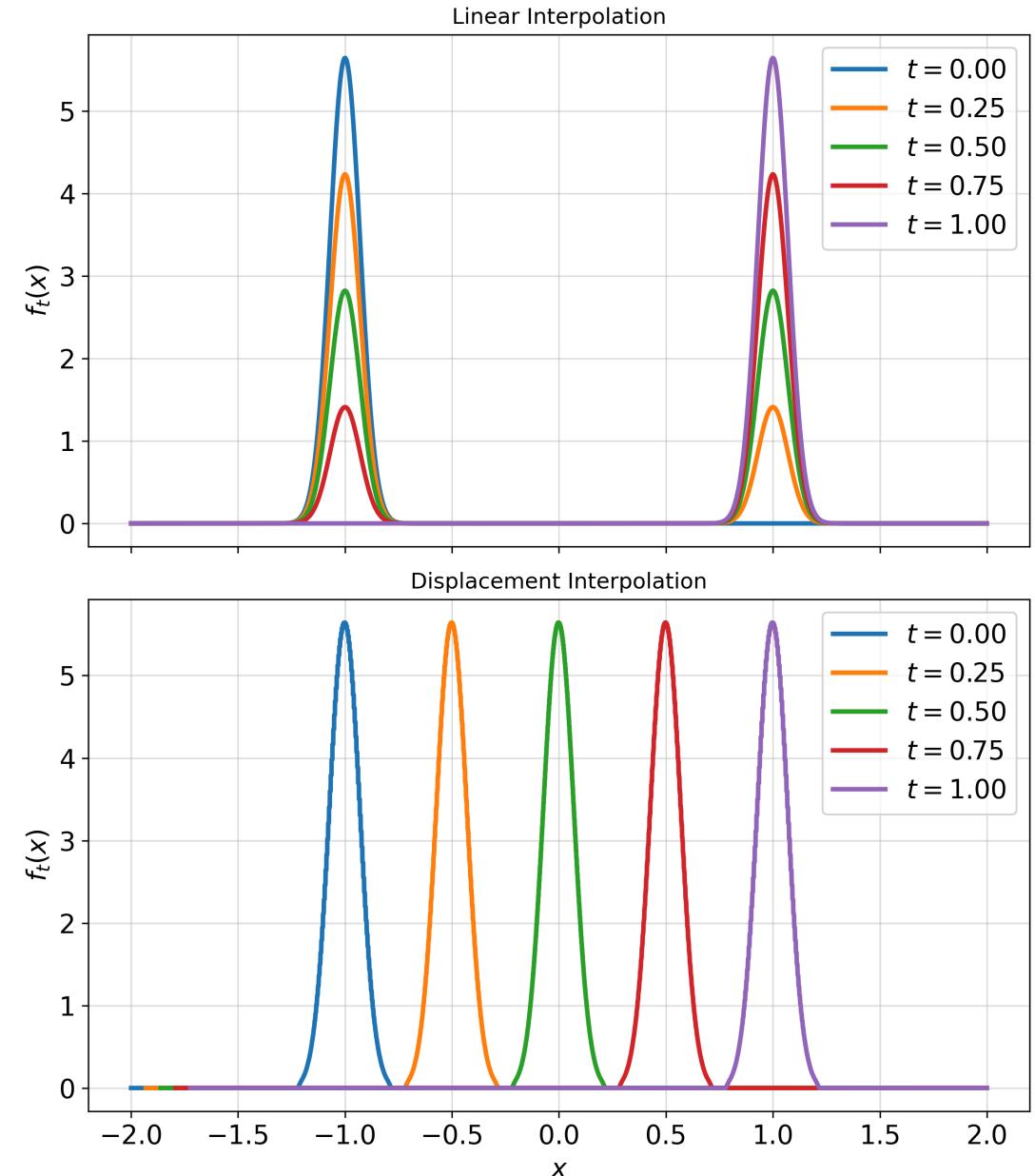


$t = 1.00$



# 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

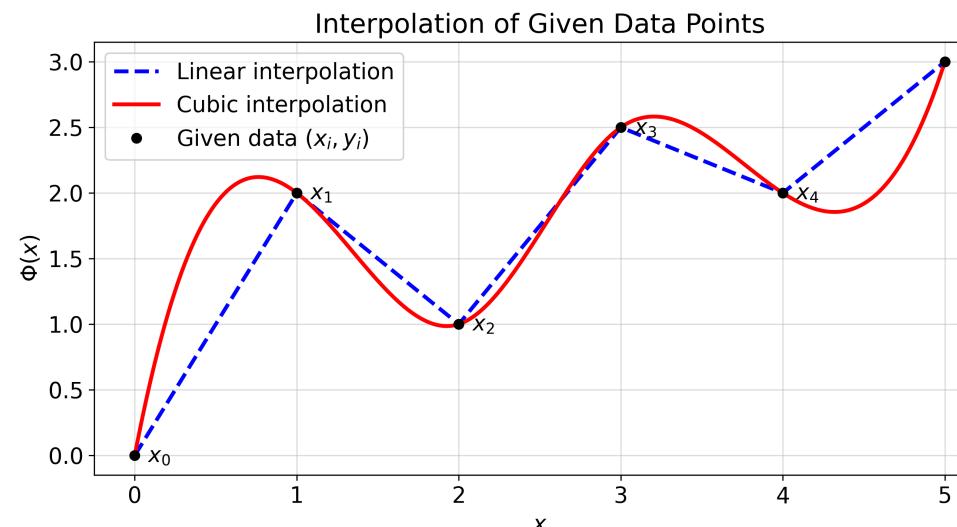
**Interpolation.** Given  $n + 1$  pairs  $(x_i, y_i)$ , we look for a function  $\Phi = \Phi(x)$  such that

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

where  $y_i$  are some given values representing the exact value for a function  $f$  at  $x_i$  or experimental data.

$\Phi$  is the *interpolant* that approximate such function/distribution interpolating the  $y_i$  at the nodes  $x_i$ .

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



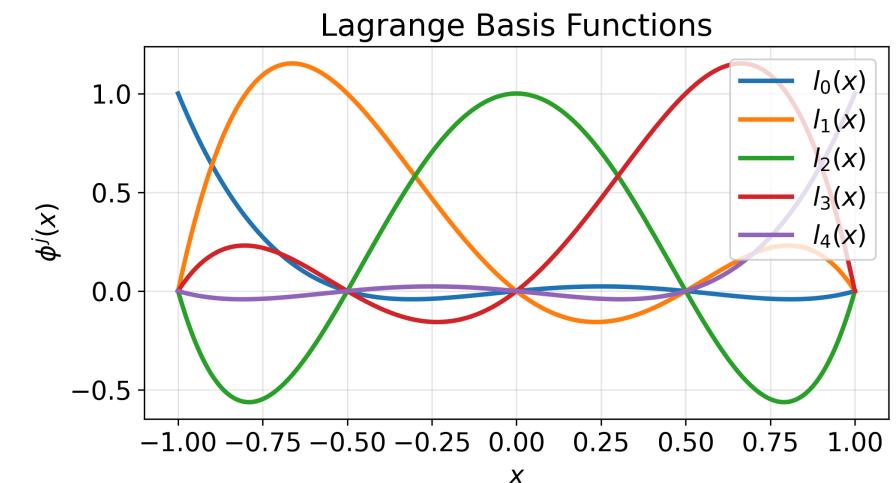
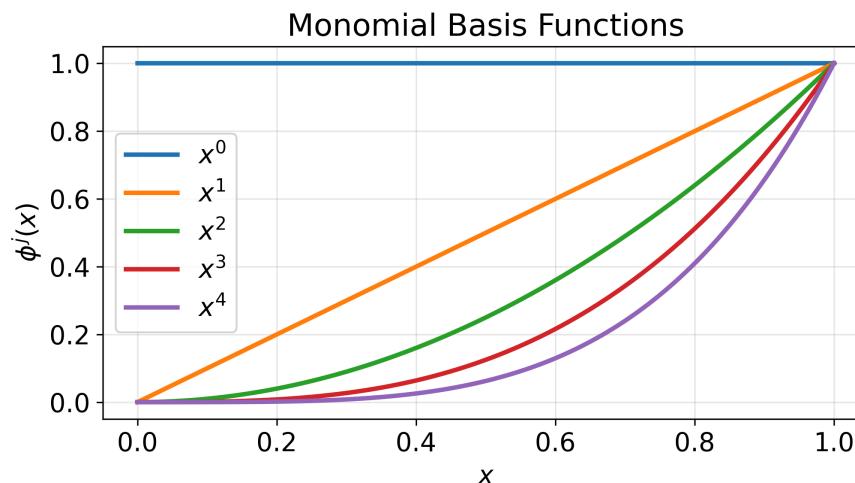
# Interpolants

- Monomial interpolation

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

- Lagrange polynomial interpolation

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



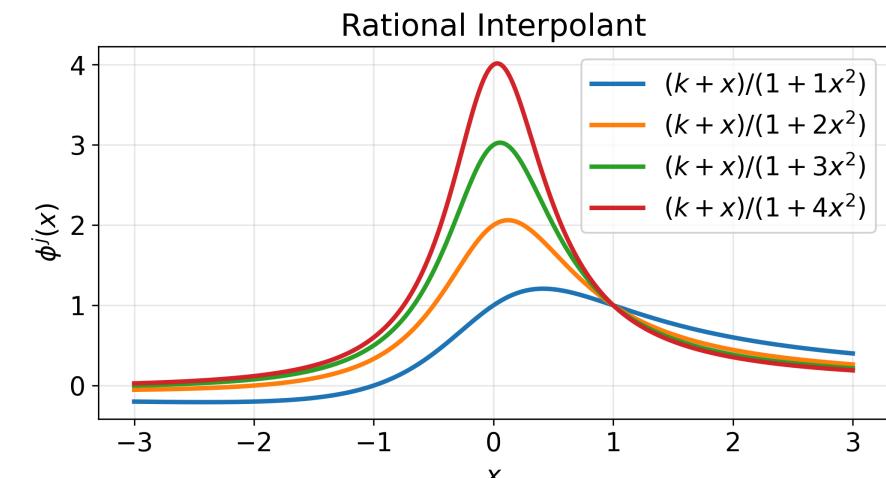
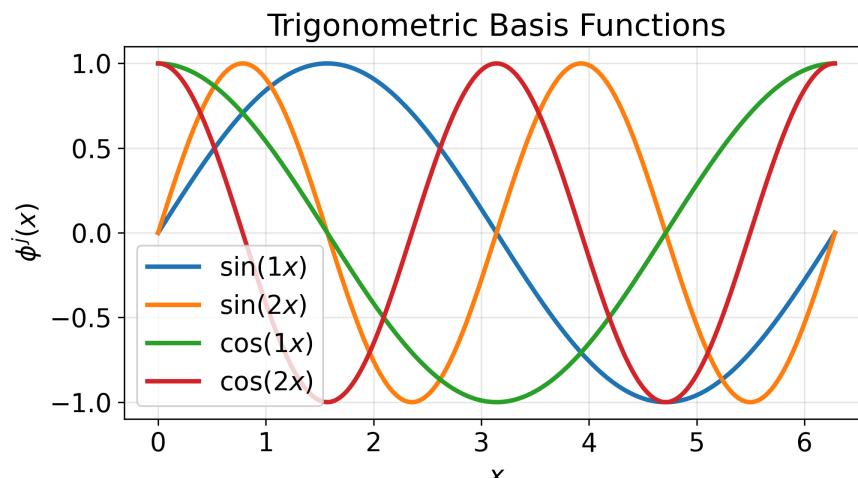
# Interpolants

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

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

- 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

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

## Polynomial interpolation

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

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

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

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

To prove uniqueness, suppose that another interpolating polynomial  $\Psi_m$  of degree  $m \leq n$  exists, such that  $\Psi_m(x_i) = y_i$  for  $i = 0, \dots, n$ . 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  $i = 0, \dots, n$  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  $\text{cond}(A) = \|A^{-1}\| \|A\|$  so that

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

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

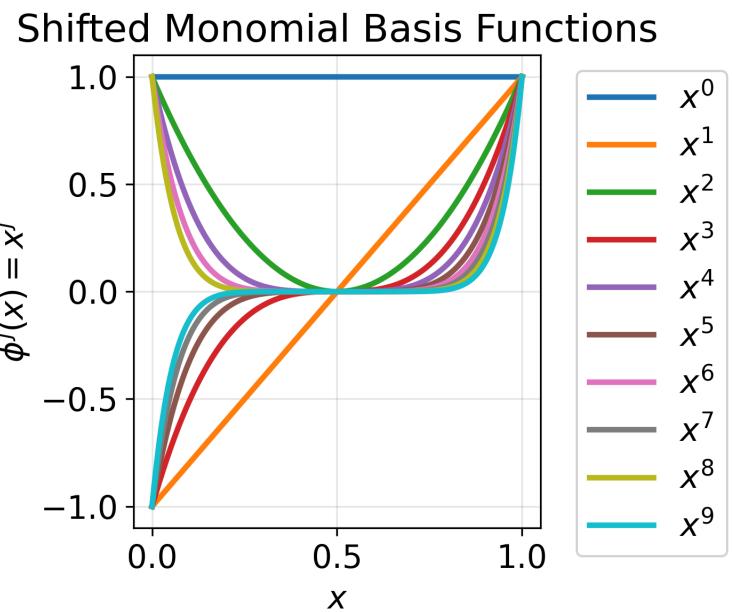
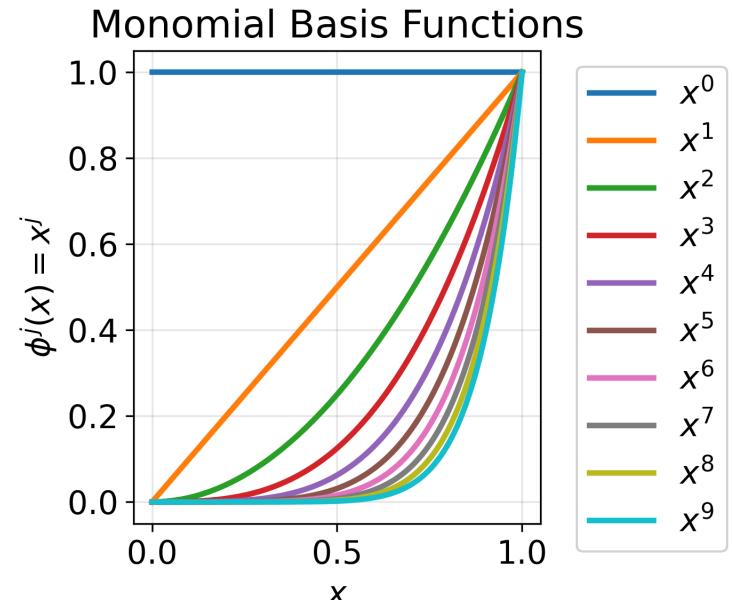
# Monomial basis

- 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 Horner'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:

$$\{\phi_j(x)\}_{j=0}^3 = \{1, x, x^2, x^3\}$$

- The interpolant is of the form be

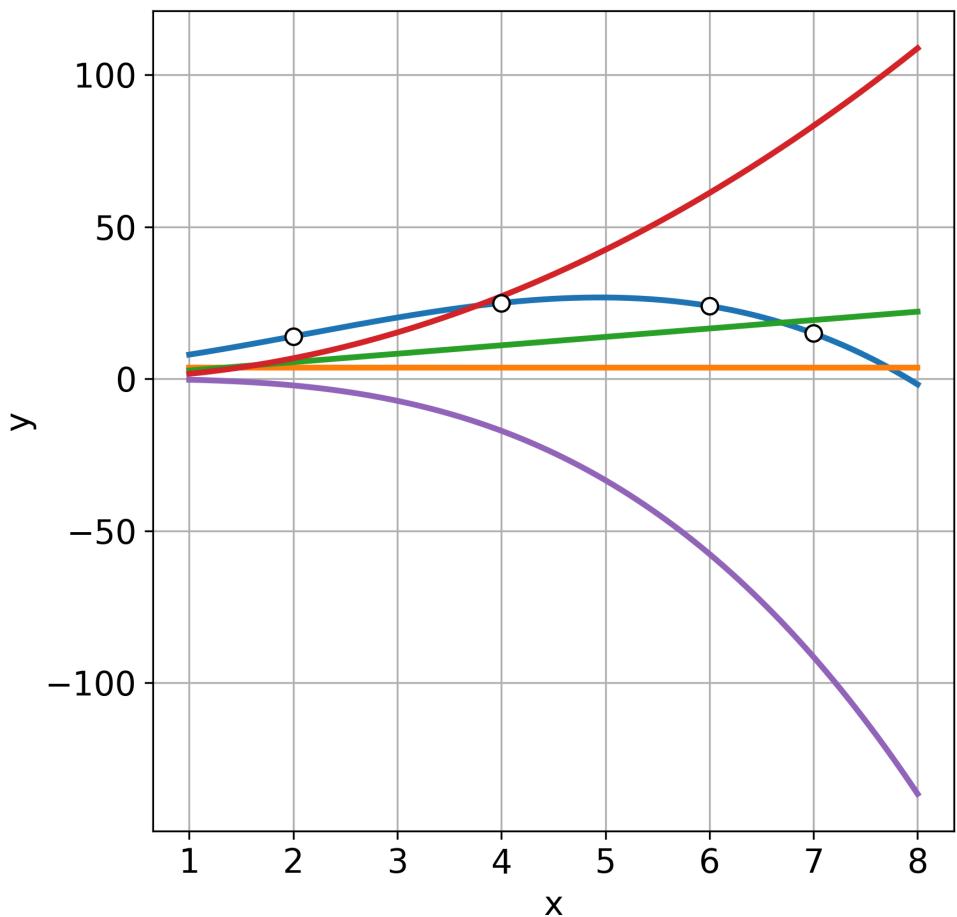
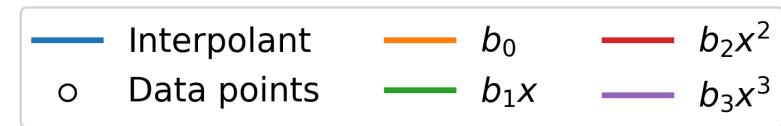
$$\Pi_3(x) = b_0 + b_1x + b_2x^2 + b_3x^3$$

- 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  $\text{cond}(A) \approx 10^3$ ) 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

$$\Pi_n(x) = b_0 + b_1x + \cdots + b_nx^n,$$

requiring  $n$  additions and  $2n - 1$  multiplications for its evaluation.

By exploiting Horner's nested evaluation scheme one can efficiently rewrite the interpolant as

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

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

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

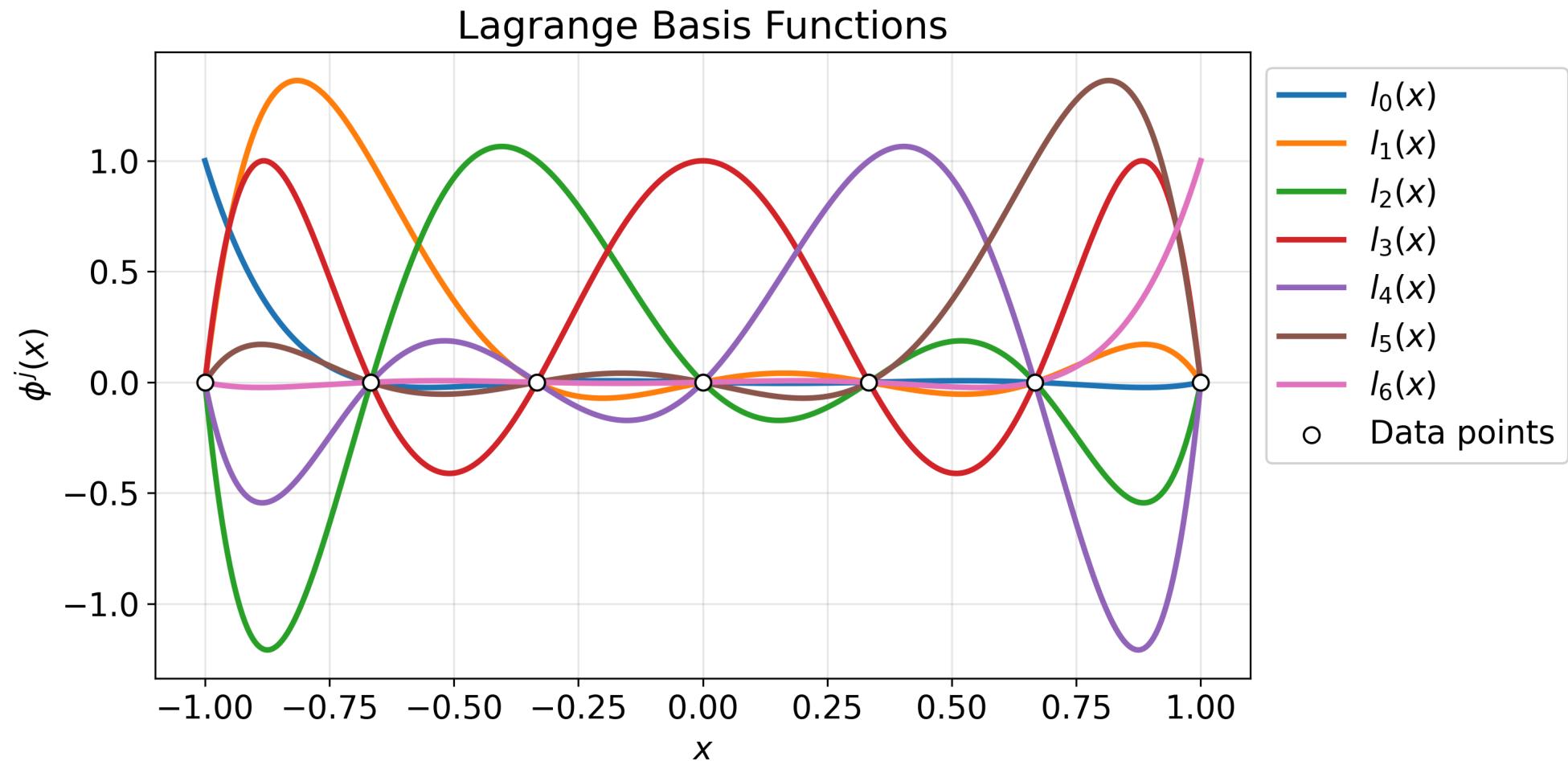
$\rightsquigarrow a_{ij} = l_j(x_i) = \delta_{ij}$ , i.e.  $A = \mathbf{Id}$  (best-conditioned matrix), no system needs to be solved and  $b_j = y_j$ .

- $l_j(x)$  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 polynomials  $\omega_{n+1}$  and barycentric weights  $q_j = \omega'_{n+1}(x_j)$

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

- If new data point  $(x_{n+1}, y_{n+1})$  is added, then the interpolant can be updated in  $\mathcal{O}(n)$  operations:  
 (i) divide each  $q_j$  by  $(x_j - x_{n+1})$ ,  $j = 0, \dots, n$ , and (ii) compute new weight  $q_{n+1}$  as before.

# 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:

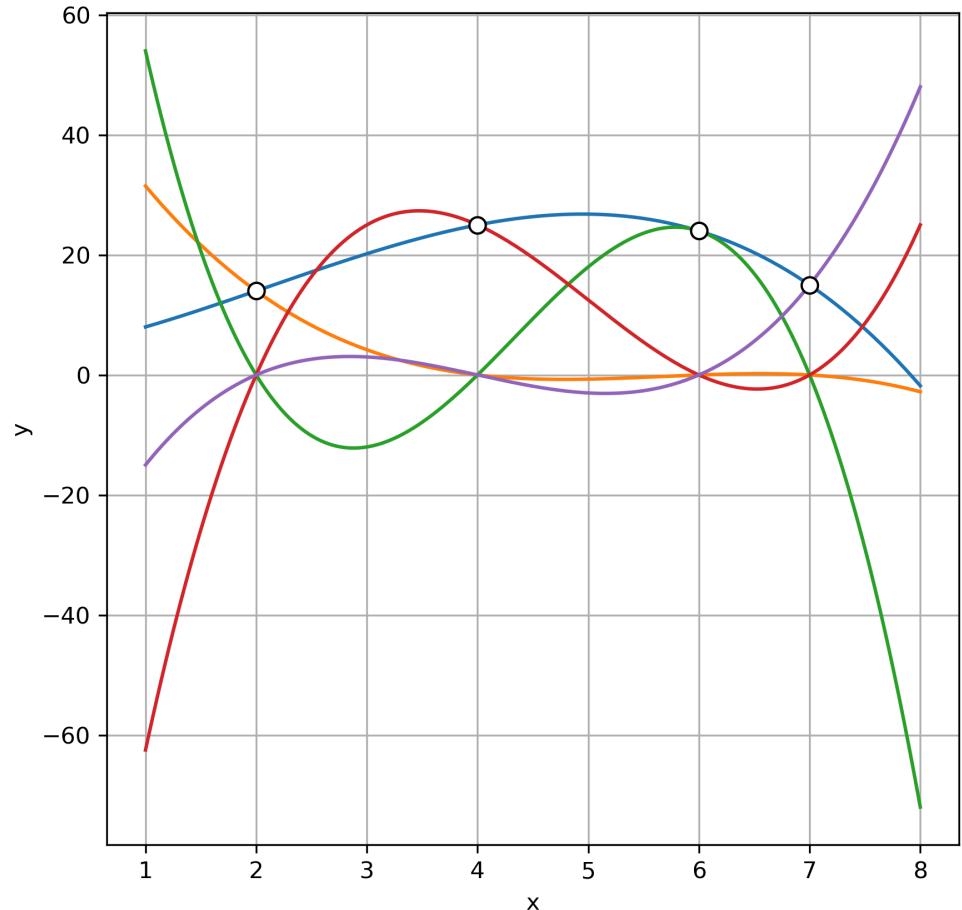
$$l_0(x) = \frac{(x - 6)(x - 4)(x - 7)}{(2 - 6)(2 - 4)(2 - 7)}, \quad l_1(x) = \frac{(x - 2)(x - 4)(x - 7)}{(6 - 2)(6 - 4)(6 - 7)}$$

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

- The interpolant is of the form be

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

<span style="color: blue;">—</span>	Interpolant	<span style="color: orange;">—</span>	$y_0 l_0$	<span style="color: red;">—</span>	$y_2 l_2$
<span style="color: black;">○</span>	Data points	<span style="color: green;">—</span>	$y_1 l_1$	<span style="color: purple;">—</span>	$y_3 l_3$



## 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  $\Pi_n$  (with  $\Pi_n(x_i) = y_i$  for  $i = 0, \dots, n$ ) as the sum of  $\Pi_{n-1}$  (with  $\Pi_{n-1}(x_i) = y_i$  for  $i = 0, \dots, n - 1$ ) 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  $i = 0, \dots, n - 1$  then the polynomial is given by  
 $\rightsquigarrow r_n(x_i) = b_n(x - x_0) \cdots (x - x_{n-1}) = b_n \omega_n(x)$  and the coefficients have to closed form

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

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

## Newton basis

Remarks.

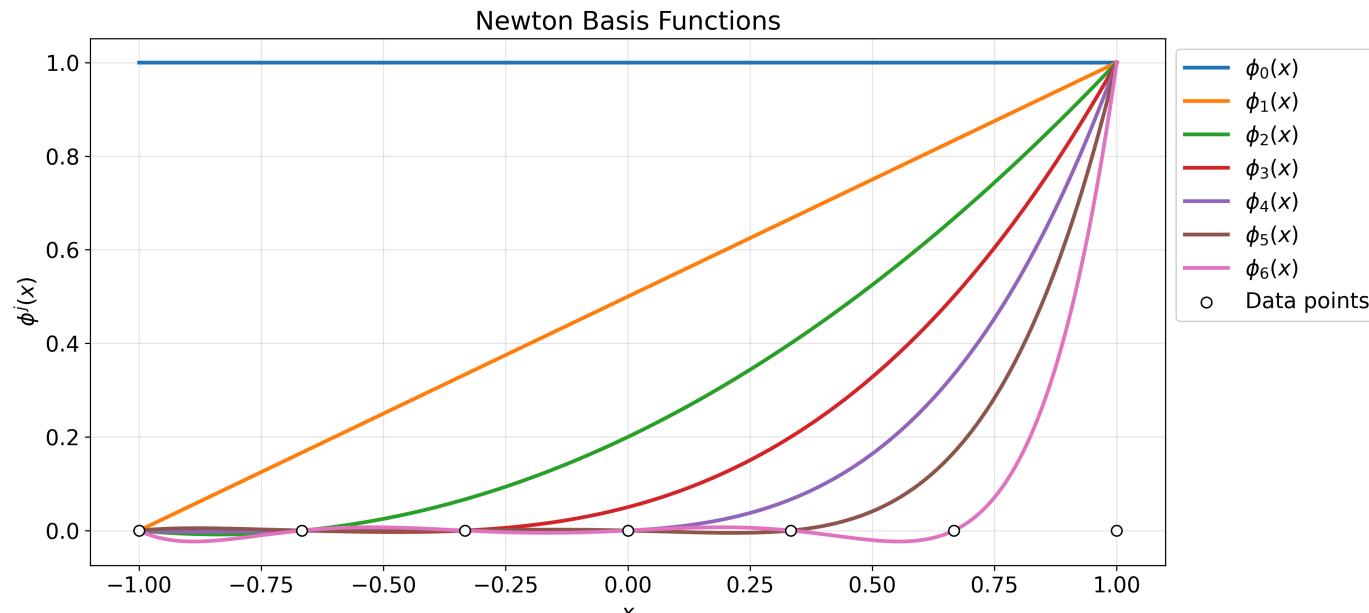
- Uniqueness of the interpolating polynomial ensures that the above expression yields the same interpolating polynomial generated by the Lagrange form.
- Newton basis functions are linearly independent because  $\phi^j(x)$  has exactly degree  $j - 1$ .
- New basis function should not interfere with 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:

$$\phi^0(x) = 1, \quad \phi^1(x) = (x - 2)$$

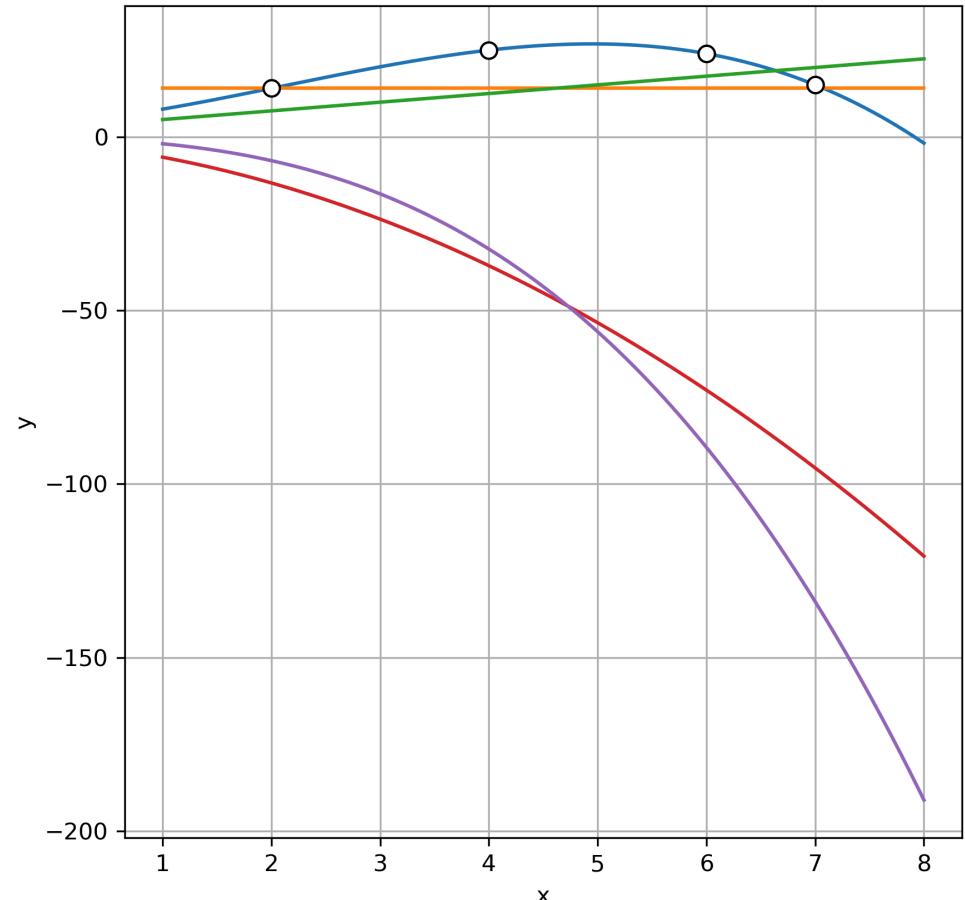
$$\phi^2(x) = (x - 2)(x - 6), \quad \phi^3(x) = (x - 2)(x - 6)(x - 4)$$

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  $\text{cond}(A) = 17$ ) 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  $x_0, x_1, \dots, x_n$ .

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

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

Then

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

where  $\xi \in I$ , with  $I$  being the smallest open interval containing  $x_0, \dots, x_n$  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  $\omega_{n+1}$  is a polynomial, then  $G \in C^{n+1}(I)$  has at least  $n + 2$  zeros in  $I$ :

$\rightsquigarrow$  the  $n + 1$  interpolation points  $x_0, x_1, \dots, x_n$  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  $\xi$ .

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  $\xi$  exists, it is not generally possible to determine its exact value.

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

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

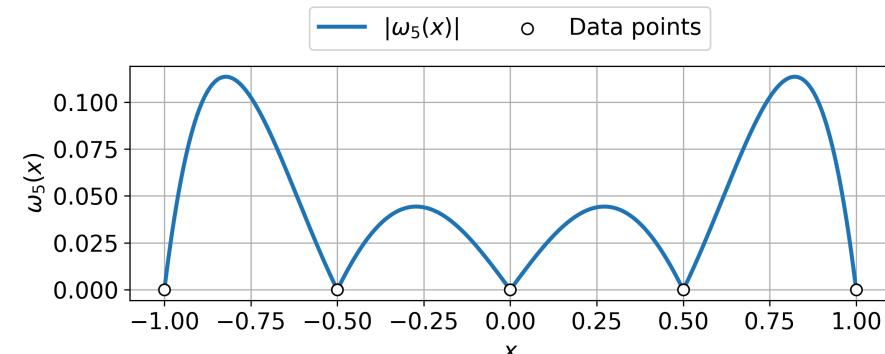
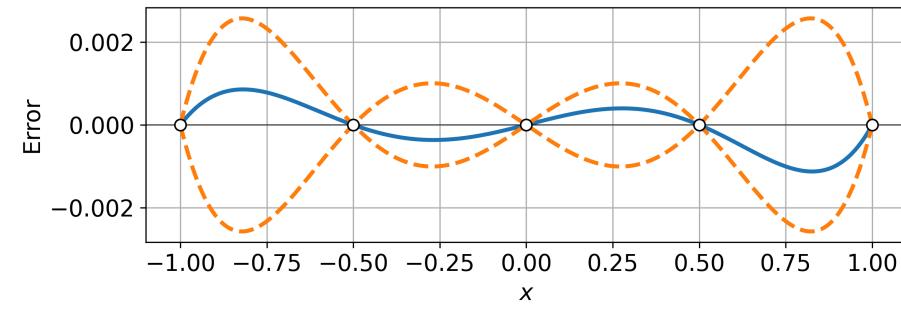
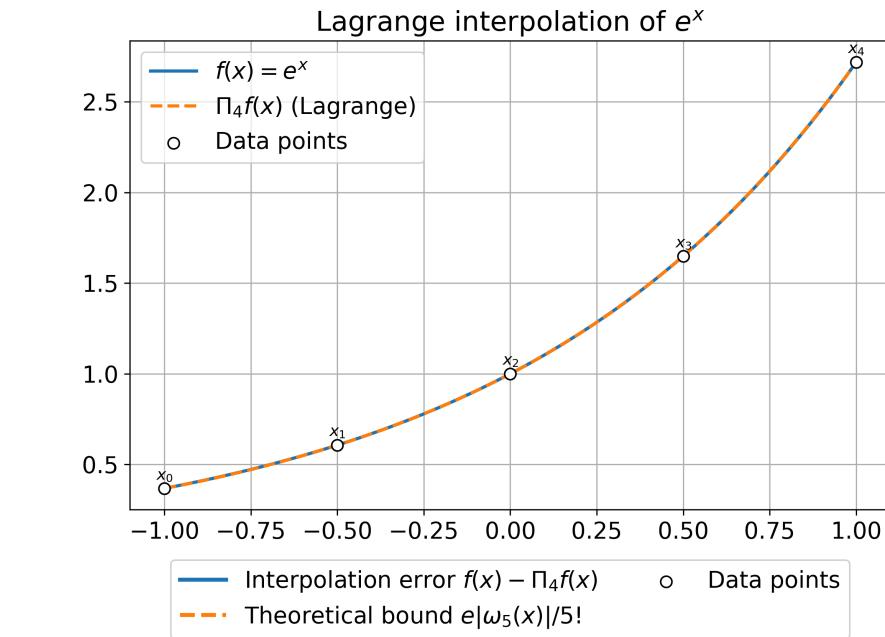
**Example.** Let  $f(x) = \exp(x)$  and  $\Pi_4 f$  interpolates  $f(x)$  at 5 equispaced points  $\{x_i = -1 + \frac{i}{2}\}_{i=0}^4$ .

- $f$  exponential and increasing  $\rightsquigarrow$

$$M = \|f^{(5)}\|_\infty = \max_{x \in [-1, 1]} |\exp(x)| = e$$

- $|\prod_{i=0}^n (x - x_i)| \leq 0.12$

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



## Convergence of polynomial interpolation

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

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

- We cannot deduce that  $\|E_n[f](x)\|_\infty \xrightarrow{n \rightarrow \infty} 0$ , only if  $|f^{(n+1)}(x)|$  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.

$$\|E_n[f]\|_\infty \xrightarrow{n \rightarrow \infty} 0.$$

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

$$\|E_n[f]\|_\infty \xrightarrow{n \rightarrow \infty} 0.$$

## Convergence of polynomial interpolation

We introduce a lower triangular matrix  $X$  of *infinite size*, called the **interpolation matrix** on  $I = [a, b]$   
~~> 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.** 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  $X$ .

~~> 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

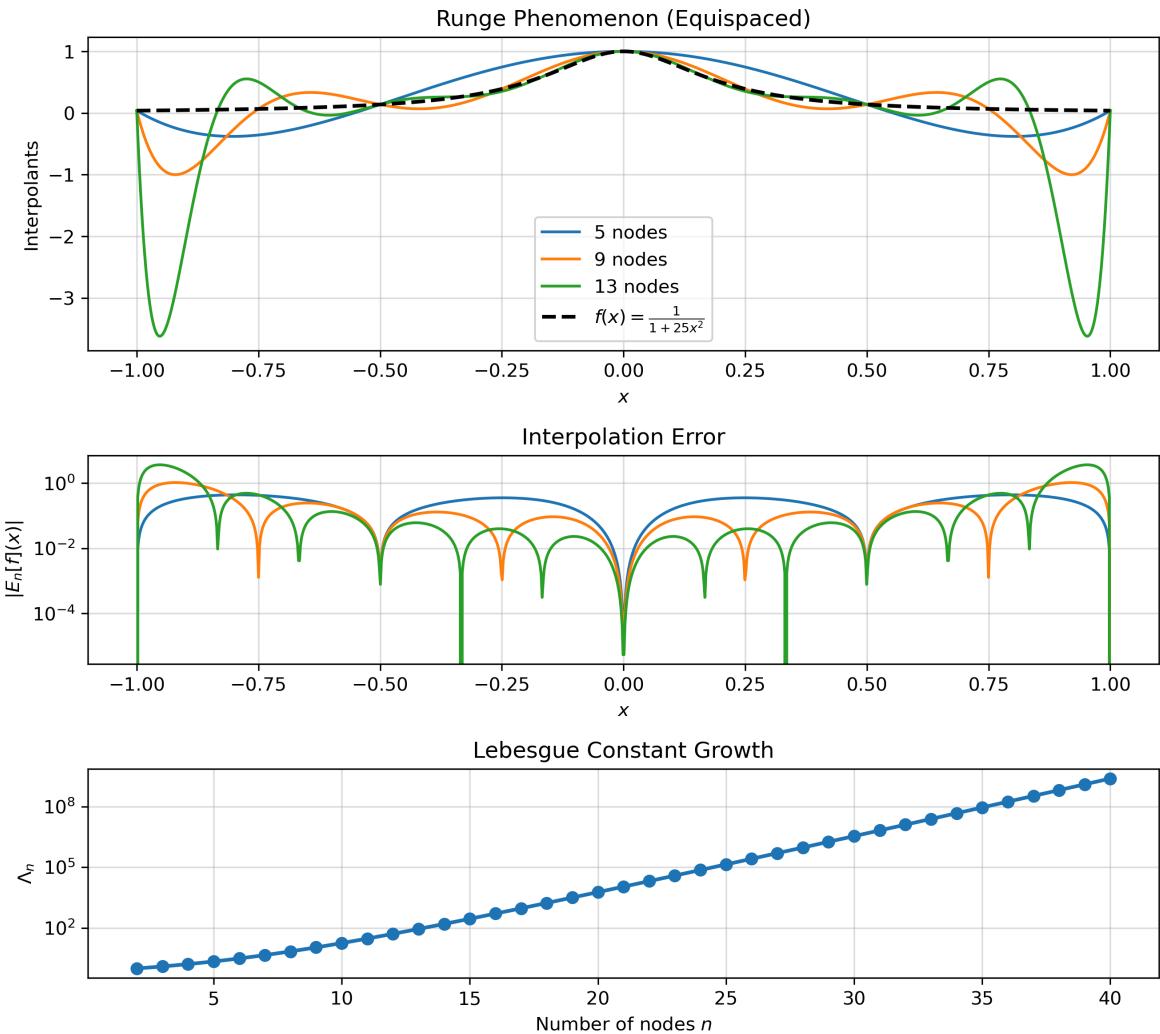
Let us consider the Lagrange interpolation with equispaced nodes  $\{x_i\}_{i=0}^n \in [-1, 1]$  for

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

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

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

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



# 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

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

~ small data perturbations give rise to small changes on the interpolant only for small Lebesgue constant.

## Lebesgue constant

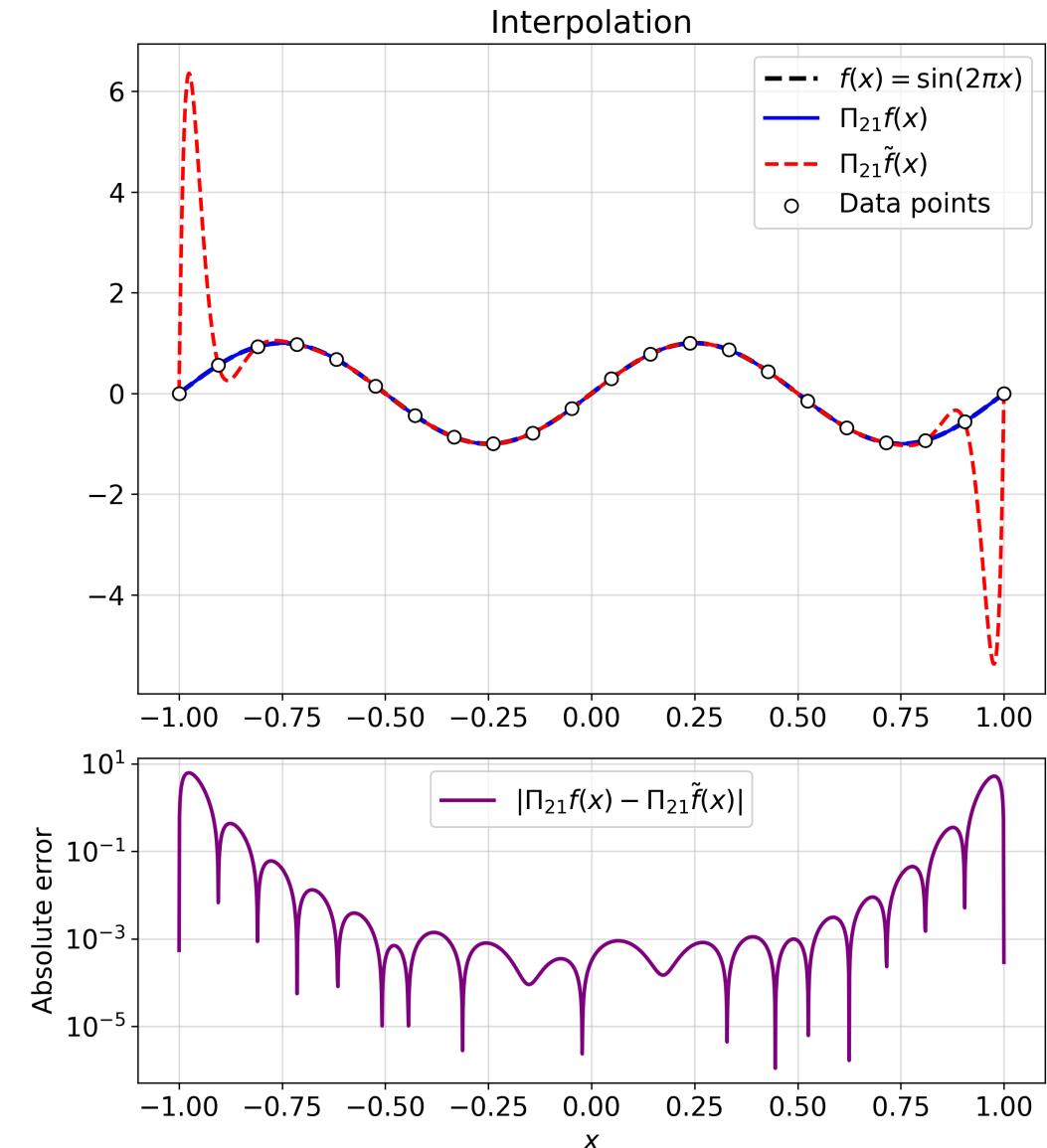
This constant plays the role of the **condition number** for the interpolation problem.

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  $\max_{x_i} |f(x_i) - \tilde{f}(x_i)| = 9.5 \times 10^{-4}$ .



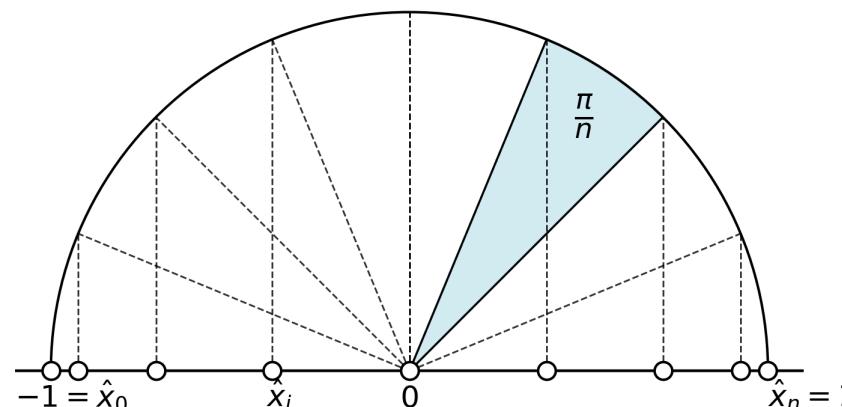
## Chebyshev-Gauss-Lobatto points

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

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

↷ If  $f$  is a continuous and differentiable function in  $I$ , then  $\Pi_n f(x) \xrightarrow{n \rightarrow \infty} f(x)$  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

