Numerical Methods I
MATH-GA 2010.001/CSCI-GA 2420.001

Benjamin Peherstorfer
Courant Institute, NYU

Based on slides by G. Stadler and A. Donev

Interpolation

Today

Last time

- Conjugate gradient method
- Function approximation

Today

- Function approximation
- Interpolation with polynomials
- Interpolation beyond polynomials

Announcements

- Homework 5 is due Mon, Nov 21 before class
- Homework 4, Problem 2b: Using built-in QR is fine; points have been updated.

Recap: Function approximation

Consider a function $f \in \mathcal{V}$ in a function space $\mathcal{V}$. Let now $\phi_1, \ldots, \phi_n$ be a basis of an n-dimensional space $\mathcal{V}_n$.

The task that we are interested in is finding a function $f^* \in \mathcal{V}_n$ that approximates f, i.e.,

$$f^*(x) = \sum_{i=1}^{n} c_i \phi_i(x) \,,$$

with coefficients $c_1, \ldots, c_n$.

If we have an inner product, the best-approximation of f in $\mathcal{V}_n$ w.r.t. the induced norm is given by the projection

$$f^* = \Pi_n f \,,$$

where Π_n is the orthogonal projection onto $\mathcal{V}_n$.

However, in many cases, we cannot directly compute the projection of f onto $\mathcal{V}_n$ because we have "too little knowledge about f" $\rightsquigarrow$ interpolation (/regression)

Recap: Interpolation

Consider n pairs of data samples $(x_i, y_i), i = 1, \ldots, n$ with

$$y_i = f(x_i)$$

Based on $\{(x_i, y_i)\}_{i=1}^n$, we now would like to find an approximation $\tilde{f} \in \mathcal{V}_n$ that is "close" to f.

For example, we could enforce the interpolation condition, namely that it holds

$$\tilde{f}(x_i) = f(x_i)\,, \qquad i = 1, \ldots, n$$

We could also use regression ($m > n$) and minimize, e.g.,

$$\frac{1}{m} \sum_{i=1}^{m} |y_i - \tilde{f}(x_i)|^2$$

Recap: Error decomposition of interpolation

The error of $\tilde{f}$ w.r.t. f can then typically be split into two components (we will formalize this moving forward):

$$\|\tilde{f} - f\| \leq \Lambda(x_1, \ldots, x_n)\|f^* - f\|$$

The projection error $\|f^* - f\|$ describes the best we can do in the space $\mathcal{V}_n$. Even if we had "full knowledge" of f so that we could compute $f^* = \Pi_n f$, we are limited by the space $\mathcal{V}_n$

Intuitively, we'd also expect that the error of $\tilde{f}$ depends on the points $x_1, \ldots, x_n$ at which we have samples of f. This is captured by the "constant" $\Lambda(x_1, \ldots, x_n)$ that is independent of f but depends on $x_1, \ldots, x_n$.

Recap: Polynomial interpolation

Consider $n+1$ pairs $(x_i, y_i), i = 0, \ldots, n$ of a function f with

$$y_i = f(x_i)$$

Recap: Polynomial interpolation

Consider $n+1$ pairs $(x_i, y_i), i = 0, \ldots, n$ of a function f with

$$y_i = f(x_i)$$

Let now $\mathbb{P}_n$ be the set of all polynomials up to degree n over $\mathbb{R}$ so that we have for all $P \in \mathbb{P}_n$

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \,, \qquad a_n, \ldots, a_0 \in \mathbb{R}$$

Recap: Polynomial interpolation

Consider $n+1$ pairs $(x_i, y_i), i = 0, \ldots, n$ of a function f with

$$y_i = f(x_i)$$

Let now $\mathbb{P}_n$ be the set of all polynomials up to degree n over $\mathbb{R}$ so that we have for all $P \in \mathbb{P}_n$

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \,, \qquad a_n, \ldots, a_0 \in \mathbb{R}$$

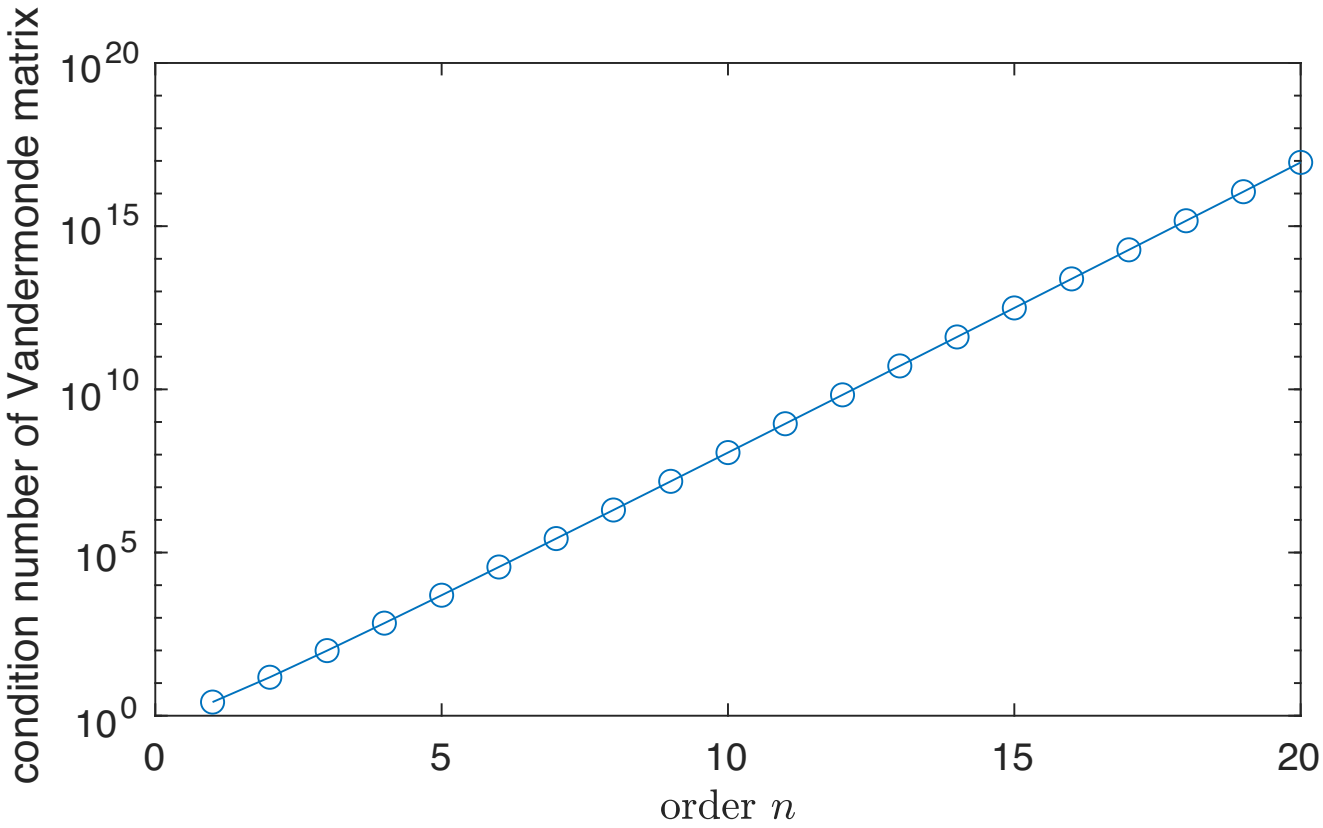
We would like to find a $P \in \mathbb{P}_n$ such that

$$P(x_i) = y_i \,, \qquad i = 0, \ldots, n$$

- The P is what $\tilde{f}$ was on the previous slide
- By saying P is a polynomial of degree n, we fixed the space $\mathcal{V}_{n+1}$ with the notation of the previous slide

Recap: Condition number of Vandermonde matrix

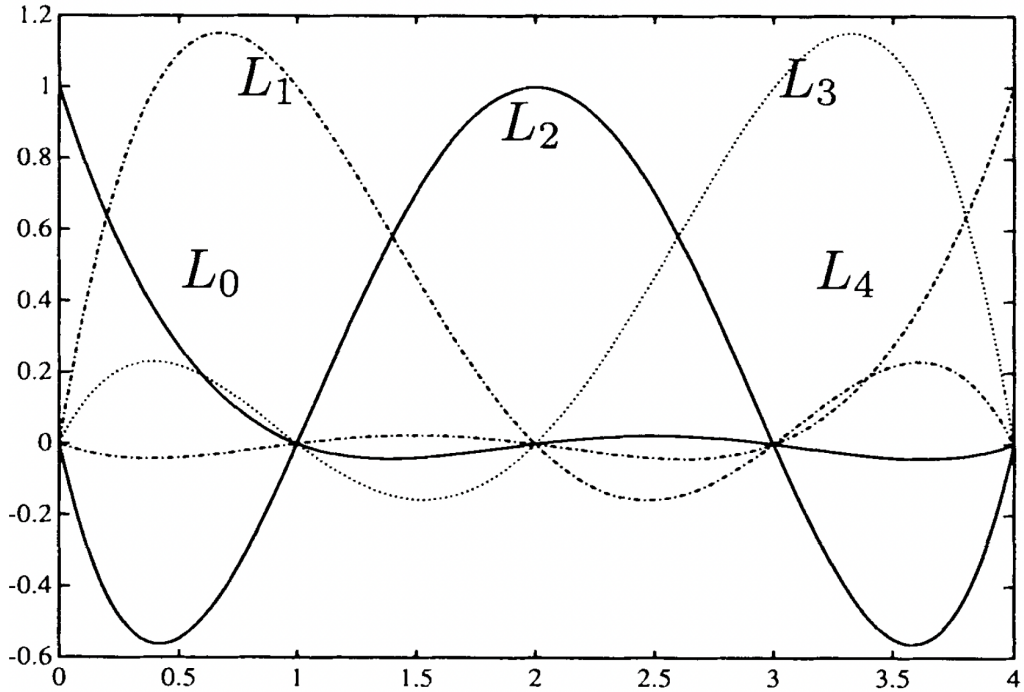
Monomial basis is not a good choice, because the condition number of corresponding system of linear equations blows up



Recap: Lagrange basis

The Lagrange polynomials $L_0, \ldots, L_n \in \mathbb{P}_n$ are uniquely defined for distinct $x_0, \ldots, x_n$

$$L_i(x_j) = \delta_{ij}\,, \qquad L_i \in \mathbb{P}_n\,.$$



Lagrange polynomials up to order $n = 4$ for equidistant $x_0, \ldots, x_4$. [Figure: Deuflhard]

The corresponding explicit formula is

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}, \qquad i = 0, \ldots, n$$

What are the coefficients $a_n, \ldots, a_0$ so that $P(x_i) = y_i$ for $i = 0, \ldots, n$?

$$P(x) = \sum_{i=0}^{n} y_i L_i(x)$$

because

$$P(x_j) = \sum_{i=0}^{n} y_i L_i(x_j) = \sum_{i=0}^{n} y_i \delta_{ij} = y_j$$

If we have the basis $L_0, \ldots, L_n$, we obtain the polynomial P for free but the cost of evaluating the polynomial is too high for practical computations

The Lagrange polynomials are orthogonal w.r.t. the following inner product over $\mathbb{P}_n$

$$\langle P, Q \rangle = \sum_{i=0}^{n} P(x_i) Q(x_i) \,, \qquad P, Q \in \mathbb{P}_n$$

Let's try to generalize this to other scalar products to find other orthogonal bases

Recap: Orthogonal polynomials

Define an inner product between functions:

$$(f, g) = \int_a^b \omega(x) f(x) g(x)\, dx,$$

where $\omega(x) > 0$ for $a \leq x \leq b$ is a weight function. The induced norm is $\|f\| := \sqrt{(f, f)}$.

Let $P_0, P_1, P_2, \ldots, P_K$ be polynomials of $0, 1, 2, \ldots, K$ order, respectively. They are called orthogonal polynomials on $[a, b]$ with respect to the weight function $\omega(x)$ if it holds

$$(P_i, P_j) = \int_a^b \omega(x) P_i(x) P_j(x) \mathrm{d}x = \delta_{ij} \gamma_i \,, \qquad i, j = 0, \ldots, K \,,$$

with $\gamma_i = \|P_i\|^2 > 0$.

To define orthogonal polynomials uniquely, we require the leading coefficient to be one, i.e.,

$$P_k(x) = x^k + \dots$$

Theorem: There exist uniquely determined orthogonal polynomials $P_k \in \mathbb{P}_k$ with leading coefficient 1. These polynomials satisfy the 3-term recurrence relation:

$$P_k(x) = (x + a_k)P_{k-1}(x) + b_k P_{k-2}(x), \quad k = 2, 3, \dots$$

with starting values $P_0 = 1$, $P_1 = x + a_1$, where

$$a_k = -\frac{(xP_{k-1}, P_{k-1})}{(P_{k-1}, P_{k-1})}, \quad b_k = -\frac{(P_{k-1}, P_{k-1})}{(P_{k-2}, P_{k-2})}$$

Proof: $\rightsquigarrow$ board

3-term recurrence:

$P_0 = 1, \quad P_1 = x + a_1$

$(P_0, P_1) = 0$

Suppose $P_0, \dots, P_{k-1}$

↳ P_j has degree j

↳ P_i, P_j orthogonal $i \neq j$

↳ leading coeff. is 1

$P_k \in \mathbb{P}_k$ degree k, normalized

$$\underbrace{P_k - x P_{k-1}}_{x^k + \dots \quad x(x^{k-1} + \dots)} \Big\} \text{ is degree } \leq k-1$$

$\mathbb{P}_{k-1}$ basis: $P_0, \dots, P_{k-1}$

$$P_k - x P_{k-1} = \sum_{j=0}^{k-1} c_j P_j$$

$$c_j = \frac{(P_k - x P_{k-1}, P_j)}{(P_j, P_j)}$$

If P_k orthogonal to $P_0, \dots, P_{k-1}$, then

$$(P_k, P_0) = 0 \qquad j = 0, \dots, k-1$$

$$c_j = -\frac{(x P_{k-1}, P_j)}{(P_j, P_j)} = -\frac{(P_{k-1}, x P_j)}{(P_j, P_j)}$$

$$c_{k-1} = -\frac{(x P_{k-1}, P_{k-1})}{(P_{k-1}, P_{k-1})}$$

$$c_{k-2} = -\frac{(P_{k-1}, x P_{k-2})}{(P_{k-2}, P_{k-2})}$$

$$P_{k-1} = (x + \alpha_{k-1}) P_{k-2} + \beta_{k-1} P_{k-3}$$

$$x P_{k-2} = P_{k-1} - \alpha_{k-1} P_{k-2} - \beta_{k-1} P_{k-3}$$

$$= -\frac{(P_{k-1}, P_{k-1} - \alpha_{k-1} P_{k-2} - \beta_{k-1} P_{k-3})}{(P_{k-2}, P_{k-2})}$$

orth.

$$= -\frac{(P_{k-1}, P_{k-1})}{(P_{k-2}, P_{k-2})}$$

Similar:

$$c_{k-3} = \dots = c_0 = 0$$

$$\Rightarrow P_k - x P_{k-1} = c_{k-1} P_{k-1} + c_{k-2} P_{k-2}$$

$$\Rightarrow \quad P_k = (x + \underbrace{c_{k-1}}_{a_k}) P_{k-1} + \underbrace{c_{k-2}}_{b_k} P_{k-2}$$

Numerics with 3-term recurrence

- 3-term recurrence can be used to compute polynomials P_k completely, or
- evaluate P_k at a point x_0 via pre-computed the a's and b's
- However, simple application of 3-term recurrence might not always be stable due to cancellation (coefficients a_k, b_k can be negative)
- Cancellation errors can be avoided with clever numerics (Sec 6.3 in Deuflhard)

Chebyshev polynomials

Chebyshev polynomials for $-1 \leq x \leq 1$ are given by the 3-term recurrence: $T_0(x) = 1$, $T_1(x) = x$,

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \quad \text{for } k \geq 2.$$

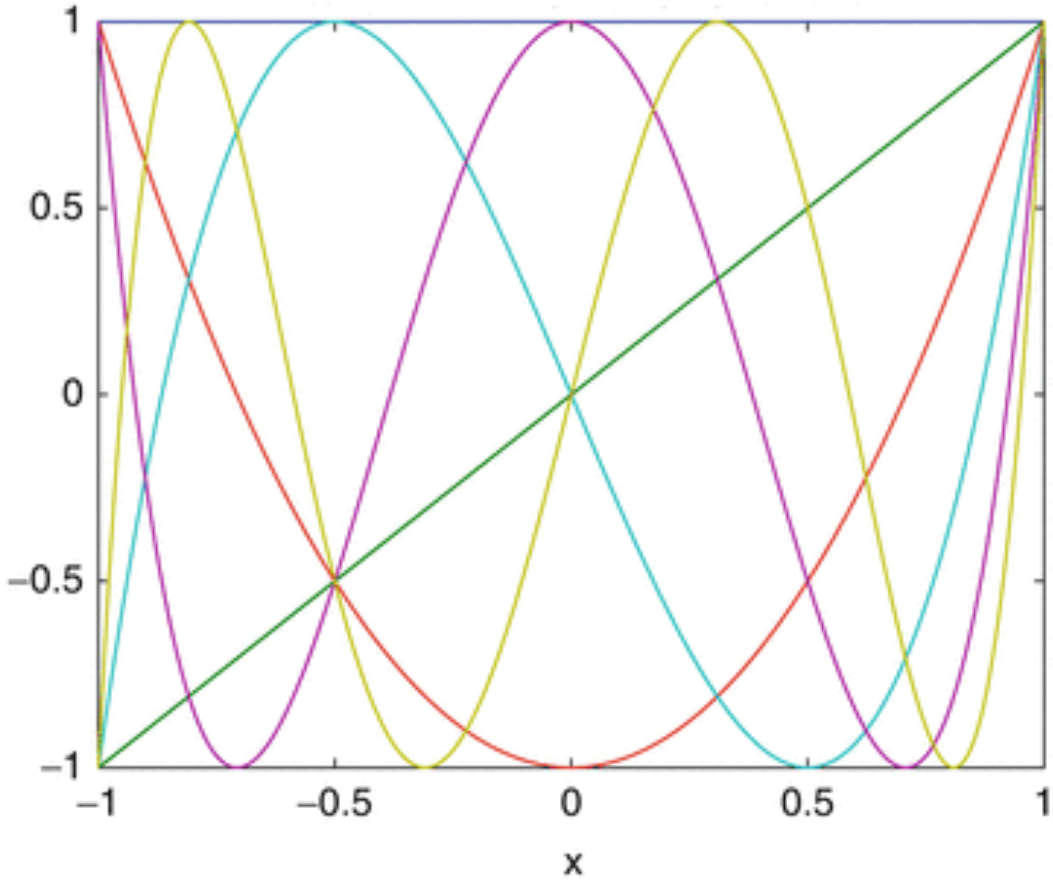
Chebyshev polynomials are orthogonal in the following inner product

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} T_n(x) T_m(x)\, dx = \begin{cases} 0 & n \neq m \\ \pi & n = m = 0 \\ \pi/2 & n = m \neq 0 \end{cases}$$

The roots of Chebyshev polynomials play an important role in interpolation and numerical quadrature.

Chebyshev polynomials are also given by $T_k(x) = \cos(k \arccos(x))$ with roots

$$x_i = \cos\left(\frac{\pi(i - 1/2)}{k}\right)$$



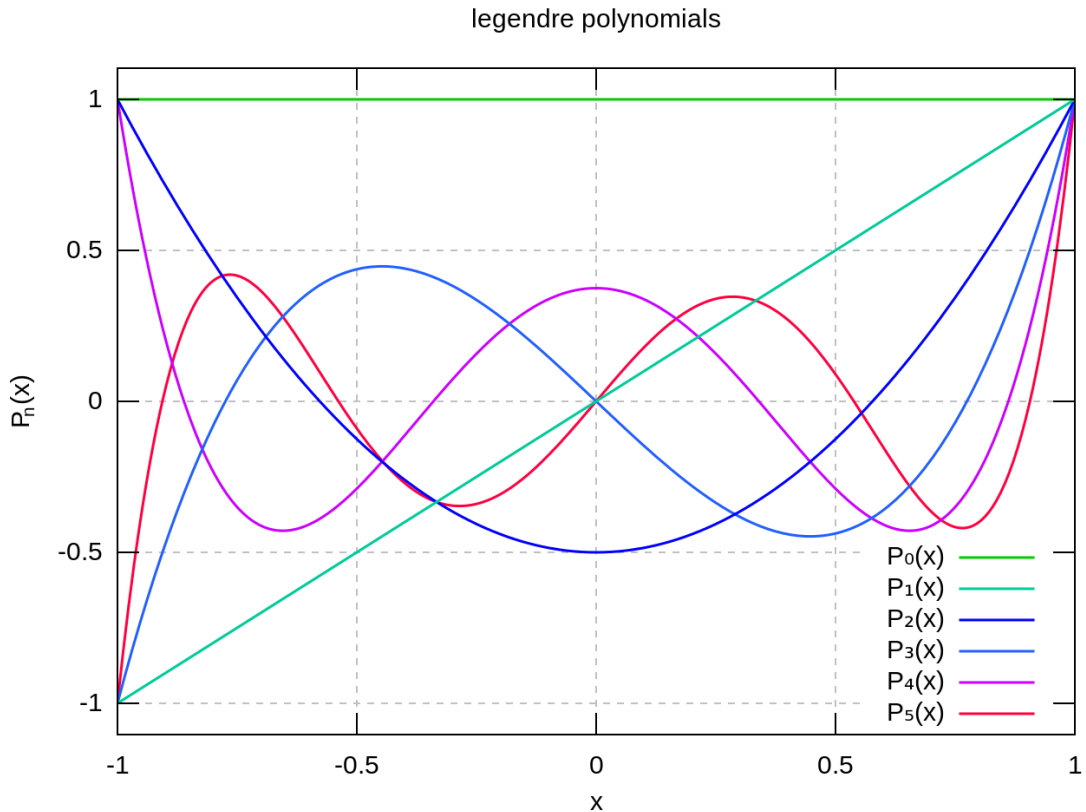
Source: Springer, Encyclopedia of Applied and Computational Mathematics.

Question: What property of the roots of the polynomials do you observe?

Legendre polynomials

Orthogonal polynomials for weight function $\omega \equiv 1$, satisfy $L_0 = 1$, $L_1 = x$, and

$$L_{k+1}(x) = \frac{2k+1}{k+1} x L_k(x) - \frac{k}{k+1} L_{k-1(x)}$$



Approximation error of polynomial interpolation

Let $f : I \to \mathbb{R}$ and let $x_0, x_1, \ldots, x_n \in I$ be $n+1$ distinct nodes. Assume $f \in C^{n+1}(I)$. Then the interpolation error at point $x \in I$ is

$$E_n(x) = f(x) - P_f(x|x_0, \ldots, x_n) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x)\,,$$

where $\xi \in I$ and

$$\omega_{n+1}(x) = \prod_{i=0}^{n}(x - x_i)\,,$$

is the nodal polynomial.

Proof: $\rightsquigarrow$ textbook

What are observations can we make?

Approximation error of polynomial interpolation

Let $f : I \to \mathbb{R}$ and let $x_0, x_1, \ldots, x_n \in I$ be $n+1$ distinct nodes. Assume $f \in C^{n+1}(I)$. Then the interpolation error at point $x \in I$ is

$$E_n(x) = f(x) - P_f(x|x_0, \ldots, x_n) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x)\,,$$

where $\xi \in I$ and

$$\omega_{n+1}(x) = \prod_{i=0}^{n}(x - x_i)\,,$$

is the nodal polynomial.

Proof: $\rightsquigarrow$ textbook

What are observations can we make?

- The error bound requires increasing smoothness with the degree n $\rightsquigarrow$ can we derive bounds for $f \in C^k$ with k fixed independent of degree n?
- The bound critically depends on ω_{n+1} $\rightsquigarrow$ contains information about the nodes

Let's say the function f is continuous, i.e., $f \in C^0([a,b])$, but not even differentiable.

Define the maximum norm

$$\|f\|_\infty = \max_{x\in[a,b]} |f(x)|$$

Given points $X = \{x_0, \ldots, x_n\} \subset [a,b]$, define the interpolation error

$$E_{n,\infty}(X) = \|f - P_f(\cdot|X)\|_\infty$$

and the best-approximation error with $f^* \in \mathbb{P}_n$

$$E_n^* = \|f - f^*\|_\infty \leq \|f - \tilde{f}\|_\infty\,, \qquad \forall \tilde{f} \in \mathbb{P}_n$$

Side remark: What does the Stone-Weierstrass theorem tell us about the best-approximation of continuous functions with polynomials?

Let's say the function f is continuous, i.e., $f \in C^0([a,b])$, but not even differentiable.

Define the maximum norm

$$\|f\|_\infty = \max_{x\in[a,b]} |f(x)|$$

Given points $X = \{x_0, \ldots, x_n\} \subset [a,b]$, define the interpolation error

$$E_{n,\infty}(X) = \|f - P_f(\cdot|X)\|_\infty$$

and the best-approximation error with $f^* \in \mathbb{P}_n$

$$E_n^* = \|f - f^*\|_\infty \leq \|f - \tilde{f}\|_\infty\,, \qquad \forall \tilde{f} \in \mathbb{P}_n$$

Side remark: What does the Stone-Weierstrass theorem tell us about the best-approximation of continuous functions with polynomials? Universal approximation: Uniform convergence (i.e., converge in $\|\cdot\|_\infty$). However, the sequence of polynomials that converges is not necessarily obtained via interpolation.

Theorem Let $f \in C^0([a,b])$ and $X = \{x_0, \ldots, x_n\} \subset [a,b]$. Then

$$E_{n,\infty}(X) = E_n^*(1 + \Lambda_n(X)),$$

where $\Lambda_n(X)$ is the *Lebesgue* constant of X, defined as

$$\Lambda_n(X) = \left\| \sum_{j=0}^{n} |L_j^{(n)}| \right\|_\infty ,$$

where $L_j^{(n)} \in \mathbb{P}_n$ is the j-th Lagrange polynomial with

$$L_j^{(n)}(x_i) = \begin{cases} 1, & i = j, \\ 0, & \text{otherwise} \end{cases}$$

Notice the decomposition of the error into component E_n^* (independent of X but dependent on f) and $\Lambda_n(X)$ (independent of f but dependent on X)

We control the best-approximation error E_n^* with the space $\mathbb{P}_n$

We control the Lebesgue constant $\Lambda_n(X)$ with the nodes $x_0, \ldots, x_n$

What are two important questions to ask about the Lebesgue constant $\Lambda_n(X)$?

*Not necessarily nested

We control the best-approximation error E_n^* with the space $\mathbb{P}_n$

We control the Lebesgue constant $\Lambda_n(X)$ with the nodes $x_0, \ldots, x_n$

What are two important questions to ask about the Lebesgue constant $\Lambda_n(X)$?

- What are "good" nodes $x_0, \ldots, x_n$ that keep $\Lambda_n(X)$ low or even minimize it?
- In the best case, how does $\Lambda_n(X)$ behave with respect to $n \to \infty$

*Not necessarily nested

We control the best-approximation error E_n^* with the space $\mathbb{P}_n$

We control the Lebesgue constant $\Lambda_n(X)$ with the nodes $x_0, \ldots, x_n$

What are two important questions to ask about the Lebesgue constant $\Lambda_n(X)$?

- What are "good" nodes $x_0, \ldots, x_n$ that keep $\Lambda_n(X)$ low or even minimize it?
- In the best case, how does $\Lambda_n(X)$ behave with respect to $n \to \infty$

Famous result by Faber (1914): Given a sequence of any nodes*
$X_n = \{x_{n,0}, x_{n,1}, \ldots, x_{n,n}\} \subset [a, b]$, then there always exists a continuous function f so that $P_f(\cdot|x_0, \ldots, x_n)$ does not converge in $\|\cdot\|_\infty$ to f for $n \to \infty$

Thus, polynomial interpolation does not allow for approximating any continuous function

*Not necessarily nested

We control the best-approximation error E_n^* with the space $\mathbb{P}_n$

We control the Lebesgue constant $\Lambda_n(X)$ with the nodes $x_0, \ldots, x_n$

What are two important questions to ask about the Lebesgue constant $\Lambda_n(X)$?

- What are "good" nodes $x_0, \ldots, x_n$ that keep $\Lambda_n(X)$ low or even minimize it?
- In the best case, how does $\Lambda_n(X)$ behave with respect to $n \to \infty$

Famous result by Faber (1914): Given a sequence of any nodes*
$X_n = \{x_{n,0}, x_{n,1}, \ldots, x_{n,n}\} \subset [a, b]$, then there always exists a continuous function f so that $P_f(\cdot|x_0, \ldots, x_n)$ does not converge in $\|\cdot\|_\infty$ to f for $n \to \infty$

Thus, polynomial interpolation does not allow for approximating any continuous function

However, interpolation works fantastic for "most" functions $\rightsquigarrow$ "Six myths of polynomial interpolation and quadrature," Trefethen, Approximation Theory and Approximation Practice and also chebfun.org

*Not necessarily nested

It also has been shown that for any possible choice $X_n = \{x_{n,0}, \ldots, x_{n,n}\}$, there exists a constant $C > 0$ such that

$$\Lambda_n(X_n) > \frac{2}{\pi} \log(n+1) - C\,, \qquad n = 0, 1, \ldots$$

Thus $\Lambda_n(X_n) \to \infty$ for $n \to \infty$

- This is an instability statement in the sense that "investing more time" (by selecting more grid points), leads to a larger Lebesgue constant
- However, the Lebesgue constant might grow very slowly with n (even slower than E_n^* decreases!)

The Lebesgue constant is also the absolute condition number of polynomial interpolation on $[a, b]$ with points X

$$\kappa_{\text{abs}} = \Lambda_n(X)$$

$\rightsquigarrow$ textbook by Deuflhard

Let's approximate

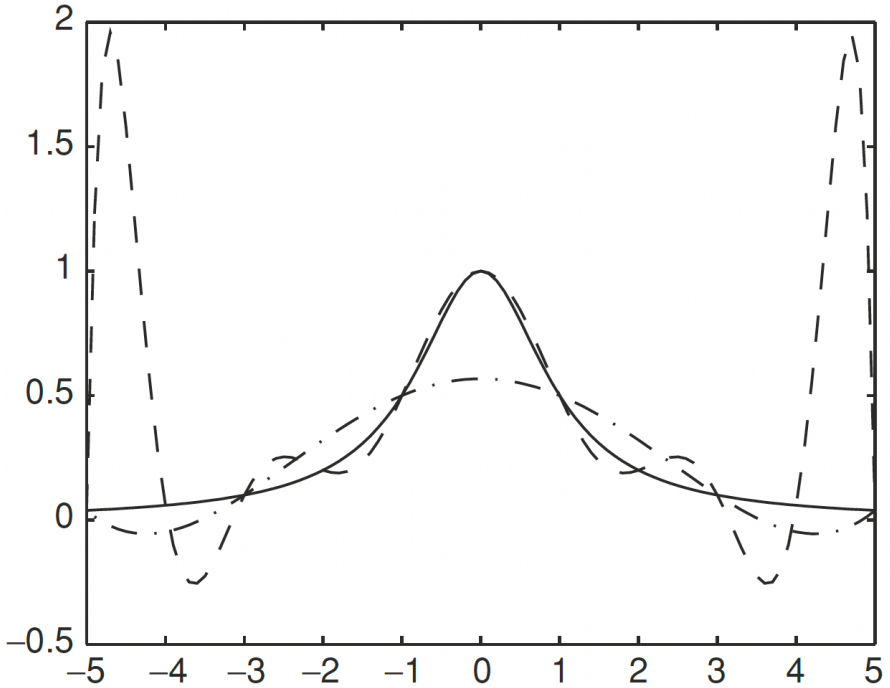
$$f(x) = \frac{1}{1+x^2}\,, \qquad -5 \le x \le 5\,,$$

using polynomial interpolation on equally spaced nodes in $[-5, 5]$.

Let's approximate

$$f(x) = \frac{1}{1+x^2}\,, \qquad -5 \le x \le 5\,,$$

using polynomial interpolation on equally spaced nodes in $[-5, 5]$.

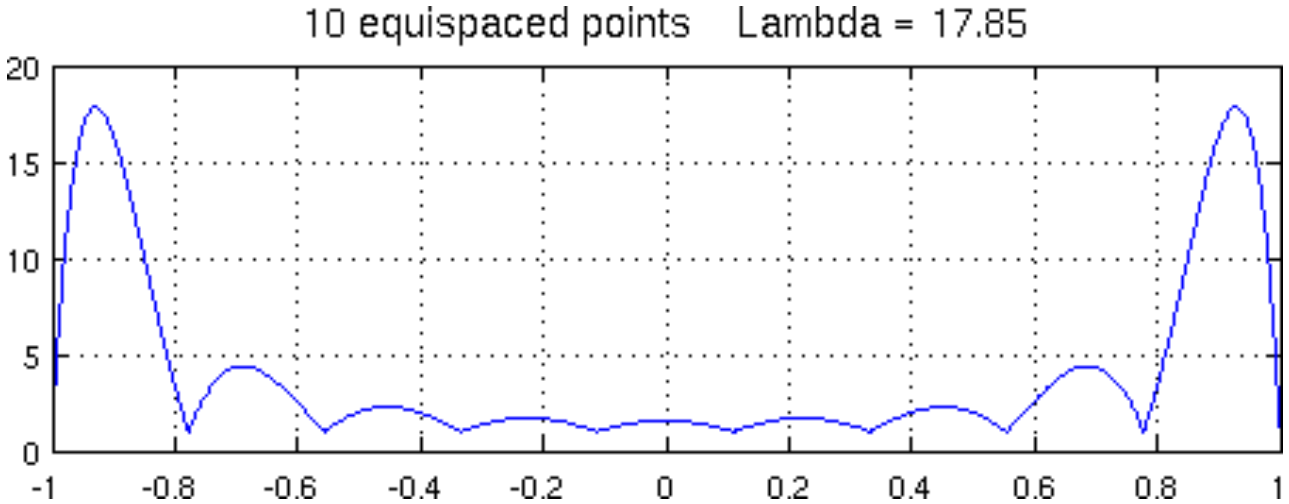


[Figure: Quarteroni]

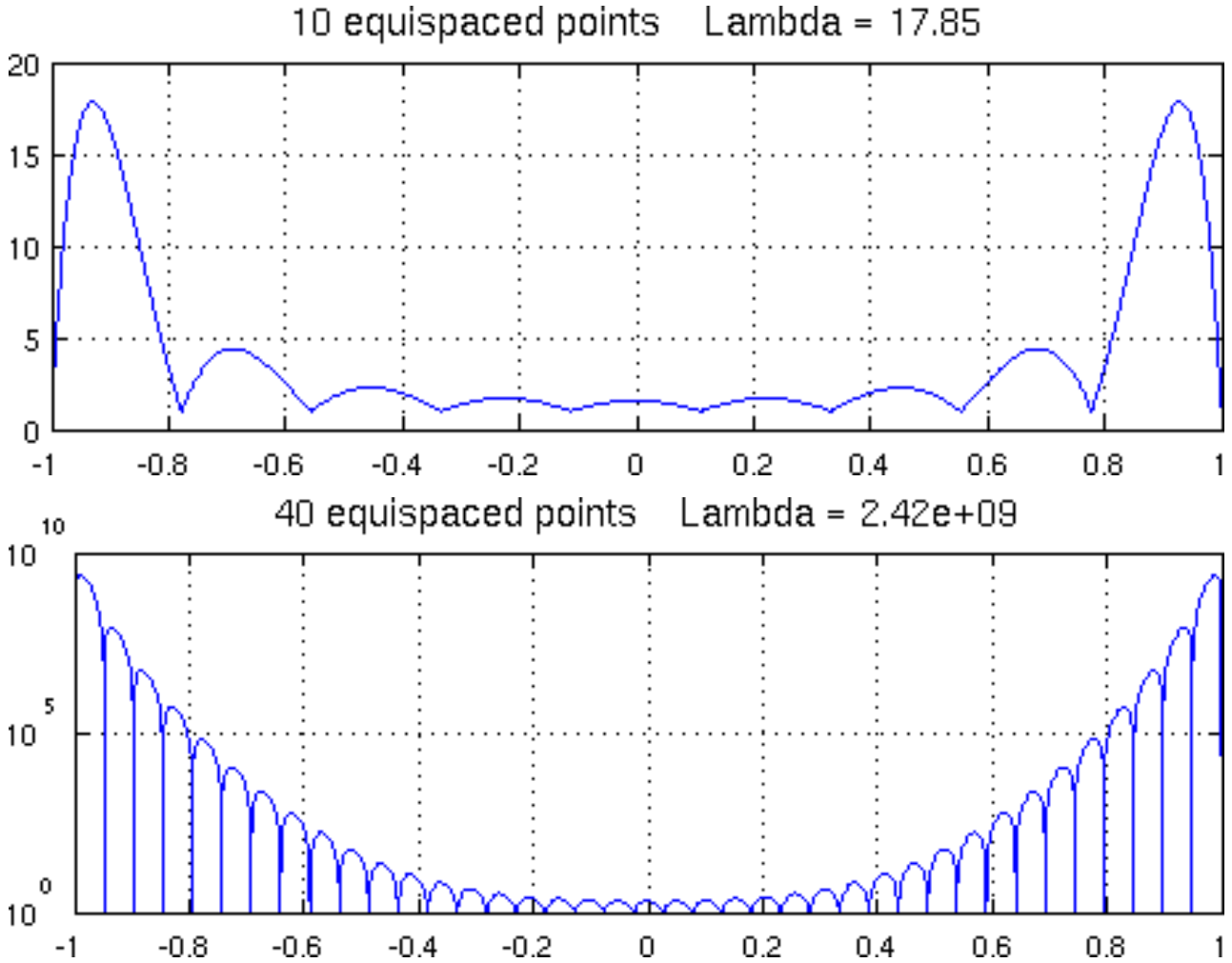
Interpolants of degree $n = 5$ and $n = 10$ of $f(x) = 1/(1+x^2)$ on equidistant nodes. It can be shown that polynomial interpolation does not converge for $|x| > 4$ for this f

It is a very common situation that interpolation on equidistant nodes leads to high oscillations near the interval ends $\rightsquigarrow$ Runge's phenomenon

For equidistant nodes, the Lebesgue constant grows dramatically near the interval ends

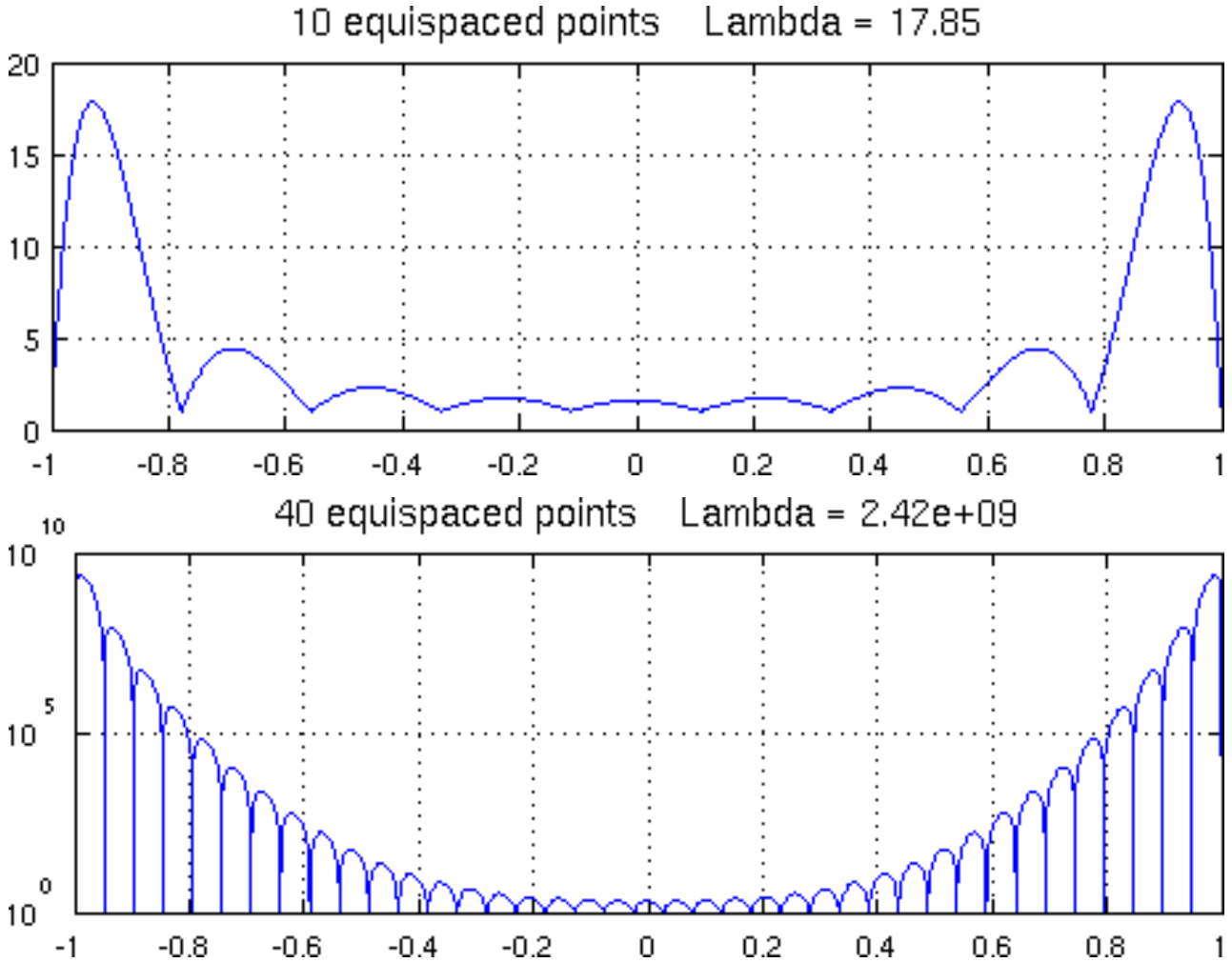


For equidistant nodes, the Lebesgue constant grows dramatically near the interval ends



What is a way out of this?

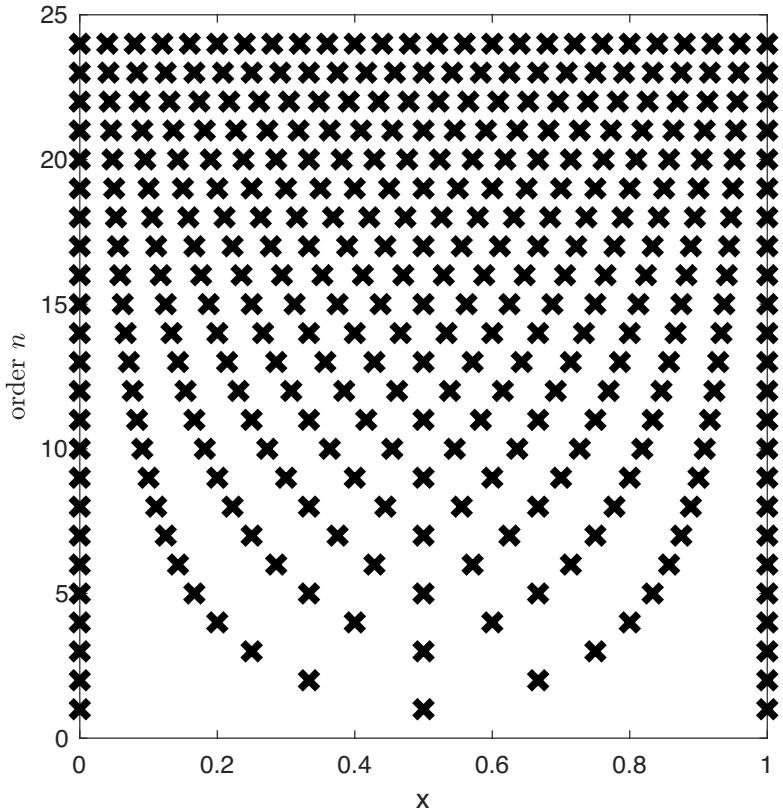
For equidistant nodes, the Lebesgue constant grows dramatically near the interval ends



What is a way out of this? $\rightsquigarrow$ Use different nodes

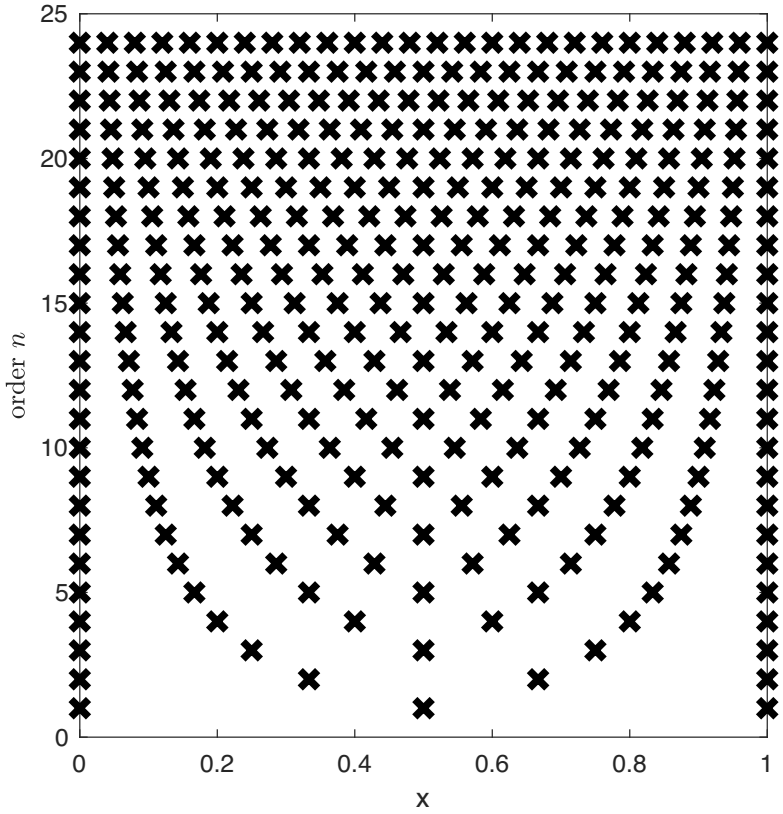
Non-equispaced points

equispaced

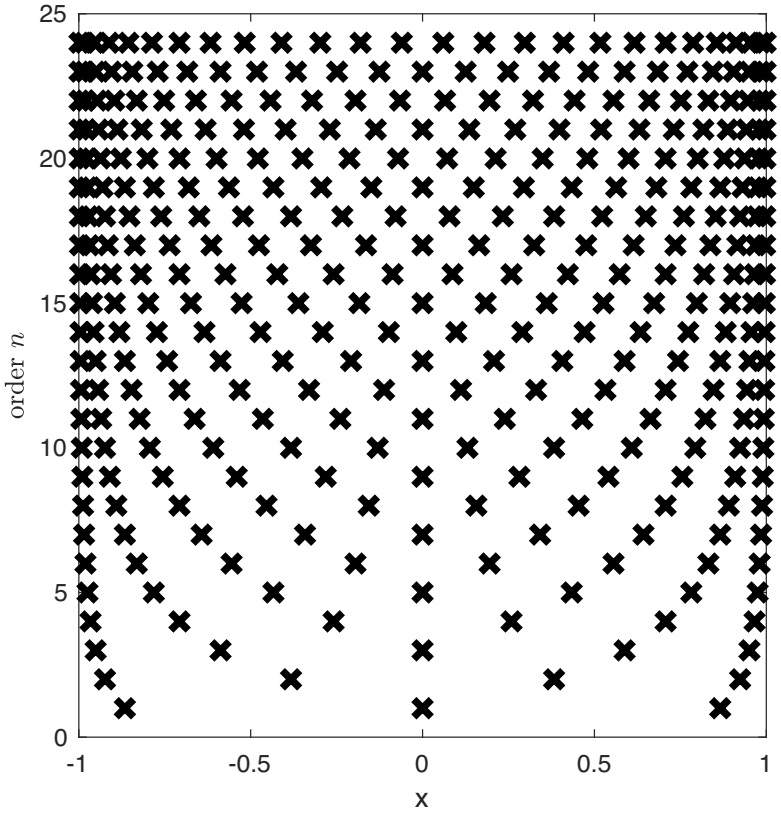


Non-equispaced points

equispaced



Chebyshev nodes



Lebesgue constants for different orders:

n	Λ_n for equidistant nodes	Λ_n for Chebyshev nodes
5	3.106292	2.104398
10	29.890695	2.489430
15	512.052451	2.727778
20	10986.533993	2.900825

Chebyshev nodes are the roots of the Chebyshev polynomials:

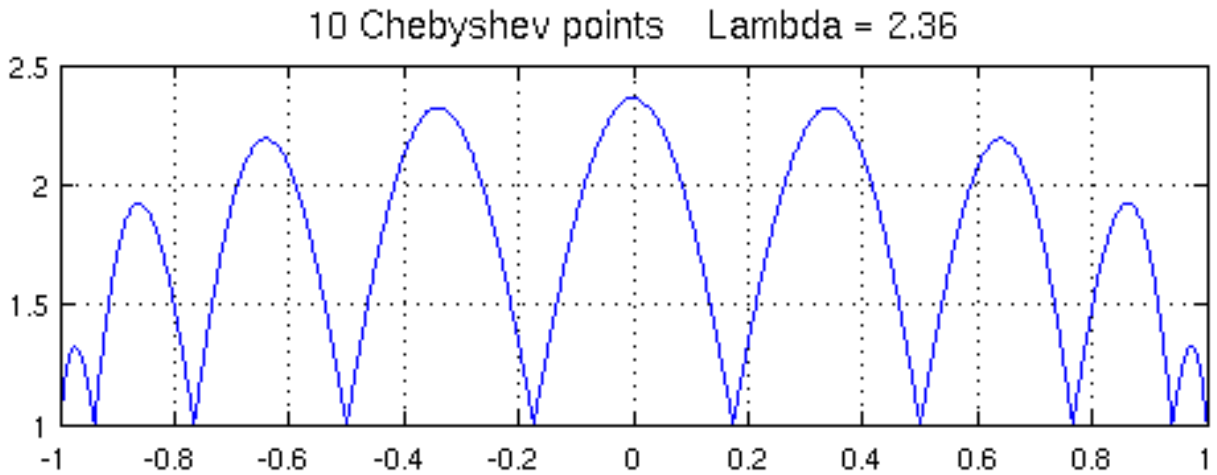
$$t_i = \cos\left(\frac{2i+1}{2n+2}\pi\right), \text{ for } i = 0, \ldots, n$$

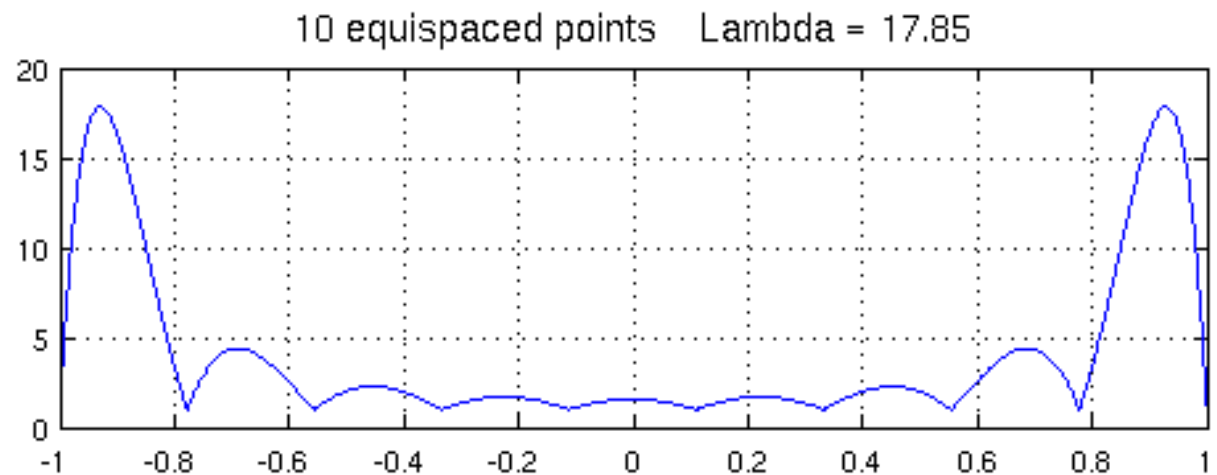
Lebesgue constant for Chebyshev nodes is bounded as

$$\Lambda_n \leq \frac{2}{\pi}\log(n+1) + 1\,,$$

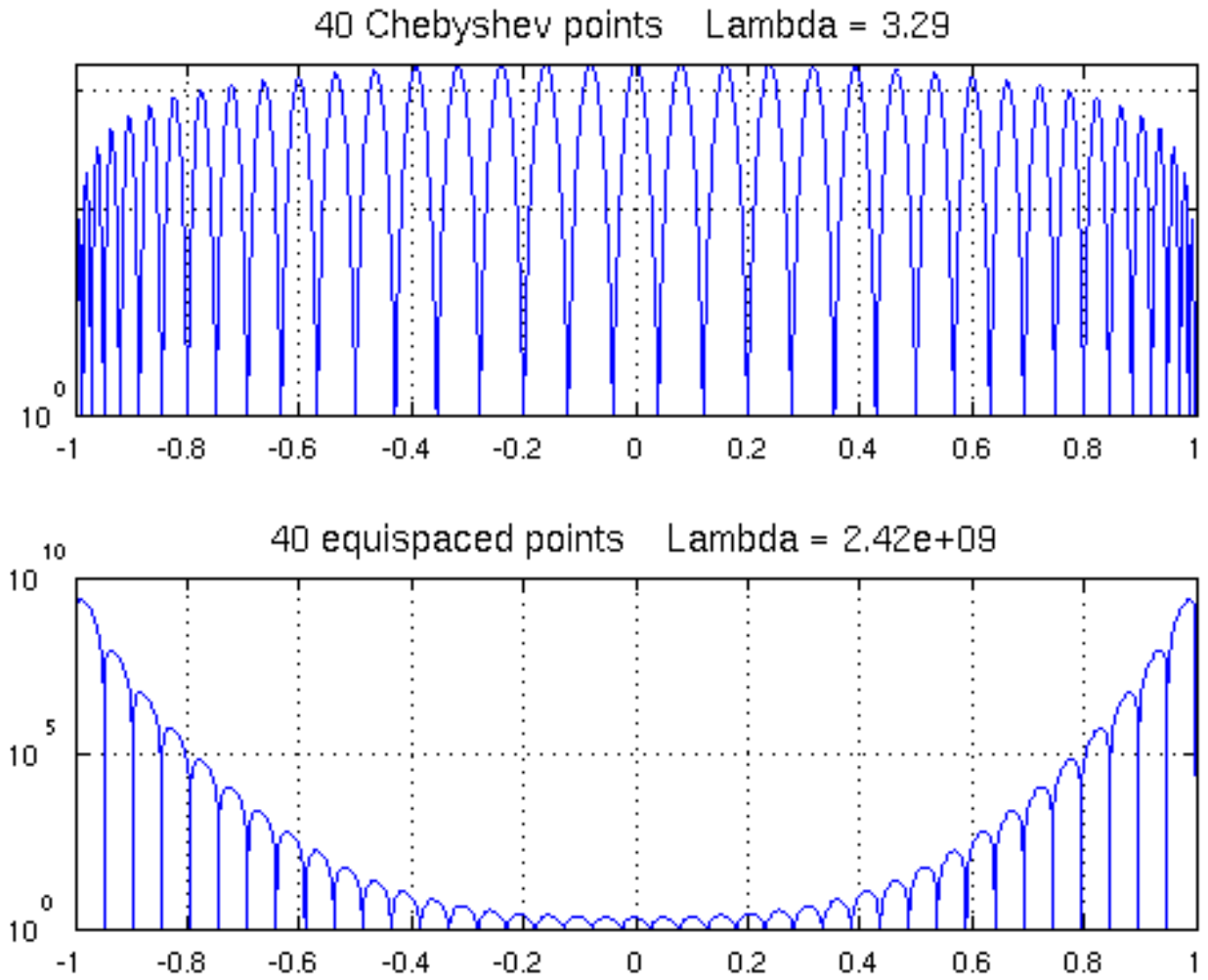
which is close to the lower bound from previous slides

Lebesgue constant for $n = 10$, uniform vs. Chebyshev nodes:





Lebesgue constant for $n = 40$, uniform vs. Chebyshev nodes:



The Lebesgue constant for Chebyshev points is bounded as

$$\Lambda_n \leq \frac{2}{\pi} \log(n+1) + 1\,,$$

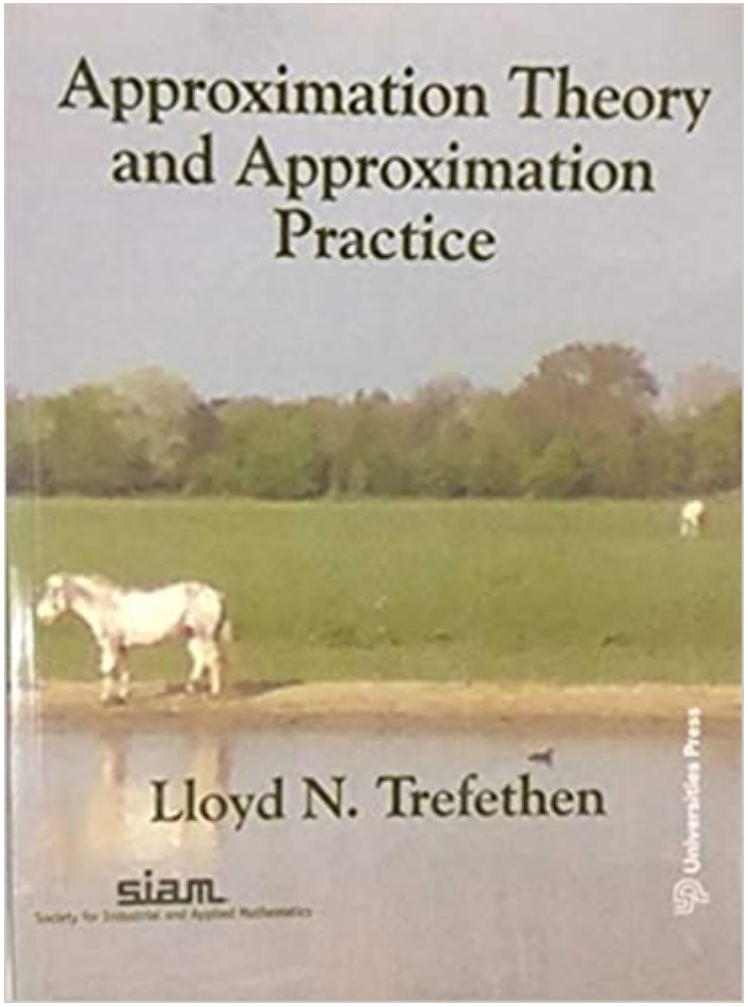
We know that for all interpolation points there exists a continuous function f such that polynomial interpolation does not converge (result by Faber, see previous slides)

This is also true for Cheybshev points! However, if f is Lipschitz continuous, then one can show uniform convergence.

The smoother the function (e.g., continuously differentiable, ν-times continuously differentiable, analytic), the faster interpolation on Chebyshev nodes converges uniformly.

This is a general principle: The more regularity there is in the function, the easier we can approximate it (faster convergence w.r.t. number of degrees of freedom of the approximation)

Beyond the material we cover here...



Pointwise evaluation of interpolating polynomials

Aitken Lemma: The interpolating polynomial $P_f(\cdot|x_0,\ldots,x_n)$ satisfies the recurrence relation

$$P_f(x|x_0,\ldots,x_n) = \frac{(x_0-x)P_f(x|x_1,\ldots,x_n) - (x_n-x)P_f(x|x_0,\ldots,x_{n-1})}{x_0-x_n}$$

Proof $\rightsquigarrow$ board

$$q(x) = \frac{(x_0 - x)\, P_p(x \mid x_1, \dots, x_n) - (x_n - x)\, P_p(x \mid x_0, \dots, x_{n-1})}{x_0 - x_n}$$

then for $i = 1, \dots, n-1$

$$q(x_i) = \frac{(x_0 - x_i)\, f(x_i) - (x_n - x_i)\, f(x_i)}{x_0 - x_n}$$

$$= \frac{x_0 - x_n}{x_0 - x_n} \quad f(x_i)$$

$$q(x_0) = \frac{0 - (x_n - x_0)\, f(x_0)}{x_0 - x_n} = f(x_0)$$

$$q(x_n) = \dots = f(x_n)$$

Pointwise evaluation of interpolating polynomials

Aitken Lemma: The interpolating polynomial $P_f(\cdot|x_0,\ldots,x_n)$ satisfies the recurrence relation

$$P_f(x|x_0,\ldots,x_n) = \frac{(x_0-x)P_f(x|x_1,\ldots,x_n) - (x_n-x)P_f(x|x_0,\ldots,x_{n-1})}{x_0-x_n}$$

Proof $\rightsquigarrow$ board

Introduce the notation

$$P_{ik} = P_f(x|x_{i-k},\ldots,x_i)\,, \qquad i \geq k$$

then $P_f(x|x_0,\ldots,x_n) = P_{nn}$ can be computed based on the Neville scheme

$$\begin{aligned} P_{i0} &= f(x_i)\,, \quad i = 0,\ldots,n \\ P_{ik} &= P_{i,k-1} + \frac{x-x_i}{x_i-x_{i-k}}(P_{i,k-1} - P_{i-1,k-1})\,, \quad i \geq k \end{aligned}$$

$\rightsquigarrow$ a numerically stable and efficient ($\mathcal{O}(n^2)$) way to evaluate (!) $P_f(x|x_0,\ldots,x_n)$

Hermite interpolation

Given

$$a = x_0 \leq x_1 \leq \ldots \leq x_n = b$$

with possibly duplicated nodes. Less information than degrees of freedom?

Hermite interpolation

Given

$$a = x_0 \leq x_1 \leq \ldots \leq x_n = b$$

with possibly duplicated nodes. Less information than degrees of freedom? Therefore, if the node x_i occurs k times, the corresponding node values correspond to $f(x_i), f'(x_i), \ldots, f^{k-1}(x_i)$.

The Hermite interpolation polynomial $p(x)$ is a polynomial of order n, which coincides with the nodal values (and, for duplicated nodes, derivatives at nodal values) at the nodes.

Hermite interpolation

Given

$$a = x_0 \leq x_1 \leq \ldots \leq x_n = b$$

with possibly duplicated nodes. Less information than degrees of freedom? Therefore, if the node x_i occurs k times, the corresponding node values correspond to $f(x_i), f'(x_i), \ldots, f^{k-1}(x_i)$.

The Hermite interpolation polynomial $p(x)$ is a polynomial of order n, which coincides with the nodal values (and, for duplicated nodes, derivatives at nodal values) at the nodes.

Aitken lemma for Hermite interpolation: The (Hermite) interpolating polynomial $P = P_f(\cdot|x_0, \ldots, x_n)$ satisfies, if $x_i \neq x_j$, the recurrence relation:

$$P_f(x|x_0, \ldots, x_n) = \frac{(x_i - x)P_f(x|x_0, \ldots, \hat{x_j}, \ldots, x_n) - (x_j - x)P_f(x|x_0, \ldots, \hat{x_i}, \ldots, x_n)}{x_i - x_j},$$

where the hat symbol indicates that the corresponding node is omitted.

Polynomial interpolation in Newton basis

The Newton basis $\omega_0, \ldots, \omega_n$ is given by

$$\omega_i(x) := \prod_{j=0}^{i-1}(x - x_j) \in \mathbb{P}_i.$$

Would like to find coefficients $c_0, c_1, \ldots, c_n$ of interpolating polynomial in Newton basis

$$P_f(x|x_0, \ldots, x_n) = c_0\omega_0(x) + c_1\omega_1(x) + \cdots + c_n\omega_n(x)$$

Newton polynomial basis

The leading coefficient a_n of the interpolation polynomial (monomial basis!)

$$P_f(x|x_0, \ldots, x_n) = a_n x^n + \cdots + a_0$$

is called the *n-th divided difference*, $[x_0, \ldots, x_n]f := a_n$.

The divided differences are the coefficients $c_0, \ldots, c_n$: The interpolation polynomial $P_f(\cdot|x_0, \ldots, x_n)$ for $x_0 \leq x_1 \leq \cdots \leq x_n$ (not necessarily distinct and thus need $f \in C^{n+1}$) is given by

$$P(x) = \sum_{i=0}^{n} [x_0, \ldots, x_i]f \, \omega_i(x).$$

Furthermore,

$$f(x) = P(x) + [x_0, \ldots, x_n, x]f \, \omega_{n+1}(x).$$

Proof $\rightsquigarrow$ board

Newton differences

$n = 0$, $p(x) = a_0$

$$p(x_0) = a_0 \overset{!}{=} f(x_0) = [x_0]f$$

$$w_0(x) = 1$$

$n > 0$

$$p_{n-1}(x) = \sum_{i=0}^{n-1} [x_0, \dots, x_i]f \; w_i(x)$$

interpolates $x_0, \dots, x_{n-1}$

$$p_n(x) = [x_0, \dots, x_n]f \; x^n + a_{n-1}x^{n-1} + \dots + a_0$$

$$p_n(x) = [x_0, \dots, x_n]f \; w_n(x) + Q_{n-1}(x)$$

$$Q_{n-1} \in \mathbb{P}_{n-1}$$

$$w_n(x) = \prod_{i=0}^{n-1} (x_i - x)$$

$$w_n(x_i) = 0 \qquad \text{for } i = 0, \dots, n-1$$

$$Q_{n-1}(x_i) = p_n(x_i) - [x_0, \dots, x_n]f \; w_n(x_i)$$

$$= f(x_i)$$

Q_{n-1} interpolates $x_0, \dots, x_{n-1}$

$$Q_{n-1} = \sum_{i=0}^{n-1} [x_0, \dots, x_i] f \, \omega_i$$

$$P_n = [x_0, \dots, x_n] f \, \omega_n + \sum_{i=0}^{n-1} [x_0, \dots, x_i] f \, \omega_i$$

$$P_{n+1} = \sum_{i=0}^{n} [x_0 \dots, x_i] f \, \omega_i + [x_0, \dots, x_n, x_{n+1}] f \, \omega_{n+1}$$

Divided differences

The divided differences $[x_0, \ldots, x_n]f$ satisfy the following properties:

- $[x_0, \ldots, x_n]P = 0$ for all $P \in \mathbb{P}_{n-1}$ $\rightsquigarrow$ hierarchy
- If $x_0 = \ldots = x_n$:

$$[x_0, \ldots, x_n]f = \frac{f^{(n)}(x_0)}{n!}$$

- The following recurrence relation holds for $x_i \neq x_j$:

$$[x_0, \ldots, x_n]f = \frac{([x_0, \ldots, \hat{x_i}, \ldots, x_n]f - [x_0, \ldots, \hat{x_j}, \ldots, x_n]f)}{x_j - x_i}$$

- $[x_0, \ldots, x_n]f = \frac{1}{n!} f^{(n)}(\tau)$ with a $a \leq \tau \leq b$, if f in C^{n+1}

Divided differences

Let us use divided differences to compute the coefficients for the Newton basis for the cubic interpolation polynomial p that satisfies $p(0) = 1$, $p(0.5) = 2$, $p(1) = 0$, $p(2) = 3$.

x_i				
0	$[x_0]f = 1$			
$\frac{1}{2}$	$[x_1]f = 2$	$[x_0 x_1]f = \frac{[x_1]f - [x_0]f}{x_1 - x_0} = \frac{2-1}{\frac{1}{2}-0} = 2$		
1	$[x_2]f = 0$	$[x_1 x_2]f = \ldots = -4$	$[x_0 x_1 x_2]f = \ldots = -6$	
2	$[x_3]f = 3$	- - - - - - - - -	- - - - - - - - -	$[x_0, \ldots, x_3]f = \frac{16}{3}$

$$p(x) = \sum_{i=0}^{3} c_i w_i(x)$$

$$p(x) = c_0 \cdot 1 + c_1 (x-0) + c_2 (x-0)(x-\tfrac{1}{2})$$
$$+ c_3 (x-0)(x-\tfrac{1}{2})(x-1)$$

$$= \cdots =$$

$$= 1 + 2x - 6x(x-\tfrac{1}{2}) + \tfrac{16}{3} x (x-\tfrac{1}{2})(x-1)$$

Divided differences

Let us use divided differences to compute the coefficients for the Newton basis for the cubic interpolation polynomial p that satisfies $p(0) = 1$, $p(0.5) = 2$, $p(1) = 0$, $p(2) = 3$.

x_i				
0	$[x_0]f = 1$			
0.5	$[x_1]f = 2$	$[x_0x_1]f = \frac{[x_1]f - [x_0]f}{x_1 - x_0} = 2$		
1	$[x_2]f = 0$	$[x_1x_2]f = \frac{[x_2]f - [x_1]f}{x_2 - x_1} = -4$	$[x_0x_1x_2]f = -6$	
2	$[x_3]f = 3$	$[x_2x_3]f = \frac{[x_3]f - [x_2]f}{x_3 - x_2} = 3$	$[x_1x_2x_3]f = \frac{14}{3}$	$\frac{16}{3}$

Thus, the interpolating polynomial is

$$p(x) = 1 + 2x + (-6)x(x - 0.5) + \frac{16}{3}x(x - 0.5)(x - 1).$$

Divided differences

Let us now use divided differences to compute the coefficients for the Newton basis for the cubic interpolation polynomial p that satisfies $p(0) = 1$, $p'(0) = 2$, $p''(0) = 1$, $p(1) = 3$.

x_i				
0	$[x_0]f = 1$			
0	$[x_0]f = 1$	$[x_0x_1]f = p'(0) = 2$		
0	$[x_0]f = 1$	$[x_1x_2]f = p'(0) = 2$	$[x_0x_1x_2]f = \frac{p''(0)}{2!} = \frac{1}{2}$	
1	$[x_3]f = 3$	$[x_2x_3]f = \frac{[x_3]f - [x_0]f}{x_3 - x_0} = 2$	$[x_1x_2x_3]f = 0$	$[x_0x_1x_2x_3]f = -\frac{1}{2}$

Thus, the interpolating polynomial is

$$p(t) = 1 + 2t + \frac{1}{2}t^2 + (-\frac{1}{2})t^3$$

Polynomial interpolation

- Polynomial interpolation
- Hermite interpolation
- (Least squares with polynomials)
- What else?

Polynomial interpolation

- Polynomial interpolation
- Hermite interpolation
- (Least squares with polynomials)
- What else? Splines, i.e., piecewise polynomial interpolation

Splines

Assume $(l+2)$ pairwise disjoint nodes:

$$a = x_0 < x_1 < \ldots < x_{l+1} = b.$$

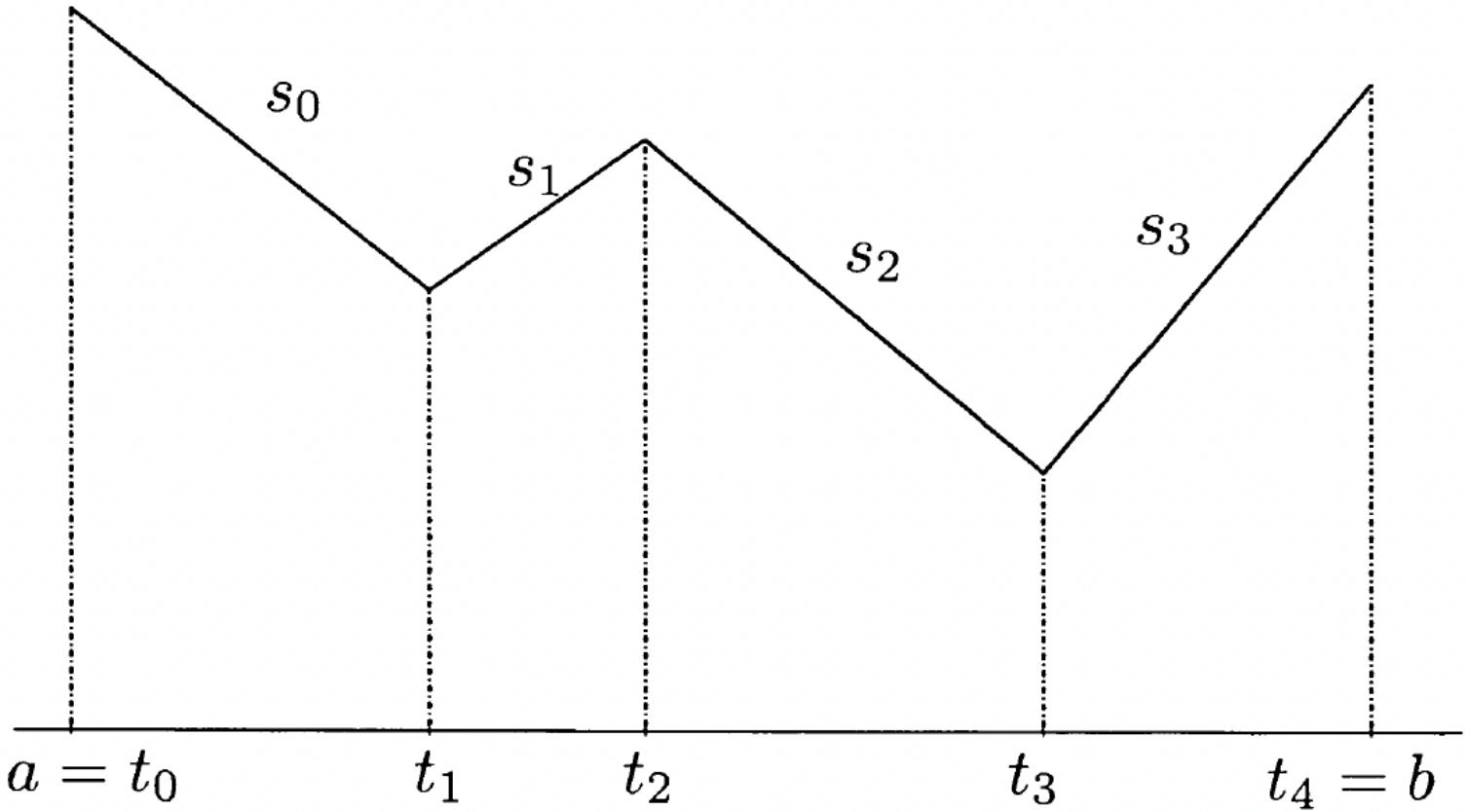
A spline of degree $k-1$ (order k) is a function in C^{k-2} which on each interval $[x_i, x_{i+1}]$ coincides with a polynomial in $\mathbb{P}_{k-1}$.

Most important examples:

- linear splines, $k = 2$
- cubic splines, $k = 4$

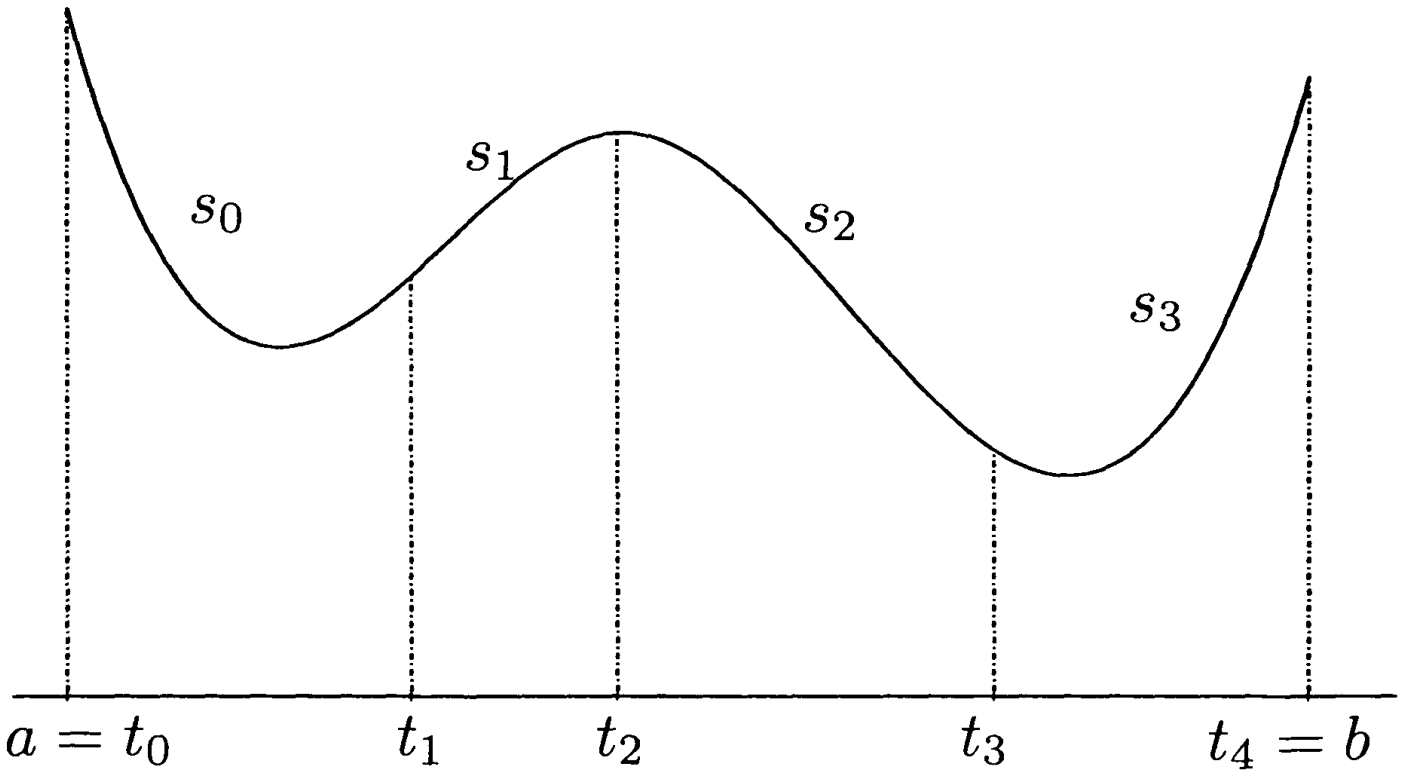
Splines

Linear spline (piecewise linear polynomial)



Splines

Cubic splines look smooth:



Trigonometric Interpolation for periodic functions

Instead of polynomials, use $\sin(jt), \cos(jt)$ for different $j \in \mathbb{N}$.

For $N \geq 1$, we define the set of complex trigonometric polynomials of degree $\leq N-1$ as

$$\boldsymbol{T}_{N-1} := \left\{ \sum_{j=0}^{N-1} c_j e^{ijt}, c_j \in \mathbb{C} \right\},$$

where $i = \sqrt{-1}$.

Complex interpolation problem: Given pairwise distinct nodes $t_0, \ldots, t_{N-1} \in [0, 2\pi)$ and corresponding nodal values $f_0, \ldots, f_{N-1} \in \mathbb{C}$, find a trigonometric polynomial $p \in \boldsymbol{T}_{N-1}$ such that $p(t_i) = f_i$, for $i = 0, \ldots, N-1$.

- There exists exactly one $p \in \boldsymbol{T}_{N-1}$, which solves this interpolation problem.

- Choose the equidistant nodes $t_k := \frac{2\pi k}{N}$ for $k = 0, \ldots, N-1$. The trigonometric polynomial that satisfies $p(t_i) = f_i$ for $i = 0, \ldots, N-1$ has the coefficients

$$c_j = \frac{1}{N} \sum_{k=0}^{N-1} e^{-\frac{2\pi ijk}{N}} f_k.$$

- For equidistant nodes, the linear map from $\mathbb{C}^N \to \mathbb{C}^N$ defined by $(f_0, \ldots, f_{N-1}) \mapsto (c_0, \ldots, c_{N-1})$ is called the discrete Fourier transformation (DFT). The Fast Fourier Transform (FFT) is a (very famous!) algorithm that computes the DFT and its inverse in $O(N \log N)$ flops. $\rightsquigarrow$ Numerical Methods II

Conclusions

- Interpolation means approximating function values in the interior of a domain when there are known samples of the function at a set of interior and boundary nodes
- Given a set of basis functions, interpolation amounts to solving a linear system of equations for the coefficients; there are clever choices to avoid explicitly computing the solution of the system of equations
- Interpolation on equidistant nodes is a bad idea: Not accurate and not stable! Instead, use Chebyshev nodes, then also higher-order polynomials are safe!
- Another option is to use piecewise polynomial interpolation such as splines, which is very common when solving PDEs numerically
- What about higher dimensions?