Part II

Interpolation and Approximation

Introduction

Distinguish two fundamental concepts:

(I) **data interpolation** (point interpolation, also includes CAD applications):

Given: data points $(\mathbf{x}_i, \mathbf{y}_i)$, $i = 1, \dots, m$, $\mathbf{x}_i \in D \subset \mathbb{R}^n$, $\mathbf{y}_i \in \mathbb{R}^d$

Goal: *reconstruction of a (continuous) function* $\mathbf{f} : D \mapsto \mathbb{R}^d$ satisfying interpolation conditions

$$\boxed{f(\mathbf{x}_i) = \mathbf{y}_i, \quad i = 1, \dots, m}$$

Additional requirements:
- smoothness of $\mathbf{f}$, e.g. $\mathbf{f} \in C^1$, etc.
- shape of $\mathbf{f}$ (positivity, monotonicity, convexity)

Example 4.0.1 (Constitutive relations (*ger.* Kennlinien) from measurements).

In this context: t, $y \mathrel{\hat{=}}$ two state variables of a physical system, a functional dependence $y = y(t)$ is assumed.

Known: several *accurate* measurements

$$(t_i, y_i)\ , \quad i = 1, \ldots, m$$

Examples:

t	y
voltage U	current I
pressure p	density ρ
magnetic field H	magnetic flux B
...	...

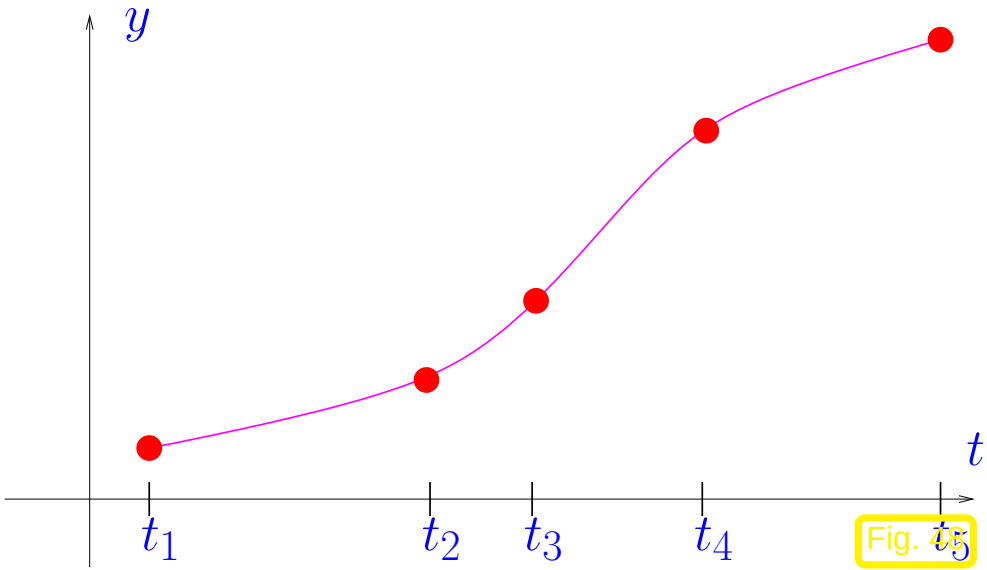


Fig. 78

Meaning of attribute "accurate": justification for interpolation. If measured values y_i were affected by considerable errors, one would not impose the interpolation conditions but opt for data fitting.

◇

(II) **function approximation**:

Given: function $\mathbf{f} : D \subset \mathbb{R}^n \mapsto \mathbb{R}^d$ (often in procedural form `y=feval(x)`)

Goal: Find a "simple"$^{(*)}$ function $\widetilde{\mathbf{f}} : D \mapsto \mathbb{R}^d$ such that the difference $\mathbf{f} - \widetilde{\mathbf{f}}$ is "small"$^{(\clubsuit)}$

$^{(*)}$: "simple" $\sim$ described by small amount of information, easy to evaluate (e.g, polynomial or piece-wise polynomial $\widetilde{\mathbf{f}}$)

$^{(\clubsuit)}$ "small" $\sim$ $\left\|\mathbf{f}-\widetilde{\mathbf{f}}\right\|$ small for some norm $\|\cdot\|$ on space $C(D)$ of continous functions, e.g. L^2-norm $\|\mathbf{g}\|_2^2 := \int\limits_D |\mathbf{g}(x)|^2 dx$, maximum norm $\|\mathbf{g}\|_\infty := \max\limits_{x\in D} |\mathbf{g}(x)|$

Example 4.0.2 (Taylor approximation).

$$f \in C^k(I), \quad I \text{ interval}, \quad k \in \mathbb{N}, \qquad T_k(t) := \frac{f^{(k)}(t_0)}{k!}\,(t-t_0)^k, \quad t_0 \in I\ .$$

The Taylor polynomial T_k of degree k approximates f in a neighbourhood $J \subset I$ of t_0 (J can be small!). The Taylor approximation is easy and direct but inefficient: a polynomial of lower degree gives the same accuracy.

$\diamond$

Another technique: Approximation by interpolation

$$\mathbf{f} \xrightarrow{\text{sampling}} (\mathbf{x}_i, \mathbf{y}_i := f(\mathbf{x}_i))_{i=1}^m \xrightarrow{\text{interpolation}} \widetilde{\mathbf{f}}: \quad \widetilde{\mathbf{f}}(\mathbf{x}_i) = \mathbf{y}_i\ .$$

free choice of nodes $\mathbf{x}_i$

Remark 4.0.3 (Interpolation and approximation: enabling technologies).

Approximation and interpolation are useful for several numerical tasks, like integration, differentiation and computation of the solutions of differential equations.

► this is a “foundations” part of the course △

Remark 4.0.4 (Function representation).

! General function $f : D \subset \mathbb{R} \mapsto \mathbb{K}$, D interval, contains an “infinite amount of information”.

? How to represent f on a computer?

→ Idea: parametrization, a finite number of parameters $\alpha_1, \ldots, \alpha_n$, $n \in \mathbb{N}$, characterizes f.

Special case: Representation with finite linear combination of basis functions

$b_j : D \subset \mathbb{R} \mapsto \mathbb{K}$, $j = 1, \ldots, n$:

$$f = \sum_{j=1}^{n} \alpha_j b_j \quad , \quad \alpha_j \in \mathbb{K} \, .$$

➜ $f \in$ finite dimensional function space $V_n := \mathrm{Span}\,\{b_1, \ldots, b_n\}$.

$b_j(t) = t^{j-1}$ leads to polynomial interpolation

$b_j(t) = \cos((j-1)\arccos t)$ leads to Chebychev interpolation

$b_j(t) = e^{2\pi i j t}$ leads to trigonometrical interpolation

△

5 Polynomial Interpolation

5.1 Polynomials

Notation: Vector space of the polynomials of degree $\leq k$, $k \in \mathbb{N}$:

$$\mathcal{P}_k := \{t \mapsto \alpha_k t^k + \alpha_{k-1} t^{k-1} + \cdots + \alpha_1 t + \alpha_0 \,,\, \alpha_j \in \mathbb{K}\} \,. \tag{5.1.1}$$

Terminology: the functions $t \mapsto t^k$, $k \in \mathbb{N}_0$, are called monomials

$t \mapsto \alpha_k t^k + \alpha_{k-1} t^{k-1} + \cdots + \alpha_0$ = monomial representation of a polynomial.

Obvious: $\mathcal{P}_k$ is a vector space. What is its dimension?

Theorem 5.1.1 (Dimension of space of polynomials)**.**

$$\dim \mathcal{P}_k = k+1 \quad \textit{and} \quad \mathcal{P}_k \subset C^\infty(\mathbb{R}).$$

Proof. Dimension formula by linear independence of monomials.

Why are polynomials important in computational mathematics ?

- ➔ Easy to compute, integrate and differentiate
- ➔ Vector space & algebra
- ➔ Analysis: Taylor polynomials & power series

Remark 5.1.1 (Polynomials in python)*.*

python: $\alpha_k t^k + \alpha_{k-1} t^{k-1} + \cdots + \alpha_0$ ➔ Vector $(\alpha_k, \alpha_{k-1}, \ldots, \alpha_0)$ (ordered!). △

Remark 5.1.2 (Horner scheme)*.*

Evaluation of a polynomial in monomial representation: Horner scheme

$$p(t) = (t \cdots t(t(\alpha_n t + \alpha_{n-1}) + \alpha_{n-2}) + \cdots + \alpha_1) + \alpha_0 \,. \tag{5.1.2}$$

Code 5.1.3: Horner scheme, polynomial in python format

```
def horner(p, x):
    y = p[0]
    for i in xrange(1,len(p)):
        y = x * y + p[i]
    return y
```

Asymptotic complexity: $O(n)$

Use: numpy "built-in"-function `polyval`(p,x);.

△

5.2 Newton basis and divided differences [10, Sect. 8.2.4]

Want: adding another data point should not affect *all* basis polynomials!

Tool: "update friendly" representation: Newton basis for $\mathcal{P}_n$

$$N_0(t) := 1\,, \quad N_1(t) := (t - t_0)\,, \quad \dots \quad , \quad N_n(t) := \prod_{i=0}^{n-1}(t - t_i)\,. \tag{5.2.1}$$

Note: $N_n \in \mathcal{P}_n$ with *leading coefficient* 1.

➢ LSE for polynomial interpolation problem in Newton basis:

$$a_j \in \mathbb{R}: \quad a_0 N_0(t_j) + a_1 N_1(t_j) + \cdots + a_n N_n(t_j) = y_j, \quad j = 0, \dots, n\,.$$

$\Leftrightarrow$ triangular linear system

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & (t_1 - t_0) & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 1 & (t_n - t_0) & \cdots & \prod_{i=0}^{n-1}(t_n - t_i) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}\,.$$

Solution of the system with forward substitution:

$$a_0 = y_0\,,$$

$$
\begin{aligned}
a_1 &= \frac{y_1 - a_0}{t_1 - t_0} = \frac{y_1 - y_0}{t_1 - t_0}\,, \\
a_2 &= \frac{y_2 - a_0 - (t_2 - t_0)a_1}{(t_2 - t_0)(t_2 - t_1)} = \frac{y_2 - y_0 - (t_2 - t_0)\frac{y_1 - y_0}{t_1 - t_0}}{(t_2 - t_0)(t_2 - t_1)} = \frac{\dfrac{y_2 - y_0}{t_2 - t_0} - \dfrac{y_1 - y_0}{t_1 - t_0}}{t_2 - t_1}\,, \\
&\vdots
\end{aligned}
$$

Observation: same quantities computed again and again !

In order to find a better algorithm, we turn to a new interpretation of the coefficients a_j of the interpolating polynomials in Newton basis.

Newton basis polynomial $N_j(t)$: degree j and leading coefficient 1

$\Rightarrow$ a_j is the leading coefficient of the interpolating polynomial $p_{0,\dots,j}$

➢ Recursion for leading coefficients $a_{\ell,\dots,m}$ of interpolating polynomials $p_{\ell,\dots,m}$, $0 \le \ell \le m \le n$:

$$
a_{\ell,\dots,m} = \frac{a_{\ell+1,\dots,m} - a_{\ell,\dots,m-1}}{t_m - t_\ell}\,.
$$

Simpler and more efficient algorithm using divided differences:

$$
\begin{aligned}
y[t_i] &= y_i \\
y[t_i,\ldots,t_{i+k}] &= \frac{y[t_{i+1},\ldots,t_{i+k}] - y[t_i,\ldots,t_{i+k-1}]}{t_{i+k}-t_i} \qquad \text{(recursion)} \qquad (5.2.2)
\end{aligned}
$$

Recursive calculation by divided differences scheme

$$
\begin{array}{c|ccccccc}
t_0 & y[t_0] & & & & & & \\
 & & > & y[t_0,t_1] & & & & \\
t_1 & y[t_1] & & & > & y[t_0,t_1,t_2] & & \\
 & & > & y[t_1,t_2] & & & > & y[t_0,t_1,t_2,t_3], \\
t_2 & y[t_2] & & & > & y[t_1,t_2,t_3] & & \\
 & & > & y[t_2,t_3] & & & & \\
t_3 & y[t_3] & & & & & &
\end{array}
\qquad (5.2.3)
$$

the elements are computed from left to right, every "$>$" means recursion (5.2.2).

If a new datum (t_{n+1}, y_{n+1}) is added, it is enough to compute $n+2$ new terms

$$y[t_{n+1}],\ y[t_n,t_{n+1}],\ \ldots, y[t_0,\ldots,t_{n+1}].$$

Code 5.2.1: Divided differences, recursive implementation, in situ computation

```
def divdiff(t, y):
    n = y.shape[0] - 1
    if n > 0:
        y[0:n] = divdiff(t[0:n], y[0:n])
        for j in xrange(0,n):
            y[n] = (y[n] - y[j]) / (t[n] - t[j])
    return y
```

Code 5.2.2: Divided differences, non-recursive implementation, in situ computation

```
def divdiff(t, y):
    n = y.shape[0] - 1
    for l in xrange(0,n):
        for j in xrange(l+1,n+1):
            y[j] = (y[j] - y[l]) / (t[j] - t[l])
    return y
```

By derivation: computed finite differences are the coefficients of interpolating polynomials in Newton basis:

$$p(t) = a_0 + a_1(t - t_0) + a_2(t - t_0)(t - t_1) + \cdots + a_n \prod_{j=0}^{n-1} (t - t_j) \tag{5.2.4}$$

$$a_0 = y[t_0],\ a_1 = y[t_0, t_1],\ a_2 = y[t_0, t_1, t_2],\ \ldots.$$

"Backward evaluation" of $p(t)$ in the spirit of Horner's scheme (→ Rem. 5.1.2, [10, Alg. 8.20]):

$$p \leftarrow a_n, \quad p \leftarrow (t - t_{n-1})p + a_{n-1}, \quad p \leftarrow (t - t_{n-2})p + a_{n-2}, \quad \dots .$$

Computational effort:

- $O(n^2)$ for computation of divided differences,
- $O(n)$ for every single evaluation of $p(t)$.

Remark 5.2.3 (Divided differences and derivatives).

If $y_0, \dots, y_n$ are the values of a smooth function f in the points $t_0, \dots, t_n$, that is, $y_j := f(t_j)$, then

$$y[t_i, \dots, t_{i+k}] = \frac{f^{(k)}(\xi)}{k!}$$

for a certain $\xi \in [t_i, t_{i+k}]$, see [10, Thm. 8.21].

△

5.3 Error estimates for polynomial interpolation

Focus: approximation of a function by global polynomial interpolation (→ Sect. **??**)

Remark 5.3.1 (Approximation by polynomials).

? Is it always possible to approximate a continuous function by polynomials?

✓ Yes! Recall the Weierstrass theorem:
A continuous function f on the interval $[a,b] \subset \mathbb{R}$ can be uniformly approximated by polynomials.

! But not by the interpolation on a fixed mesh [42, pag. 331]:
Given a sequence of meshes of increasing size $\{\mathcal{T}_j\}_{j=1}^{\infty}$, $\mathcal{T}_j = \{x_1^{(j)}, \dots, x_j^{(j)}\} \subset [a,b]$, $a \leq x_1^{(j)} < x_2^{(j)} < \cdots < x_j^{(j)} \leq b$, there exists a continuous function f such that the sequence interpolating polynomials of f on $\mathcal{T}_j$ does not converge uniformly to f as $j \to \infty$.

△

We consider Lagrangian polynomial interpolation on node set

$$\mathcal{T} := \{t_0, \dots, t_n\} \subset I,\ I \subset \mathbb{R}, \text{ interval of length } |I|.$$

Notation: For a continuous function $f : I \mapsto \mathbb{K}$ we define the polynomial interpolation operator

$$\mathsf{I}_{\mathcal{T}}(f) := \mathsf{I}_{\mathcal{T}}(\mathbf{y}) \in \mathcal{P}_n \qquad \text{with} \qquad \mathbf{y} := (f(t_0), \dots, f(t_n))^T \in \mathbb{K}^{n+1}.$$

Goal: estimate of the interpolation error norm $\|f - \mathsf{I}_{\mathcal{T}} f\|$ (for some norm on $C(I)$).

Focus: asymptotic behavior of interpolation error for $n \to \infty$

Example 5.3.2 (Asymptotic behavior of polynomial interpolation error).

Interpolation of $f(t) = \sin t$ on equispaced nodes in $I = [0, \pi]$: $\mathcal{T} = \{j\pi/n\}_{j=0}^n$.
Interpolating polynomial $p := \mathsf{I}_{\mathcal{T}} f \in \mathcal{P}_n$.

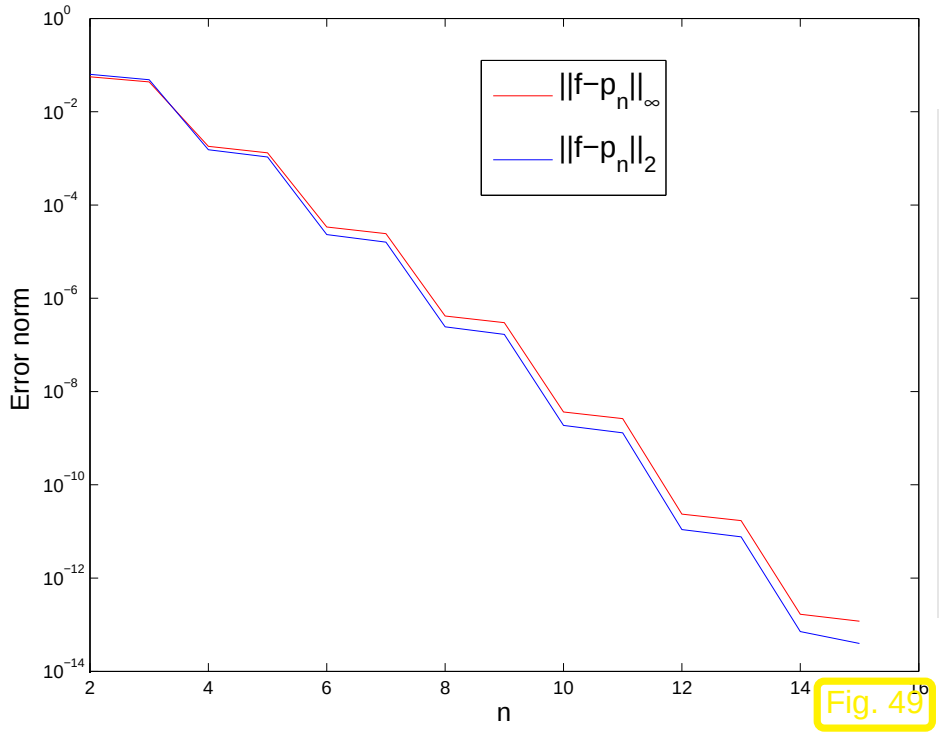


Fig. 49

computer-experiment: computation of the norms.

- L^∞-norm: sampling on a grid of meshsize $\pi/1000$.
- L^2-norm: numerical quadrature ($\longrightarrow$ Chapter 7) with trapezoidal rule (7.2.2) on a grid of meshsize $\pi/1000$.

$\diamond$

In the previous experiment we observed a clearly visible qualitative behavior of $\|f - \mathsf{I}_{\mathcal{T}} f\|$ as we increased the polynomial degree n. The prediction of the decay law for $\|f - \mathsf{I}_{\mathcal{T}} f\|$ is one goal in the study of interpolation errors.

Often this goal can be achieved, even if a rigorous quantitative bound for a norm of the interpolation error remains elusive.

Important terminology for the qualitative description of $\|f - \mathsf{I}_{\mathcal{T}} f\|$ as a function of the polynomial degree n:

$$\exists\, C \neq C(n) > 0: \quad \|f - \mathsf{I}_{\mathcal{T}} f\| \leq C\, T(n) \quad \text{for } n \to \infty\,. \tag{5.3.1}$$

Classification (best bound for $T(n)$):

$$\begin{array}{lll} \exists\, p > 0: & T(n) \leq n^{-p} & : \text{ algebraic convergence, with rate } p > 0\,, \\ \exists\, 0 < q < 1: & T(n) \leq q^{n} & : \text{ exponential convergence}\,, \end{array} \quad \forall n \in \mathbb{N}\,.$$

Convergence behavior of interpolation error is often expressed by means of the Landau-O-notation:

$$\begin{array}{ll} \text{Algebraic convergence:} & \|f - \mathsf{I}_{\mathcal{T}} f\| = O(n^{-p}) \\ \text{Exponential convergence:} & \|f - \mathsf{I}_{\mathcal{T}} f\| = O(q^{n}) \end{array} \quad \text{for } n \to \infty \text{ (“asymptotic!”)}$$

Remark 5.3.3 (Exploring convergence).

Given: pairs (n_i, ϵ_i), $i = 1, 2, 3, \ldots$, $n_i \mathrel{\hat{=}}$ polynomial degrees, $\epsilon_i \mathrel{\hat{=}}$ norms of interpolation error

❶ Conjectured: algebraic convergence: $\epsilon_i \approx Cn^{-p}$

$$\log(\epsilon_i) \approx \log(C) - p\log n_i \qquad \text{(affine linear in log-log scale).}$$

Apply linear regression (`numpy.polyfit`) to points $(\log n_i, \log \epsilon_i)$ ➢ estimate for rate p.

❶ Conjectured: exponential convergence: $\epsilon_i \approx C\exp(-\beta n_i)$

$$\log \epsilon_i \approx \log(C) - \beta n_i \qquad \text{(affine linear in lin-log scale). .}$$

Apply linear regression (Ex. 3.0.1, `numpy.polyfit`) to points $(n_i, \log \epsilon_i)$ ➢ estimate for $q := \exp(-\beta)$.

☞ Fig. 49: we suspect exponential convergence in Ex. 5.3.2.

△

Beware: same concept $\leftrightarrow$ different meanings:

- convergence of a sequence (e.g. of iterates $\boldsymbol{x}^{(k)}$ $\longrightarrow$ Sect. 1.1)
- convergence of an approximation (dependent on an approximation parameter, e.g. n)

Example 5.3.4 (Runge's example). $\longrightarrow$ Ex. **??**

Polynomial interpolation of $f(t) = \frac{1}{1+t^2}$ with equispaced nodes:

$$\mathcal{T} := \left\{ t_j := -5 + \frac{10}{n}\, j \right\}_{j=0}^{n} \quad , \quad y_j = \frac{1}{1+t_j^2} \,.j = 0, \ldots, n \,.$$

Code 5.3.5: Computing the interpolation error for Runge's example

```
# Interpolation error plot for Runge's example

from numpy import linspace, array, polyfit, polyval, max, abs, hstack,
    vstack
from matplotlib.pyplot import *

# Exact values
x = linspace(-5, 5, 1001)
f = lambda x: 1.0 / (1.0 + x**2)

fv = f(x)

# Compute approximation of increasing degree
err = []

for d in xrange(1, 21):
    t = linspace(-5, 5, d+1)
```

```
    p = polyfit(t, f(t), d)
    y = polyval(p, x)

    err.append( hstack([d, max(abs(y-fv))]) )

err = array(err)

# Plot
figure()
semilogy(err[:,0], err[:,1], "r--+")
xlabel(r"Degree d")
ylabel(r"Interpolation error (maximum norm)")
savefig("../PICTURES/rungeerrmax.eps")
```

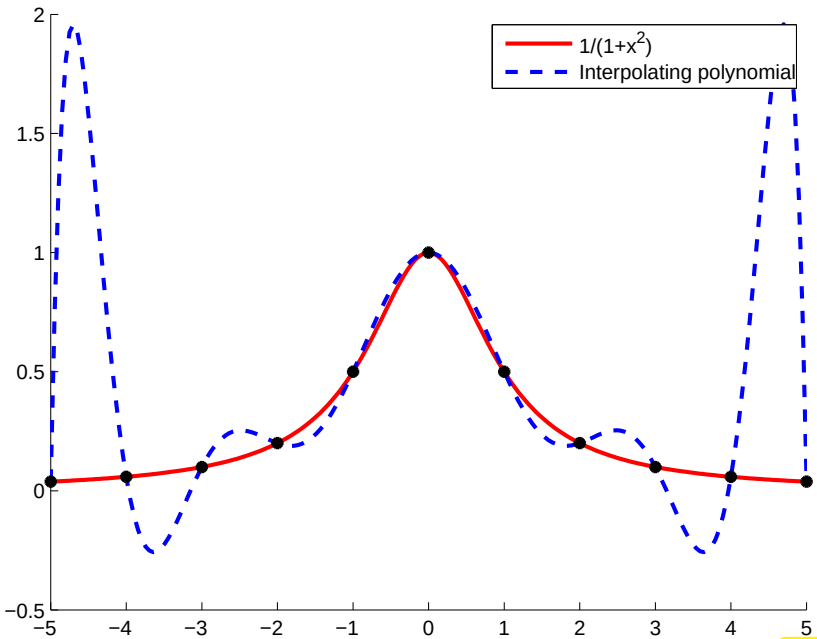


Fig. 50

Interpolating polynomial, $n = 10$

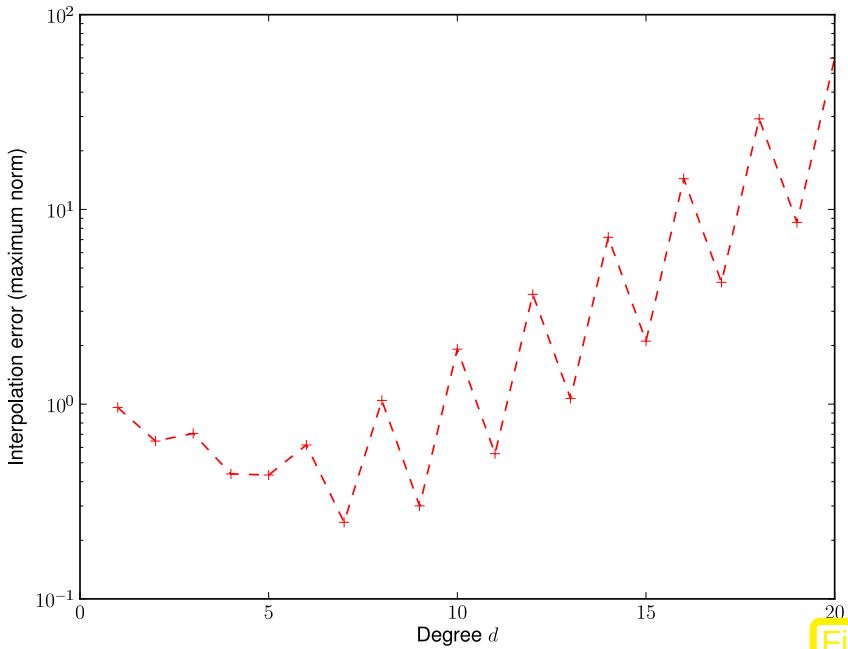


Fig. 51

Approximate $\|f - \mathcal{I}_{\mathcal{T}} f\|_\infty$ on $[-5, 5]$

Note: approximation of $\|f - \mathcal{I}_{\mathcal{T}} f\|_\infty$ by *sampling* in 1000 equidistant points.

Observation: Strong oscillations of $\mathsf{I}_{\mathcal{T}} f$ near the endpoints of the interval:

$$\|f - \mathsf{I}_{\mathcal{T}} f\|_{L^\infty(]-5,5[)} \xrightarrow{n\to\infty} \infty \ .$$

$\diamond$

Theorem 5.3.1 (Representation of interpolation error). *[10, Thm. 8.22], [29, Thm. 37.4]*
$f \in C^{n+1}(I)$: $\forall\, t \in I$: $\exists\, \tau_t \in]\min\{t, t_0, \dots, t_n\}, \max\{t, t_0, \dots, t_n\}[$:

$$f(t) - \mathsf{I}_{\mathcal{T}}(f)(t) = \frac{f^{(n+1)}(\tau_t)}{(n+1)!} \cdot \prod_{j=0}^{n} (t - t_j) \ . \tag{5.3.2}$$

Proof. Write $q(t) := \prod_{j=0}^{n} (t - t_j) \in \mathcal{P}_{n+1}$ and fix $t \in I$.

$$t \neq t_j \quad \Rightarrow \quad q(t) \neq 0 \quad \Rightarrow \quad \exists c(t) \in \mathbb{R}\colon \quad f(t) - \mathsf{I}_{\mathcal{T}}(f)(t) = cq(t)$$

$\varphi(x) := f(x) - \mathsf{I}_{\mathcal{T}}(f)(x) - cq(x)$ has $n+2$ *distinct* zeros $t_0, \dots, t_n, t$. By iterated application of the mean value theorem [52, Thm .5.2.1], we conclude

$$\varphi^{(m)} \text{ has } n+2-m \text{ distinct zeros in } I.$$

$$\Rightarrow \qquad \exists \tau_t \in I\colon \quad \varphi^{(n+1)}(x) = f^{(n+1)}(\tau_t) - c(n+1)! = 0 \ .$$

This fixes the value of $c = \frac{f^{(n+1)}(\tau_t)}{(n+1)!}$. □

The theorem can also be proved using the following lemma.

Lemma 5.3.2 (Error of the polynomial interpolation). *For* $f \in C^{n+1}(I)$*:* $\forall\, t \in I$*:*

$$f(t) - \mathsf{I}_{\mathcal{T}}(f)(t) = \int_0^1 \int_0^{\tau_1} \cdots \int_0^{\tau_{n-1}} \int_0^{\tau_n} f^{(n+1)}(t_0 + \tau_1(t_1 - t_0) + \cdots$$

$$+ \tau_n(t_n - t_{n-1}) + \tau(t - t_n))\,\mathrm{d}\tau\mathrm{d}\tau_n \cdots \mathrm{d}\tau_1 \cdot \prod_{j=0}^{n}(t - t_j)\ .$$

Proof. By induction on n, use (**??**) and the fundamental theorem of calculus [44, Sect. 3.1]:

Remark 5.3.6. Lemma 5.3.2 holds also for general polynomial interpolation with multiple nodes, see (**??**).

△

In the *equation* (5.3.2) we can

- first bound the right hand side via $f^{(n+1)}(\tau_t) \leq \left\| f^{(n+1)} \right\|_{L^\infty(I)}$,
- then increase the right hand side further by switching to the maximum (in modulus) w.r.t. t (the resulting bound does no longer depend on t!),

- and, finally, take the maximum w.r.t. t on the left of $\leq$.

This yields the following interpolation error estimate:

$$\text{Thm. 5.3.1} \quad \Rightarrow \quad \|f - \mathsf{I}_{\mathcal{T}} f\|_{L^\infty(I)} \leq \frac{\left\|f^{(n+1)}\right\|_{L^\infty(I)}}{(n+1)!} \max_{t\in I} |(t-t_0)\cdot\cdots\cdot(t-t_n)| \ . \qquad (5.3.3)$$

Interpolation error estimate requires smoothness!

Example 5.3.7 (Error of polynomial interpolation). Ex. 5.3.2 cnt'd

Theoretical explanation for exponential convergence observed for polynomial interpolation of $f(t) = \sin(t)$ on equidistant nodes: by Thm. 5.3.2 and (5.3.3)

$$\left\|f^{(k)}\right\|_{L^\infty(I)} \leq 1 \ , \quad \forall k \in \mathbb{N}_0 \quad \Rightarrow \quad \begin{aligned} \|f-p\|_{L^\infty(I)} &\leq \frac{1}{(1+n)!} \max_{t\in I} \left|(t-0)(t-\tfrac{\pi}{n})(t-\tfrac{2\pi}{n})\cdot\cdots\cdot(t-\pi)\right| \\ &\leq \frac{1}{n+1}\left(\frac{\pi}{n}\right)^{n+1} \ . \end{aligned}$$

➜ Uniform asymptotic exponential convergence of the interpolation polynomials (independently of the set of nodes $\mathcal{T}$. In fact, $\|f-p\|_{L^\infty(I)}$ decays even faster than exponential!)

Example 5.3.8 (Runge's example). Ex. 5.3.4 cnt'd

How can the blow-up of the interpolation error observed in Ex. 5.3.4 be reconciled with Thm. 5.3.2 ?

Here $f(t)=\frac{1}{1+t^2}$ allows only to conclude $|f^{(n)}(t)| = 2^n n! \cdot O(|t|^{-2-n})$ for $n\to\infty$.

➜ Possible blow-up of error bound from Thm. 5.3.1 $\to\infty$ for $n\to\infty$. ◇

Remark 5.3.9 (L^2-error estimates for polynomial interpolation).

Thm. 5.3.1 gives error estimates for the L^∞-Norm. And the other norms?

From Lemma. 5.3.2 using Cauchy-Schwarz inequality:

$$\|f-\mathsf{I}_{\mathcal{T}}(f)\|^2_{L^2(I)} = \int_I \left| \int_0^1 \int_0^{\tau_1} \cdots \int_0^{\tau_{n-1}} \int_0^{\tau_n} f^{(n+1)}(\ldots)\,\mathrm{d}\tau\mathrm{d}\tau_n\cdots\mathrm{d}\tau_1 \cdot \underbrace{\prod_{j=0}^{n}(t-t_j)}_{|t-t_j|<|I|} \right|^2 \mathrm{d}t$$

$$\leq \int_I |I|^{2n+2} \underbrace{\mathrm{vol}_{(n+1)}(S_{n+1})}_{=1/(n+1)!} \int_{S_{n+1}} |f^{(n+1)}(\ldots)|^2 \,\mathrm{d}\boldsymbol{\tau}\,\mathrm{d}t$$

$$= \int_I \frac{|I|^{2n+2}}{(n+1)!} \int_I \underbrace{\mathrm{vol}_{(n)}(C_{t,\tau})}_{\leq 2^{(n-1)/2}/n!} |f^{(n+1)}(\tau)|^2 \,\mathrm{d}\tau\mathrm{d}t \;,$$

$$S_{n+1} := \{\mathbf{x} \in \mathbb{R}^{n+1} \colon 0 \leq x_n \leq x_{n-1} \leq \cdots \leq x_1 \leq 1\} \quad \text{(unit simplex)} \;,$$

$$C_{t,\tau} := \{\mathbf{x} \in S_{n+1} \colon t_0 + x_1(t_1 - t_0) + \cdots + x_n(t_n - t_{n-1}) + x_{n+1}(t - t_n) = \tau\} \;.$$

This gives the bound for the L^2-norm of the error:

$$\Rightarrow \qquad \|f - \mathsf{I}_{\mathcal{T}}(f)\|_{L^2(I)} \leq \frac{2^{(n-1)/4}|I|^{n+1}}{\sqrt{(n+1)!n!}} \left(\int_I |f^{(n+1)}(\tau)|^2 \,\mathrm{d}\tau\right)^{1/2} . \tag{5.3.4}$$

Notice: $f \mapsto \left\| f^{(n)} \right\|_{L^2(I)}$ defines a seminorm on $C^{n+1}(I)$

(Sobolev-seminorm, measure of the smoothness of a function).

Estimates like (5.3.4) play a key role in the analysis of numerical methods for solving partial differential equations ($\longrightarrow$ course “Numerical methods for partial differential equations”).

5.4 Chebychev Interpolation

Perspective: function approximation by polynomial interpolation

➢ Freedom to choose interpolation nodes judiciously

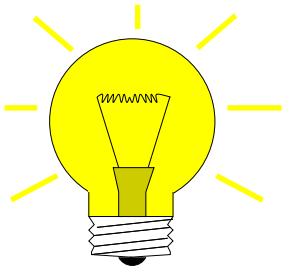
5.4.1 Motivation and definition

Mesh of nodes: $\mathcal{T} := \{t_0 < t_1 < \cdots < t_{n-1} < t_n\}$, $n \in \mathbb{N}$,
function $f : I \to \mathbb{R}$ continuous; without loss of generality $I = [-1, 1]$.

Thm. 5.3.1:

$$\|f - p\|_{L^\infty(I)} \leq \frac{1}{(n+1)!} \left\| f^{(n+1)} \right\|_{L^\infty(I)} \|w\|_{L^\infty(I)} \ ,$$

$$w(t) := (t - t_0) \cdot \cdots \cdot (t - t_n) \ .$$



Idea: choose nodes $t_0, \dots, t_n$ such that $\|w\|_{L^\infty(I)}$ is minimal!

Equivalent to finding $q \in \mathcal{P}_{n+1}$, with leading coefficient $= 1$, such that $\|q\|_{L^\infty(I)}$ is minimal.

Choice of $t_0, \dots, t_n$ = zeros of q (caution: t_j must belong to I).

Heuristic:
- t^* extremal point of q ➔ $|q(t^*)| = \|q\|_{L^\infty(I)}$,
- q has $n+1$ zeros in I,
- $|q(-1)| = |q(1)| = \|q\|_{L^\infty(I)}$.

Definition 5.4.1 (Chebychev polynomial)**.**
The nth *Chebychev polynomial is* $T_n(t) := \cos(n \arccos t)$, $\quad -1 \le t \le 1$.

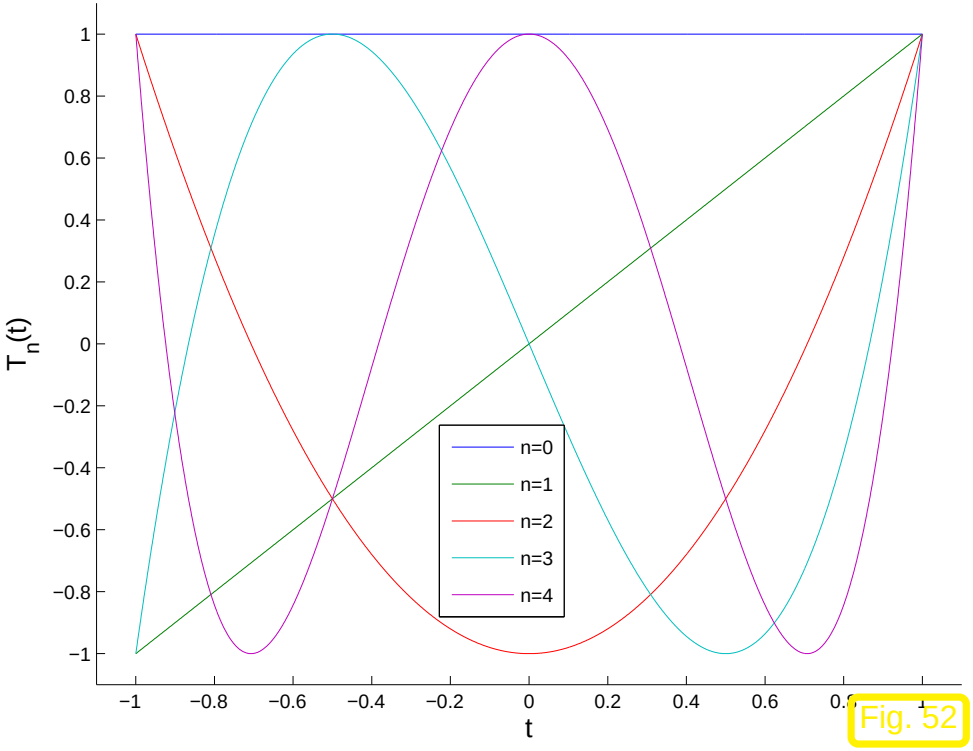


Fig. 52

Chebychev polynomials $T_0, \dots, T_4$

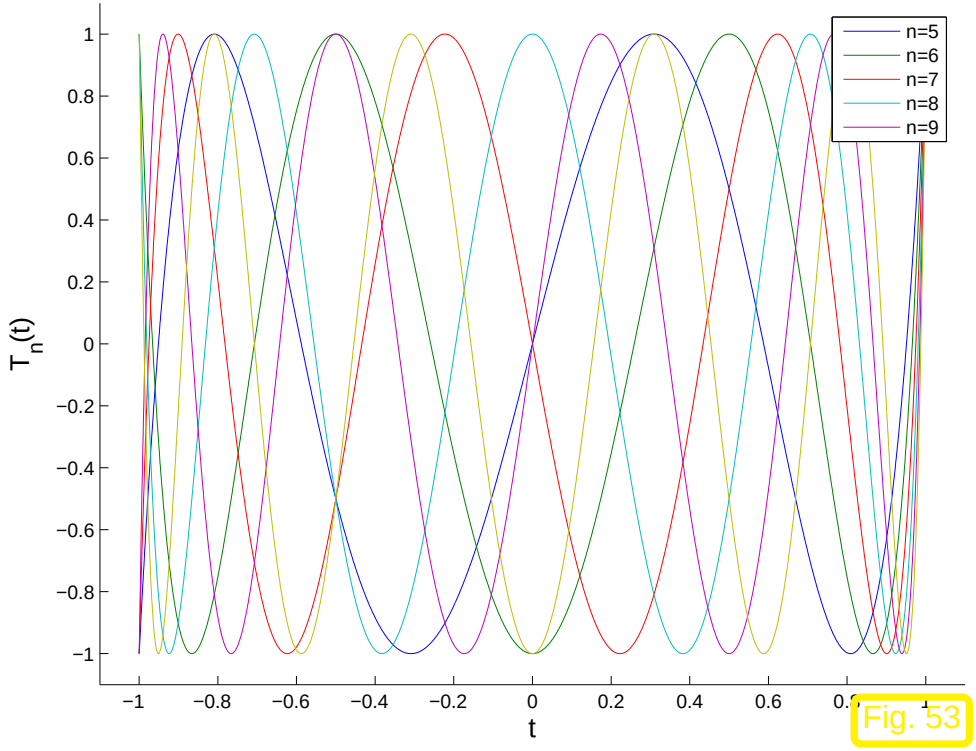


Fig. 53

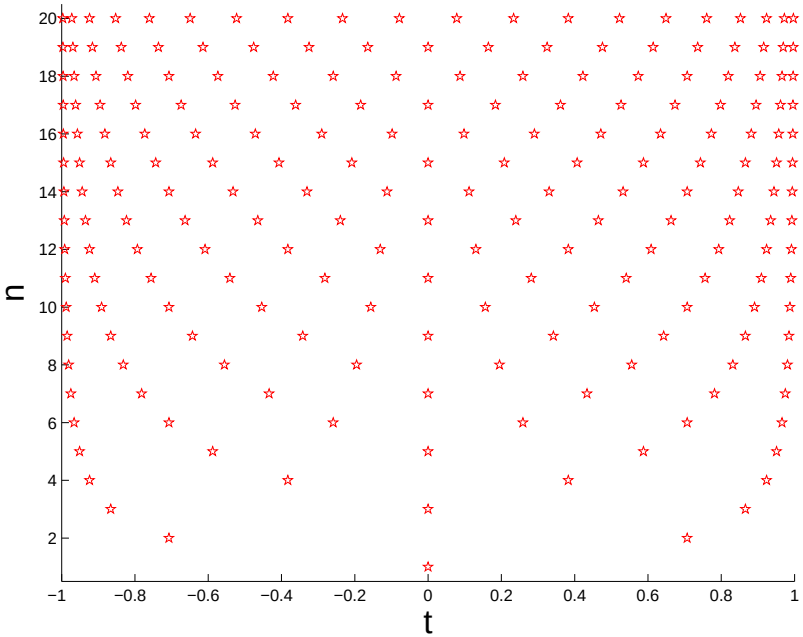
Chebychev polynomials $T_5, \dots, T_9$

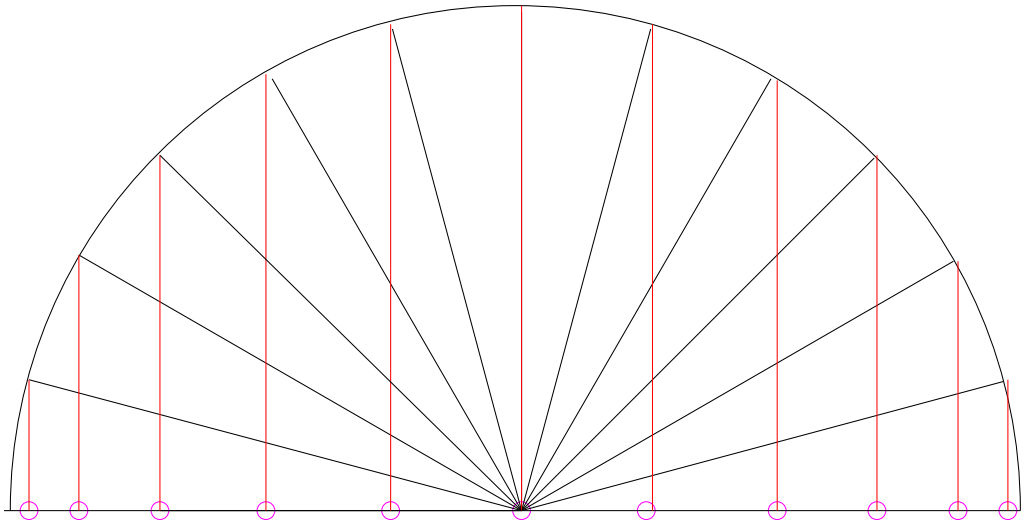
Zeros of T_n: $$t_k = \cos\left(\frac{2k-1}{2n}\pi\right) \;, \quad k = 1, \dots, n \;. \tag{5.4.1}$$

Extrema (alternating signs) of T_n:

$$|T_n(\bar{t}_k)| = 1 \Leftrightarrow \exists\, k = 0, \dots, n\colon \quad \bar{t}_k = \cos\frac{k\pi}{n} \;, \qquad \|T_n\|_{L^\infty([-1,1])} = 1 \;.$$

Chebychev nodes t_k from (5.4.1):





Remark 5.4.1 (3-term recursion for Chebychev polynomial).

3-term recursion by $\cos(n+1)x = 2\cos nx \cos x - \cos(n-1)x$ with $\cos x = t$:

$$T_{n+1}(t) = 2t\,T_n(t) - T_{n-1}(t) \quad , \quad T_0 \equiv 1 \; , \quad T_1(t) = t \; , \quad n \in \mathbb{N} \; . \tag{5.4.2}$$

This implies:
- $T_n \in \mathcal{P}_n$,
- leading coefficients equal to 2^{n-1},
- T_n linearly independent,
- T_n basis of $\mathcal{P}_n = \mathrm{Span}\,\{T_0, \ldots, T_n\}$, $n \in \mathbb{N}_0$.

$\triangle$

Theorem 5.4.2 (Minimax property of the Chebychev polynomials)**.**

$$\|T_n\|_{L^\infty([-1,1])} = \inf\{\|p\|_{L^\infty([-1,1])} : p \in \mathcal{P}_n, p(t) = 2^{n-1}t^n + \cdots\}\ , \quad \forall n \in \mathbb{N}\ .$$

Proof. See [14, Section 7.1.4.] (indirect) Assume

$$\exists q \in \mathcal{P}_n\ , \text{ leading coefficient } = 2^{n-1}: \quad \|q\|_{L^\infty([-1,1])} < \|T_n\|_{L^\infty([-1,1])}\ .$$

$(T_n - q)(x) > 0$ in local maxima of T_n
$(T_n - q)(x) < 0$ in local minima of T_n

From knowledge of local extrema of T_n, see (**??**):

$$T_n - q \text{ changes sign at least } n+1 \text{ times}$$

$$\Rightarrow \qquad T_n - q \text{ has at least } n \text{ zeros}$$

$$T_n - q \equiv 0, \text{ because } T_n - q \in \mathcal{P}_{n-1} \text{ (same leading coefficient!)}$$

□

Application to approximation by polynomial interpolation:

For $I = [-1,1]$

- "optimal" interpolation nodes $\mathcal{T} = \left\{ \cos\left(\frac{2k+1}{2(n+1)}\pi\right) \;, k = 0,\dots,n \right\}$,
- $w(t) = (t-t_0)\cdots(t-t_{n+1}) = 2^{-n}T_{n+1}(t) \quad , \quad \|w\|_{L^\infty(I)} = 2^{-n}$,

  with leading coefficient 1.

Then, by Thm. 5.3.1,

$$\|f - \mathsf{I}_{\mathcal{T}}(f)\|_{L^\infty([-1,1])} \leq \frac{2^{-n}}{(n+1)!} \left\| f^{(n+1)} \right\|_{L^\infty([-1,1])} \; . \tag{5.4.3}$$

Remark 5.4.2 (Chebychev polynomials on arbitrary interval).

How to use Chebychev polynomial interpolation on an arbitrary interval?

Scaling argument: interval transformation requires the transport of the functions

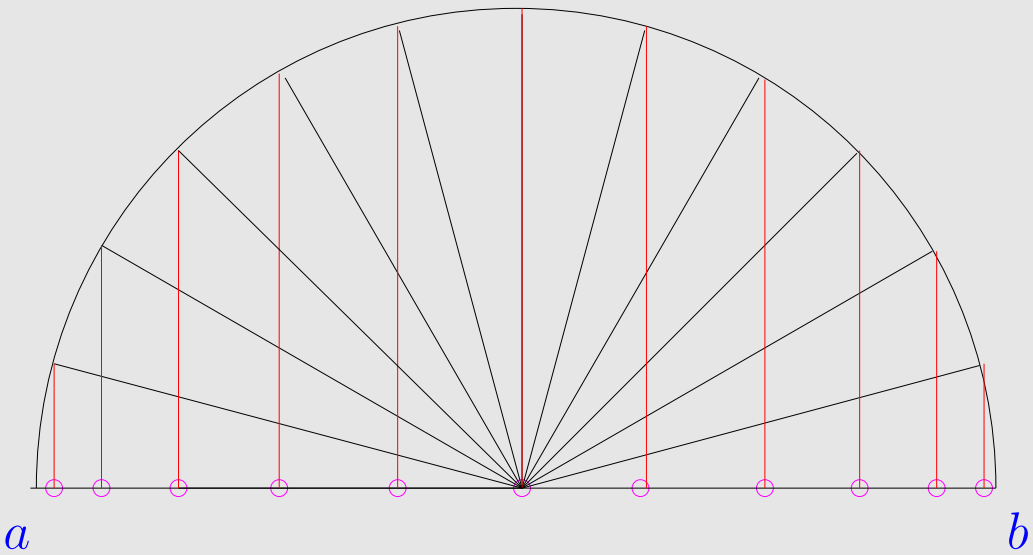
$$[-1,1] \xrightarrow{\widehat{t} \mapsto t := a + \frac{1}{2}(\widehat{t}+1)(b-a)} [a,b] \quad \leftrightarrow \quad \widehat{f}(\widehat{t}) := f(t) \; .$$

$$p \in \mathcal{P}_n \quad \wedge \quad p(t_j) = f(t_j) \quad \Leftrightarrow \quad \widehat{p} \in \mathcal{P}_n \quad \wedge \quad \widehat{p}(\widehat{t}_j) = \widehat{f}(\widehat{t}_j) \; .$$

$$\frac{\mathrm{d}^n \widehat{f}}{\mathrm{d}\widehat{t}^n}(\widehat{t}) = (\tfrac{1}{2}|I|)^n \, \frac{\mathrm{d}^n f}{\mathrm{d}t^n}(t)$$

$$\|f - \mathsf{I}_{\mathcal{T}}(f)\|_{L^\infty(I)} = \left\| \widehat{f} - \mathsf{I}_{\widehat{\mathcal{T}}}(\widehat{f}) \right\|_{L^\infty([-1,1])} \leq \frac{2^{-n}}{(n+1)!} \left\| \frac{\mathrm{d}^{n+1}\widehat{f}}{\mathrm{d}\widehat{t}^{n+1}} \right\|_{L^\infty([-1,1])}$$

$$\leq \frac{2^{-2n-1}}{(n+1)!} |I|^{n+1} \left\| f^{(n+1)} \right\|_{L^\infty(I)} \; . \tag{5.4.4}$$



The Chebychev nodes in the interval $I = [a,b]$ are

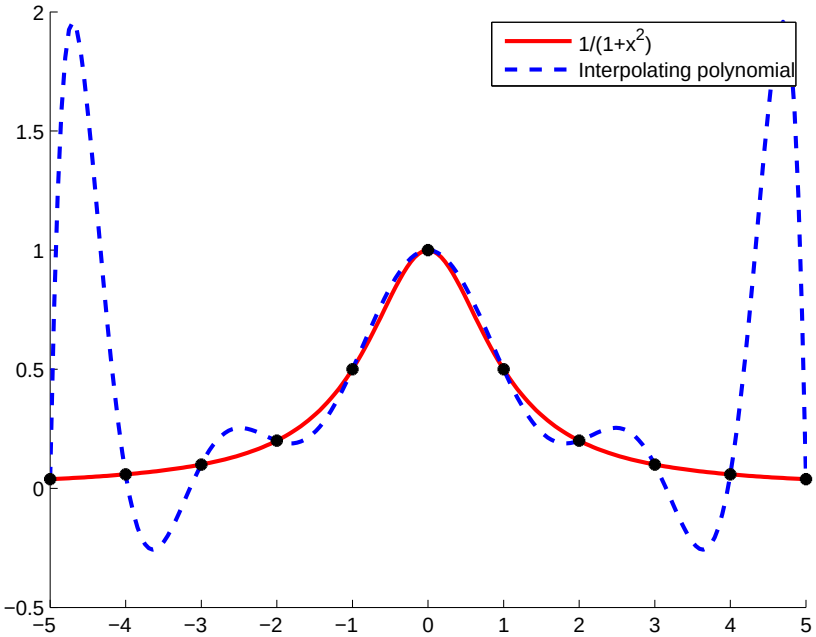
$$t_k := a + \tfrac{1}{2}(b-a)\left(\cos\left(\frac{2k+1}{2(n+1)}\pi\right) + 1\right) \; , \tag{5.4.5}$$

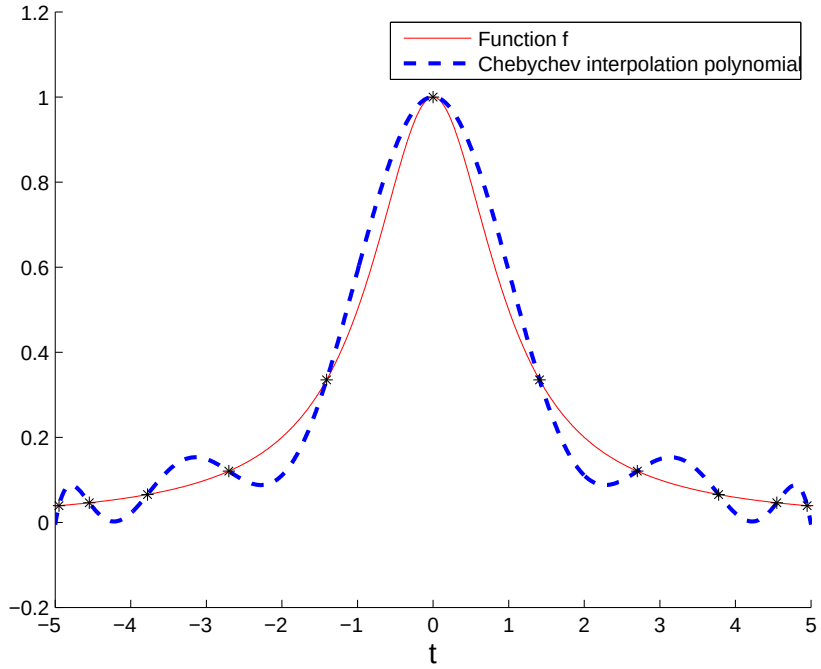
$$k = 0, \ldots, n \; .$$

5.4.2 Chebychev interpolation error estimates

Example 5.4.3 (Polynomial interpolation: Chebychev nodes versus equidistant nodes).

Runge's function $f(t) = \frac{1}{1+t^2}$, see Ex. 5.3.4, polynomial interpolation based on uniformly spaced nodes and Chebychev nodes:





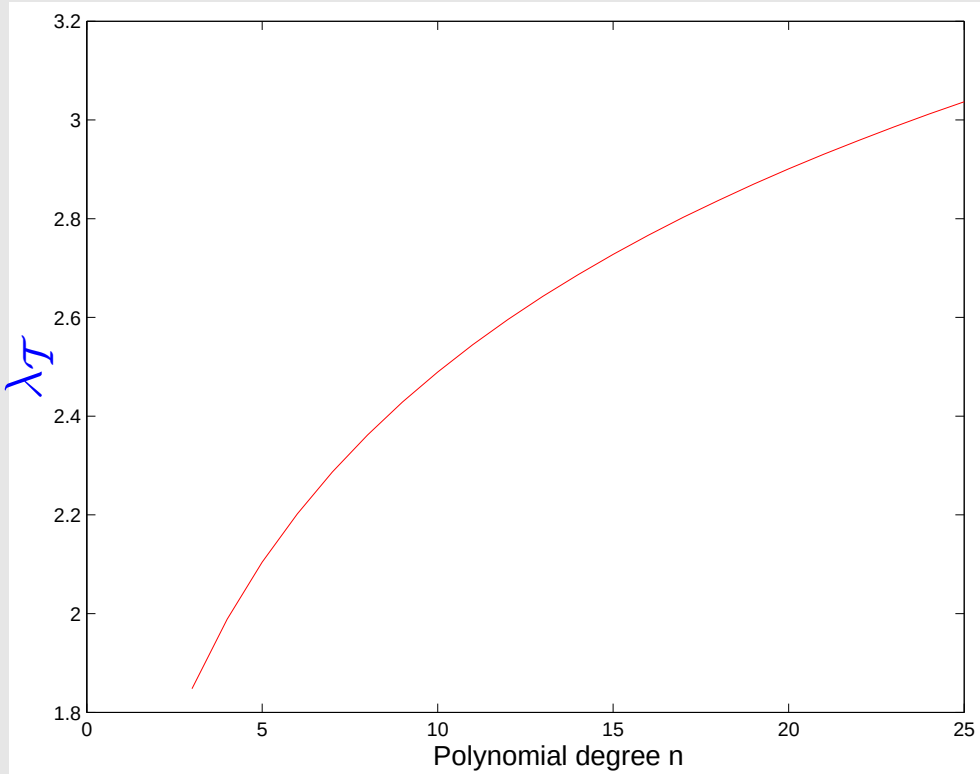
◇

Remark 5.4.4 (Lebesgue Constant for Chebychev nodes).

Theory [7, 56, 55]:

$$\lambda_{\mathcal{T}} \sim \frac{2}{\pi}\log(1+n) + o(1) \ ,$$

$$\boxed{\lambda_{\mathcal{T}} \leq \tfrac{2}{\pi}\log(1+n) + 1} \ . \qquad (5.4.6)$$



△

Example 5.4.5 (Chebychev interpolation error).

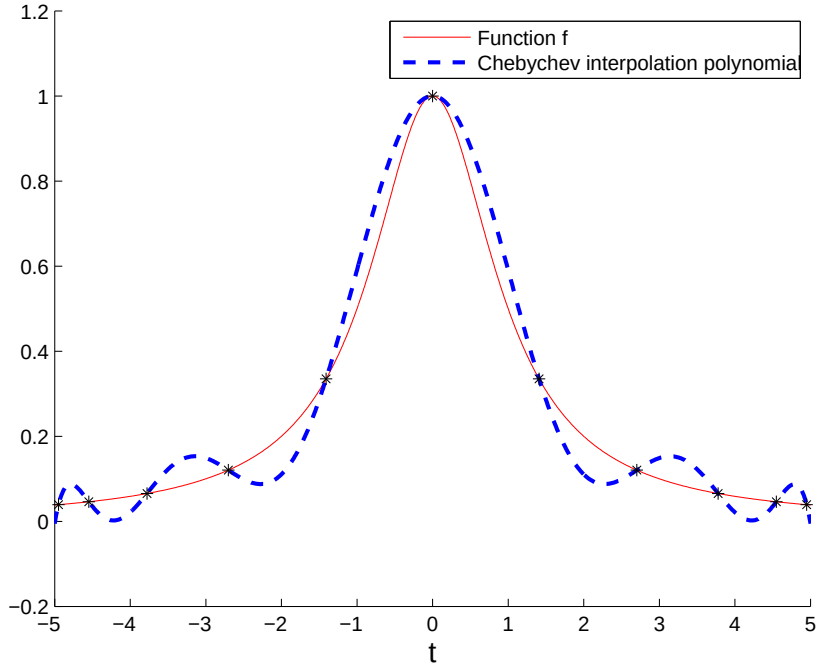
For $I = [a,b]$ let $x_l := a + \frac{b-a}{N} l,\ l = 0, \ldots, N,\ N = 1000$ we approximate the norms of the error

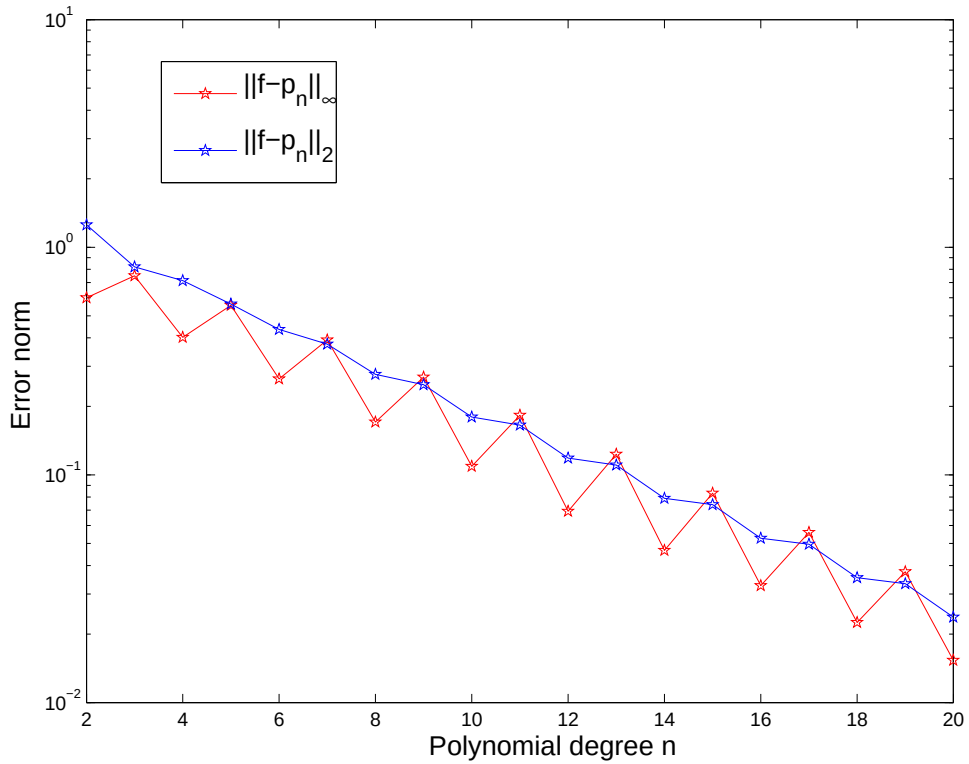
$$||f - p||_\infty \approx \max_{0 \leq l \leq N} |f(x_l) - p(x_l)|$$

$$||f-p||_2^2 \approx \frac{b-a}{2N} \sum_{0\le l<N} \left(|f(x_l)-p(x_l)|^2 + |f(x_{l+1})-p(x_{l+1})|^2 \right)$$

① $f(t) = (1+t^2)^{-1}$, $I = [-5,5]$ (see Ex. 5.3.4)

Interpolation with $n = 10$ Chebychev nodes (plot on the left).





Notice: exponential convergence of the Chebychev interpolation:

$$p_n \to f, \quad \|f - \mathsf{I}_n f\|_{L^2([-5,5])} \approx 0.8^n$$

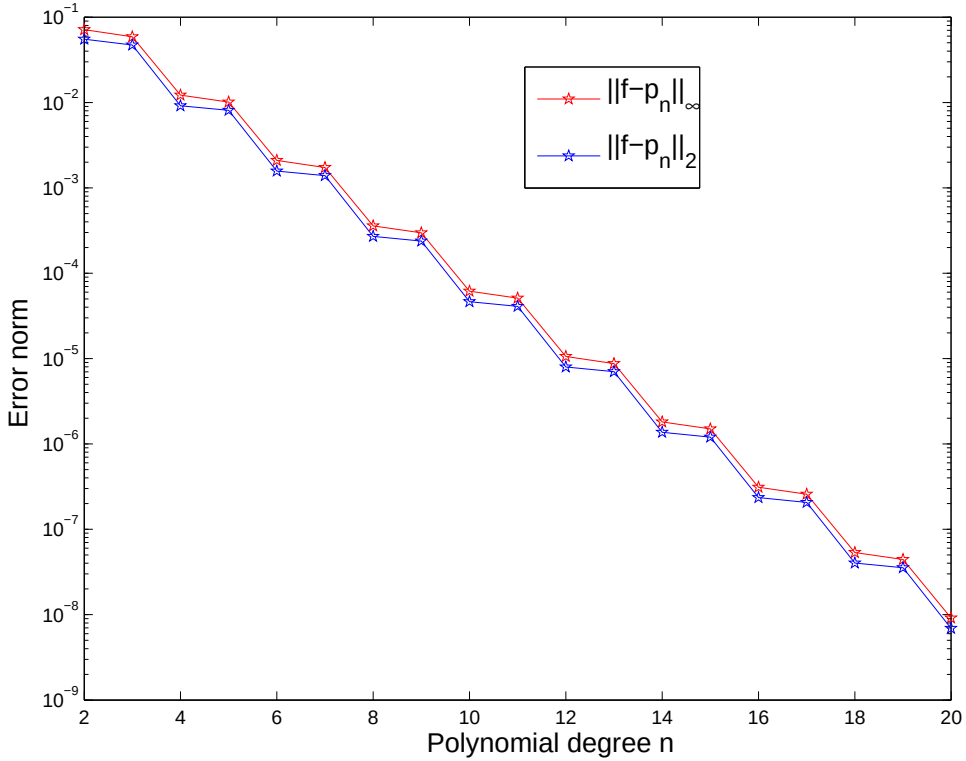
.

Now: the same function $f(t) = (1+t^2)^{-1}$

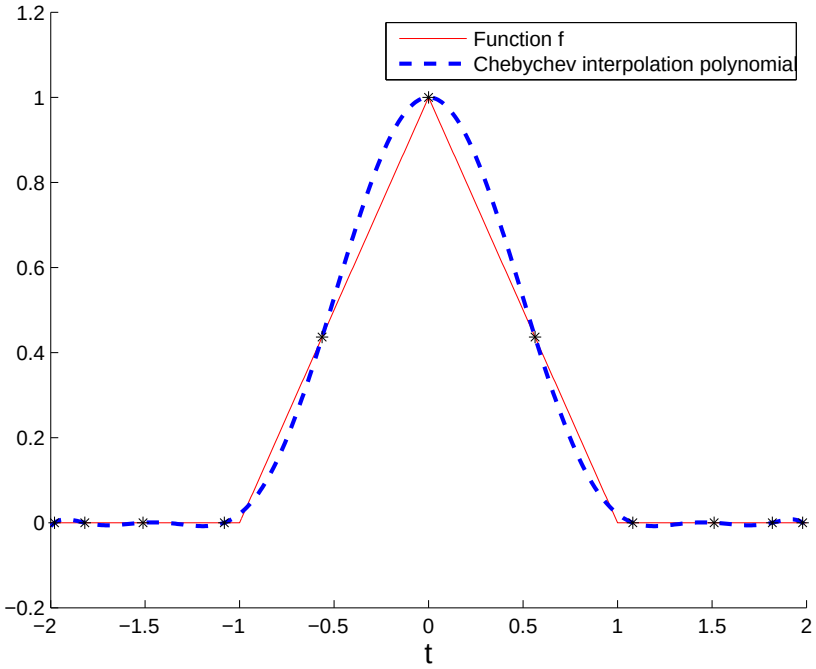
on a smaller interval $I = [-1, 1]$.

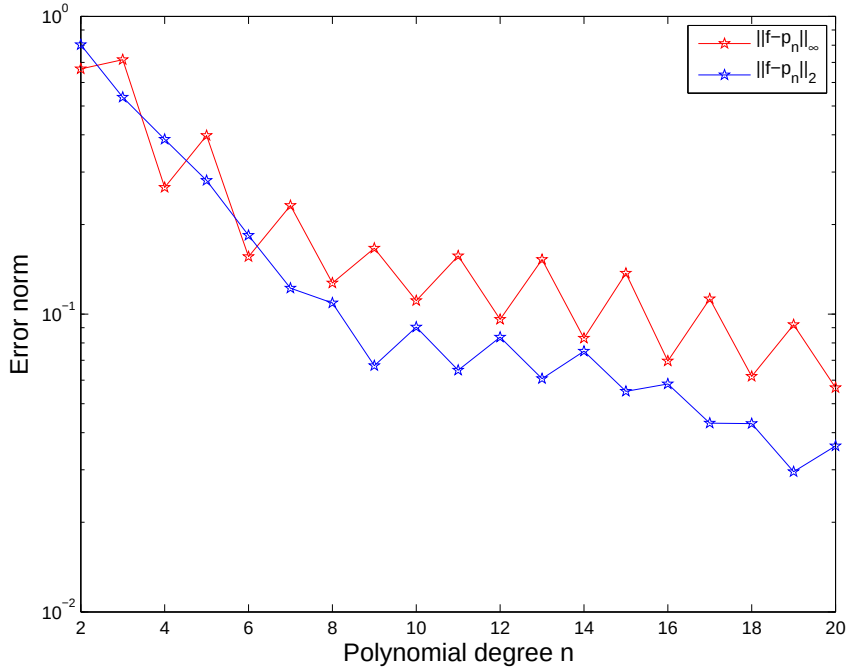
(Faster) exponential convergence:

$$\|f - \mathsf{I}_n f\|_{L^2([-1,1])} \approx 0.42^n.$$



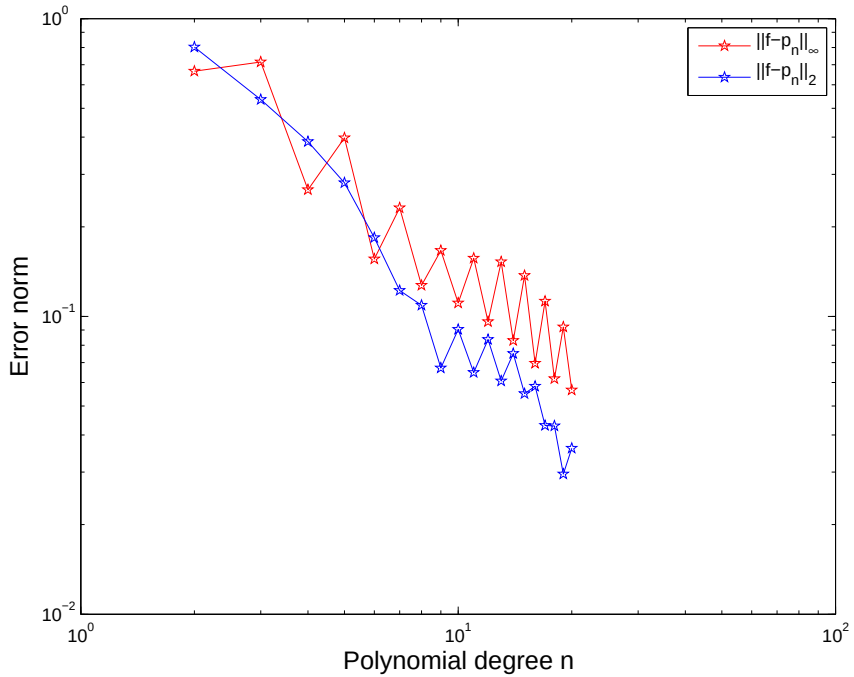
② $f(t) = \max\{1 - |t|, 0\}$, $I = [-2, 2]$, $n = 10$ nodes (plot on the left).
$f \in C^0(I)$ but $f \notin C^1(I)$.



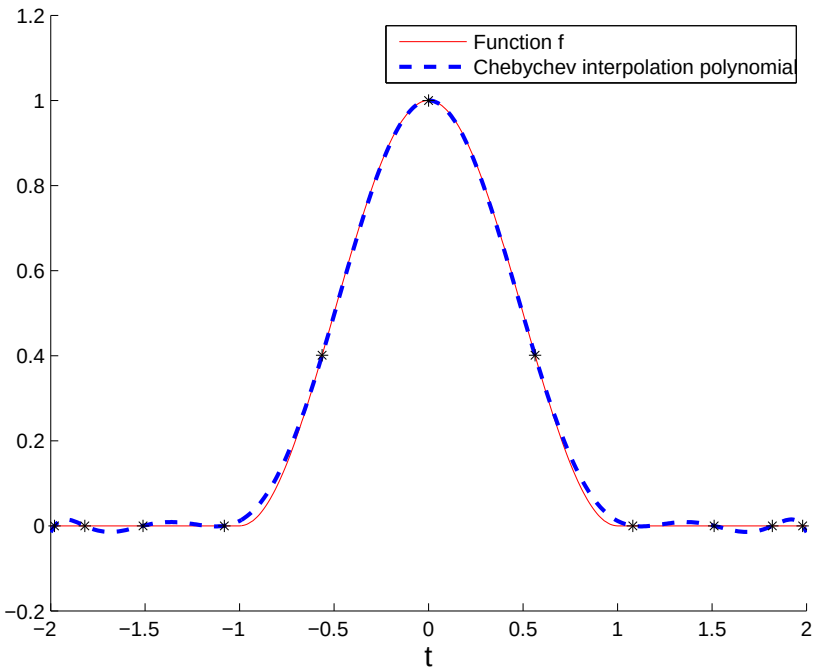


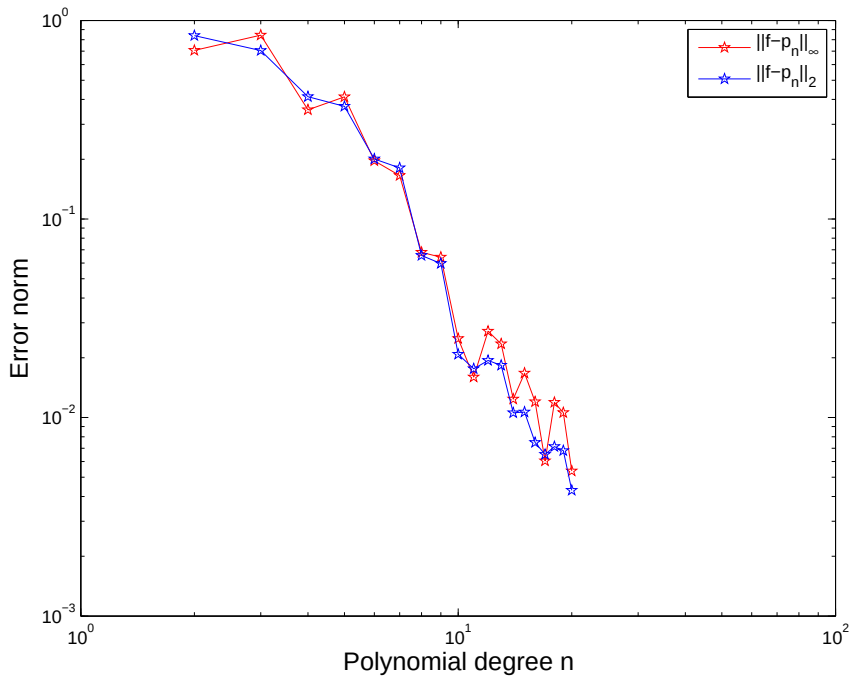
From the double logarithmic plot, notice →

- no exponential convergence
- algebraic convergence (**?**)



③ $f(t) = \begin{cases} \frac{1}{2}(1+\cos \pi t) & |t| < 1 \\ 0 & 1 \le |t| \le 2 \end{cases}$ $\quad I = [-2,2], \quad n = 10$ (plot on the left).





Notice: only algebraic convergence.

◇

Summary of observations, *cf.* Rem. 5.3.3:

- Essential role of smoothness of f: slow convergence of approximation error of the Cheychev interpolant if f enjoys little smoothness, *cf.* also (5.3.3),
- for smooth $f \in C^\infty$ approximation error of the Cheychev interpolant seems to decay to zero exponentially in the polynomial degree n.

5.4.3 Chebychev interpolation: computational aspects

Theorem 5.4.3 (Orthogonality of Chebychev polynomials)**.**
The Chebychev polynomials are orthogonal with respect to the scalar product

$$\langle f, g \rangle = \int_{-1}^{1} f(x)g(x)\frac{1}{\sqrt{1-x^2}}dx \ . \tag{5.4.7}$$

Theorem 5.4.4 (Discrete orthogonality of Chebychev polynomials)**.**
The Chebychev polynomials $T_0, \ldots, T_n$ are orthogonal in the space $\mathcal{P}_n$ with respect to the scalar product:

$$(f, g) = \sum_{k=0}^{n} f(x_k) g(x_k) \ , \tag{5.4.8}$$

where $x_0, \ldots, x_n$ are the zeros of T_{n+1}.

① Computation of the coefficients of the interpolation polynomial in Chebychev form:

Theorem 5.4.5 (Representation formula)**.**
The interpolation polynomial p of f in the Chebychev nodes $x_0, \ldots, x_n$ (the zeros of T_{n+1}) is given by:

$$p(x) = \frac{1}{2} c_0 + c_1 T_1(x) + \ldots + c_n T_n(x) \ , \tag{5.4.9}$$

with

$$c_k = \frac{2}{n+1} \sum_{l=0}^{n} f\left(\cos\left(\frac{2l+1}{n+1} \cdot \frac{\pi}{2}\right)\right) \cdot \cos\left(k \frac{2l+1}{n+1} \cdot \frac{\pi}{2}\right) \ . \tag{5.4.10}$$

For sufficiently large n ($n \geq 15$) it is convenient to compute the c_k with the FFT; the direct computation of the coefficients needs $(n+1)^2$ multiplications, while FFT needs only $O(n \log n)$.

② Evaluation of polynomials in Chebychev form:

Theorem 5.4.6 (Clenshaw algorithm)**.**
Let $p \in \mathcal{P}_n$ be an arbitrary polynomial,

$$p(x) = \frac{1}{2}c_0 + c_1 T_1(x) + \ldots + c_n T_n(x) \ .$$

Set

$$\begin{aligned} d_{n+2} &= d_{n+1} = 0 \\ d_k &= c_k + (2x) \cdot d_{k+1} - d_{k+2} \qquad \text{for } k = n, n-1, \ldots, 0. \end{aligned} \tag{5.4.11}$$

Then $\quad p(x) = \frac{1}{2}(d_0 - d_2)$.

Code 5.4.6: Clenshaw algorithm

```
def clenshaw(a,x):
    # Degree of polynomial
    n = a.shape[0] - 1
    d = tile( reshape(a,n+1,1), (x.shape[0], 1) )
    d = d.T
    for j in xrange(n, 1, -1):
        d[j-1,:] = d[j-1,:] + 2.0*x*d[j,:]
        d[j-2,:] = d[j-2,:] - d[j,:]
    y = d[0,:] + x*d[1,:]
    return y
```

While using recursion it is important how the error (e.g. rounding error) propagates.

Simple example:

$$x_{n+1} = 10x_n - 9,$$
$$x_0 = 1 \Rightarrow x_n = 1 \quad \forall n$$
$$x_0 = 1 + \epsilon \Rightarrow \widetilde{x}_n = 1 + 10^n \epsilon.$$

This is not a problem here: Clenshaw algorithm is stable.

Theorem 5.4.7 (Stability of Clenshaw algorithm)**.**
Consider the perturbed Clenshaw algorithm recursion:

$$\widetilde{d}_k = c_k + 2x \cdot \widetilde{d}_{k+1} - \widetilde{d}_{k+2} + \epsilon_k\,, \quad k = n, n-1, \dots, 0$$

with $\widetilde{d}_{n+2} = \widetilde{d}_{n+1} = 0$. *Set* $\widetilde{p}(x) = \frac{1}{2}(\widetilde{d}_0 - \widetilde{d}_2)$. *Then* $|\widetilde{p}(x) - p(x)| \leq \sum_{j=0}^{n} |\epsilon_j|$ *for* $|x| \leq 1$.

Remark 5.4.7 (Chebychev representation of built-in functions)*.*

Computers use approximation by sums of Chebychev polynomials in the computation of functions like $\log, \exp, \sin, \cos, \dots$. The evaluation through Clenshaw algorithm is much more efficient than with Taylor approximation.

△

5.5 Essential Skills Learned in Chapter 5

You should know:

- what are the divided differences and thier use for polynomial interpolation
- what algebraic/exponential convergence means
- error behavior for polynomial interpolation on equally spaced points
- reasons for uing rather Chebychev interpolation
- definition and properties of the Chebychev polynomials
- error behavior for Chebychev interpolation
- how to compute with Chebychev polynomials