Numerical Analysis: Approximation of Functions

Mirko Navara
http://cmp.felk.cvut.cz/˜navara/
Center for Machine Perception, Department of Cybernetics, FEE, CTU
Karlovo náměstí, building G, office 104a

http://math.feld.cvut.cz/nemecek/nummet.html

November 12, 2014

Requirements for the assessment

- Adequate presence at seminars (you may install Maple on your computer* and do the tasks elsewhere).

* Contact Aleš Němeček for the license.

- Tasks 6×5 points

18 points suffice

References

[Navara, Němeček] Navara, M., Němeček, A.: *Numerical Methods* (in Czech). 2nd edition, CTU, Prague, 2005.

[Knuth] Knuth, D.E.: *Fundamental Algorithms.* Vol. 1 of *The Art of Computer Programming,* 3rd ed., Addison-Wesley, Reading, MA, 1997.

[KJD] Kubíček, M., Janovská, D., Dubcová, M.: *Numerical Methods and Algorithms.* Institute of Chemical Technology, Prague, 2005. `http://old.vscht.cz/mat/NM-Ang/NM-Ang.pdf`

[Num. Recipes] Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: *Numerical Recipes (The Art of Scientific Computing).* 2nd edition, Cambridge University Press, Cambridge, 1992. `http://www.nrbook.com/a/bookcpdf.php`

[Stoer, Bulirsch] Stoer, J., Bulirsch, R.: *Introduction to Numerical Analysis.* Springer Verlag, New York, 2002.

[Handbook Lin. Alg.] Hogben, L. (ed.): *Handbook of Linear Algebra.* Chapman & Hall/CRC, Boca Raton/London/New York, 2007.

Motivation

Task: A polynomial can be factorized to the product of linear polynomials,
but only for orders up to 4.
Numerical solution is possible,
but it is imprecise and division leaves a remainder.
Also this is a subject of numerical anaysis.
Task: Linear algebra: "A system of linear equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ has a unique solution $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ if and only if $\det \mathbf{A} \neq 0$."
Programming: "Real numbers cannot be tested for equality!"
Numerical anaysis: "Imprecision of coefficients causes imprecision of outputs, which for $\det \mathbf{A} \to 0$ exceeds all bounds."
This is called an **ill-conditioned task**.
Task: Linear algebra: "$\det \mathbf{A}$ is a sum of products

$$\sum_p (-1)^{\mathrm{sign}(p)} a_{1,p(1)} \cdot a_{2,p(2)} \cdot \dots \cdot a_{n,p(n)} = \sum_p (-1)^{\mathrm{sign}(p)} \prod_{i \le n} a_{i,p(i)} \,,$$

where we sum up over all permutations p of indices $1, \dots, n$."
Numerical anaysis: "This represents appr. $n \cdot n!$ operations."

n	2	3	4	5	10	20	30
no. of operations	4	18	96	600	36 288 000	$4.8 \cdot 10^{19}$	$7.9 \cdot 10^{33}$

Gaussian Elimination has a complexity proportional to n^3.

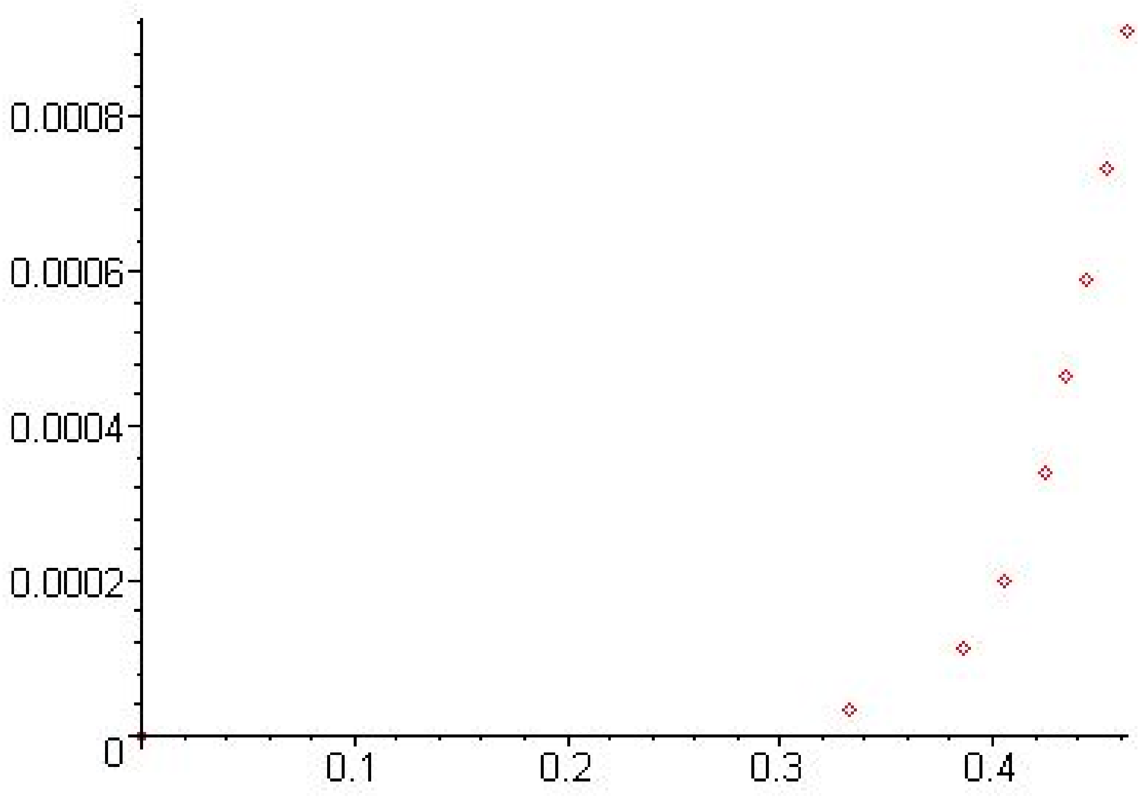
APPROXIMATION OF FUNCTIONS

Task: Estimate future currency exchange rates from the preceding ones.
Task: A random variable with Gaussian normal distribution has a cumulative distribution function

$$\Phi(u) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{u} \exp\left(\frac{-t^2}{2}\right) dt\,.$$

This is a transcendent function.
Numerical integration gives an approximate result with given precision.
(In fact, even the exponential function is computed only numerically, only the 4 basic arithmetical operations are implemented in the processor.)
Faster evaluation can be done with a prepared table of values of Gauss integral.
Task: Estimate the remaining capacity of a battery from its voltage.
We use finitely many values, but we need an approximation on the whole interval.
Moreover, we require monotonicity.

Task: A digital image is zoomed, rotated, etc. This changes the number of pixels and their orientation.
Draw the V-A characteristic of a diode from measured data:



Motivating problems on approximation

1. **Task 1**: Dependence of an exchange rate on time.
2. **Task 2**: Fast estimation of Gauss integral (using a preprocessed table).
3. **Task 3**: V-A characteristic of a diode based on measured values.

4. **Task 4**: Temperatures from a meteorological station.

Basic task of approximation

Given:
n different points $x_0, \ldots, x_{n-1} \in \mathbb{R}$ (**nodes**)
$y_0, \ldots, y_{n-1} \in \mathbb{R}$
set of functions $\mathcal{F}$, defined at least in the nodes $x_0, \ldots, x_{n-1}$
Task:
Find a function $\varphi \in \mathcal{F}$, which minimizes the differences $|y_i - \varphi(x_i)|$, $i = 0, \ldots, n-1$.
Assumption: $\mathcal{F}$ is the linear hull of k *known*, so-called **approximating functions** $\varphi_0, \ldots, \varphi_{k-1}$, $k \le n$:

$$\mathcal{F} = \langle \varphi_0, \ldots, \varphi_{k-1} \rangle = \left\{ \sum_{j<k} c_j \, \varphi_j : c_j \in \mathbb{R} \right\}.$$

It remains to determine real coefficients
c_j, $j = 0, \ldots, k-1$, of the linear combination

$$\varphi = \sum_{j<k} c_j \, \varphi_j \, .$$

Linearity of the procedure of approximation

We mostly assume linear dependence of the output on the inputs ("superposition principle"); this is related to the independence of the output on the chosen scale.
Than any approximation depends linearly on the entries of the arithmetical vector $\vec{y} = (y_0, \ldots, y_{n-1}) \in \mathbb{R}^n$.
The resulting approximation is

$$\vec{\varphi} = \sum_{j<k} c_j \, \vec{\varphi}_j \, ,$$

where
$\vec{\varphi} = (\varphi(x_0), \ldots, \varphi(x_{n-1})) \in \mathbb{R}^n$,
$\vec{\varphi}_j = (\varphi_j(x_0), \ldots, \varphi_j(x_{n-1})) \in \mathbb{R}^n$, $j = 0, \ldots, k-1$.
(No other values are used.)
Vector formulation of approximation:
For a given vector $\vec{y}$, we look for the "nearest" vector $\vec{\varphi}$ from the linear hull of **known** vectors $\vec{\varphi}_j$, $j = 0, \ldots, k-1$.
Vector $\vec{y}$ has coordinates $(y_0, \ldots, y_{n-1})$ w.r.t. the **standard basis**

$$\vec{\psi}_0 = (1, 0, \ldots, 0), \quad \vec{\psi}_1 = (0, 1, 0, \ldots, 0), \quad \ldots \quad , \quad \vec{\psi}_{n-1} = (0, \ldots, 0, 1) \, ,$$

1. For $j = 0, \ldots, n-1$, we can put $\vec{y} = \vec{\psi}_j$ and find approximating functions ϱ_j, represented by vectors $\vec{\varrho}_j$.
2. For a general input vector $\vec{y}$, the approximation is a linear combination of results of the special cases:

$$\varphi = \sum_{j<n} y_j \, \varrho_j \, .$$

Functions ϱ_j, $j = 0, \ldots, n-1$, give sufficient information about the solution for any $\vec{y} = (y_0, \ldots, y_{n-1})$.

Interpolation

Special case of approximation, where we require exact match at nodes.
Given:
n different nodes $x_0, \ldots, x_{n-1} \in \mathbb{R}$
$y_0, \ldots, y_{n-1} \in \mathbb{R}$
set of functions $\mathcal{F}$, defined at least at the nodes $x_0, \ldots, x_{n-1}$
Task: Select a function $\varphi \in \mathcal{F}$ such that

$$\varphi(x_i) = y_i, \qquad i = 0, \ldots, n-1 \, .$$

Simple interpolation

Assumption:

$$\mathcal{F} = \langle \varphi_0, \ldots, \varphi_{n-1} \rangle = \left\{ \sum_{j<n} c_j\, \varphi_j : c_j \in \mathbb{R} \right\}.$$

Substitution gives

$$\sum_{j<n} c_j\, \varphi_j(x_i) = y_i, \qquad i = 0, \ldots, n-1\,,$$

which is a system of n linear equations for unknowns c_j, $j = 0, \ldots, n-1$.
For uniqueness, we put $k = n$ and we require linear independence of arithmetical vectors $\vec{\varphi}_j = (\varphi_j(x_0), \ldots, \varphi_j(x_{n-1}))$, $j = 0, \ldots, k-1$.
The complexity is proportional to n^3; we shall achieve smaller complexity for special tasks.

Polynomial interpolation

We interpolate by a polynomial of degree less than n, i.e., $\leq n-1$.
It has n coefficients, which is needed for existence and uniqness of the solution.
Interpolating functions can be chosen as $\varphi_j(t) = t^j$, $j = 0, \ldots, n-1$.
It may be advantageous to use another bases of the n-dimensional linear space of all polynomials of degree less than n.

Advantages of polynomial interpolation

- It is easy to integrate them, differentiate, etc.
- Solvability:

  Theorem: There is exactly on polynomial of degree less than n which solves the interpolation task for n nodes.

- Universality:

  Weierstrass Theorem: Let f be a *continuous* function at a *closed* interval I and let $\varepsilon > 0$. Then there exists a polynomial φ such that
  $$\forall t \in I : |f(t) - \varphi(t)| \leq \varepsilon\,.$$

Disadvantages of polynomial interpolation

- Weierstrass Theorem does not say anything about the degree of the polynomial, thus it may be useless.
- Polynomial dependencies are rather rare in practice.

Simple interpolation works, but we shall show better methods.

Lagrange construction of the interpolating polynomial

1. We solve special cases when the jth entry of vectoru $\vec{y}$ is unit, others are zeros; this results in polynomials ϱ_j, $j = 0, \ldots, n-1$,

$$\varrho_j(x_i) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

(δ_{ij} is the **Kronecker delta**)
Polynomial ϱ_j of degree $\leq n-1$ has $n-1$ roots $x_0, \ldots, x_{j-1}, x_{j+1}, \ldots, x_{n-1}$;

$$\begin{aligned} \varrho_j(t) &= e_j(t - x_0) \ldots (t - x_{j-1})(t - x_{j+1}) \ldots (t - x_{n-1}) \\ &= e_j \prod_{i<n, i\neq j} (t - x_i)\,, \end{aligned}$$

where e_j follows from the equality $\varrho_j(x_j) = 1$:

$$e_j = \frac{1}{\prod\limits_{i<n, i\neq j} (x_j - x_i)}\,, \qquad \varrho_j(t) = \frac{\prod\limits_{i<n, i\neq j} (t - x_i)}{\prod\limits_{i<n, i\neq j} (x_j - x_i)}\,.$$

2. The general solution is the linear combination

$$\varphi = \sum_{j<n} y_j\, \varrho_j\,,$$

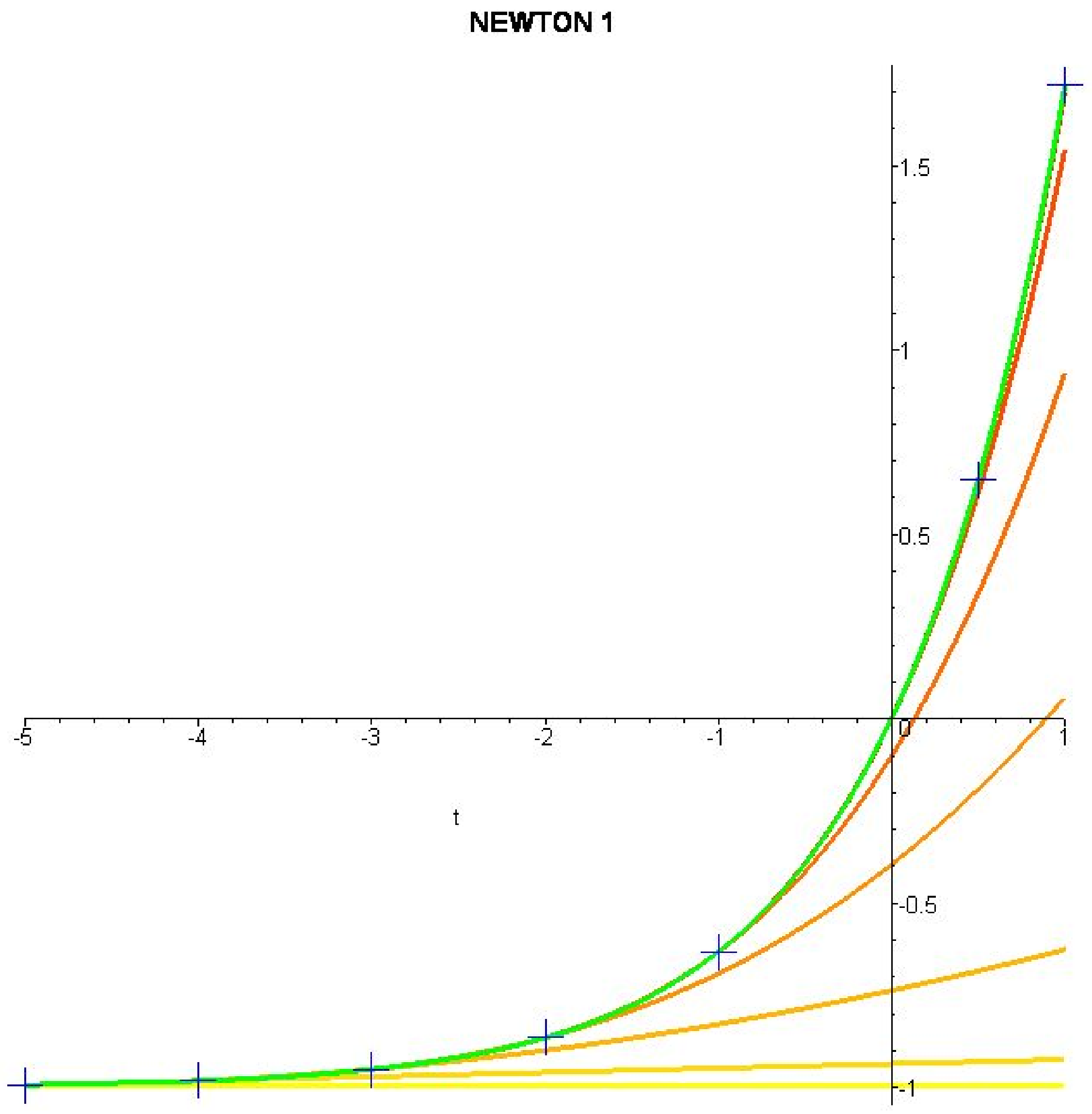
$$\varphi(t) = \sum_{j<n} y_j\, \varrho_j(t) = \sum_{j<n} y_j\, \frac{\prod\limits_{i<n, i\neq j} (t - x_i)}{\prod\limits_{i<n, i\neq j} (x_j - x_i)}\,.$$

Check:

$$\varphi(x_i) = \sum_{j<n} y_j\, \varrho_j(x_i) = \sum_{j<n} y_j\, \delta_{ij} = y_i\,.$$

The complexity is proportional to n^2.
The idea of the Lagrange construction of the interpolating polynomial is applicable to more general tasks.
Motivation for dissatisfaction:
Approximating exchange rates, we continuously receive new data.
Is it necessary to comput all again from scratch?
Some intermediate results can be used, provided that we recorded them.
The previous result can be used and only corrected by certain polynomial.

Newton construction of the interpolating polynomial



It is still the same (unique) polynomial
The constant polynomial $c_0 = y_0$ has the right value at x_0.
The resulting polynomial φ is a sum of an appropriate polynomial ω_0 vanishing at x_0: $t \mapsto (t - x_0)\,\omega_0(t)$, where ω_0 is a polynomial of degree $\leq n-2$:

$$\varphi(t) = c_0 + (t - x_0)\,\omega_0(t)\,, \qquad c_0 = y_0\,, \qquad \omega_0(t) = \frac{\varphi(t) - c_0}{t - x_0}$$

(cancel out). Values at nodes: $\omega_0(x_i) = \dfrac{y_i - c_0}{x_i - x_0}$. Induction:
Polynomial ω_0 is of the form (ω_1 is a polynomial of degree $\leq n-3$):

$$\begin{aligned}
&\omega_0(t) = c_1 + (t - x_1)\,\omega_1(t)\,, && c_1 = \omega_0(x_1)\,, \quad \omega_1(t) = \frac{\omega_0(t) - c_1}{t - x_1}\,,\\
&\omega_1(t) = c_2 + (t - x_2)\,\omega_2(t)\,, && c_2 = \omega_1(x_2)\,, \quad \omega_2(t) = \frac{\omega_1(t) - c_2}{t - x_2}\,,\\
&\omega_2(t) = c_3 + (t - x_3)\,\omega_3(t)\,, && c_3 = \omega_2(x_3)\,, \quad \omega_3(t) = \frac{\omega_2(t) - c_3}{t - x_3}\,,\\
&\dots\\
&\omega_{n-2}(t) = c_{n-1} = \omega_{n-3}(x_{n-2})\,.
\end{aligned}$$

Back substitution gives

$$\begin{aligned}\varphi(t) &= c_0 + (t-x_0)\cdot[c_1 + (t-x_1)\cdot[c_2 + \\ &\quad + \ldots (t-x_{n-3})\cdot[c_{n-2} + (t-x_{n-2})\cdot c_{n-1}]\ldots]]\,.\end{aligned}$$

Expansion (not recommended!):

$$\begin{aligned}\varphi(t) &= c_0 + (t-x_0)\,c_1 + (t-x_0)\,(t-x_1)\,c_2 + \ldots + c_{n-1}\prod_{i<n-1}(t-x_i) \\ &= \sum_{j<n} c_j \prod_{i<j}(t-x_i)\,.\end{aligned}$$

The complexity of the computation of coefficients is $\sim n^2$, then each function value is obtained with the complexity $\sim n$.

Nevill's algorithm

We need to evaluate the interpolating polynomial just for a single argument t; the coefficients are irrelevant.
One point (x_i, y_i) gives an approximation by a constant polynomial y_i.
Two points (x_i, y_i), (x_{i+1}, y_{i+1}) are approximated by a linear polynomial; its value at t is a linear combination of values y_i, y_{i+1}, namely

$$y_{i,i+1} = \frac{(t-x_{i+1})\,y_i + (x_i - t)\,y_{i+1}}{x_i - x_{i+1}}\,,$$

etc.
$y_{i,i+1,\ldots,i+m-1}$ ... the value at t of the interpolating polynomial with m nodes $x_i, x_{i+1}, \ldots, x_{i+m-1}$
$y_{i+1,\ldots,i+m-1,i+m}$ ... the value at t of the interpolating polynomial with m nodes $x_{i+1}, \ldots, x_{i+m-1}, x_{i+m}$
The value at t of the interpolating polynomial with $m+1$ nodes $x_i, x_{i+1}, \ldots, x_{i+m}$ is

$$y_{i,i+1,\ldots,i+m} = \frac{(t-x_{i+m})\,y_{i,i+1,\ldots,i+m-1} + (x_i - t)\,y_{i+1,\ldots,i+m-1,i+m}}{x_i - x_{i+m}}$$

We proceed from values $y_0, y_1, \ldots, y_{n-1}$ to $y_{0,1,\ldots,n-1} = \varphi(t) =$ result.
Numerical errors can be reduced if, instead of the polynomial values, we compute only their differences,

$$\begin{aligned}C_{i,m} &= y_{i,\ldots,i+m} - y_{i,\ldots,i+m-1}\,,\\ D_{i,m} &= y_{i,\ldots,i+m} - y_{i+1,\ldots,i+m}\,,\end{aligned}$$

using recurrent formulas

$$\begin{aligned}C_{i,m} &= \frac{(x_i - t)\,(C_{i+1,m-1} - D_{i,m-1})}{x_i - x_{i+m}}\,,\\ D_{i,m} &= \frac{(x_{i+m} - t)\,(C_{i+1,m-1} - D_{i,m-1})}{x_i - x_{i+m}}\,.\end{aligned}$$

The result is, e.g.,

$$y_{0,1,\ldots,n-1} = y_0 + \sum_{m=1}^{n-1} C_{0,m}\,.$$

Error of approximation by the interpolating polynomial

Zero at nodes (only numerical errors occur).
Error in other points if we approximate a function f by the interpolating polynomial.

Derivation of the error of approximation by the interpolating polynomial

Assumptions:
f has continuous derivatives up to order n at a *closed* interval $I \supseteq \{x_0, \ldots, x_{n-1}\}$,
$u \in I \setminus \{x_0, \ldots, x_{n-1}\}$

We define a function

$$g_u(t) = \underbrace{f(t) - \varphi(t)}_{\text{error of interpolation at } t} - P_u\, W(t)\,,$$

where $P_u = \dfrac{f(u) - \varphi(u)}{W(u)}$ is a suitable constant,
$W(t) = \prod\limits_{i<n} (t - x_i)$ is a polynomial of degree n with roots $x_0, \ldots, x_{n-1}$ and a unit coefficient at the highest power, t^n (these conditions determine the polynomial uniquely),
Its nth derivative is $W^{(n)}(t) = n!$

$$g_u = f - \varphi - P_u\, W$$

has roots $u, x_0, \ldots, x_{n-1}$ and a continuous nth derivative

$$g_u^{(n)} = f^{(n)} - P_u\, n!$$

Between every two roots of g_u, there is a root of its derivative g_u'.
Between $n+1$ roots of g_u, there are
$\geq n$ roots of g_u',
$\geq n-1$ roots of g_u'',
...
≥ 1 root of $g_u^{(n)}$, say ξ.

$$0 = g_u^{(n)}(\xi) = f^{(n)}(\xi) - P_u\, n! = f^{(n)}(\xi) - \frac{f(u) - \varphi(u)}{W(u)}\, n!$$

$$f(u) - \varphi(u) = \frac{f^{(n)}(\xi)}{n!}\, W(u)$$

$$f(u) - \varphi(u) = \frac{f^{(n)}(\xi)}{n!}\, W(u)$$

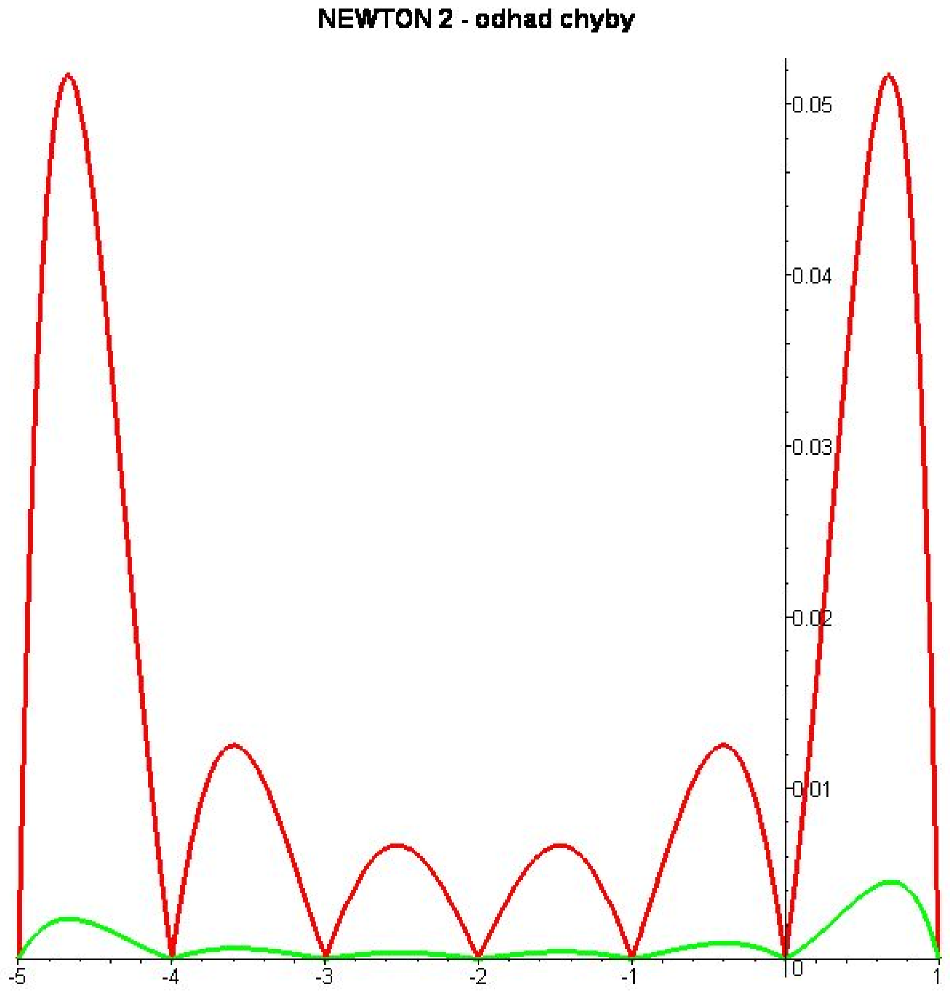
$$|f(u) - \varphi(u)| = \frac{|f^{(n)}(\xi)|}{n!}\, |W(u)|$$

We replace $|f^{(n)}(\xi)|$ by its upper estimate M_n: $\forall t \in I : |f^{(n)}(t)| \leq M_n$

$$|f(u) - \varphi(u)| \leq \frac{M_n}{n!}\, |W(u)|$$

Using an upper estimate $\overline{W}$ of W, $\forall t \in I : |W(t)| \leq \overline{W}$, we get an upper estimate independent of u:

$$|f(u) - \varphi(u)| \leq \frac{M_n}{n!}\, \overline{W}$$



(Red: upper estimate of error; green: real error.)
Two cases:

- **Extrapolation**: approximation outside the interval $I(x_0, \ldots, x_{n-1})$ of nodes
- **Interpolaci (in the narrow sense)**: approximation outside the interval $I(x_0, \ldots, x_{n-1})$

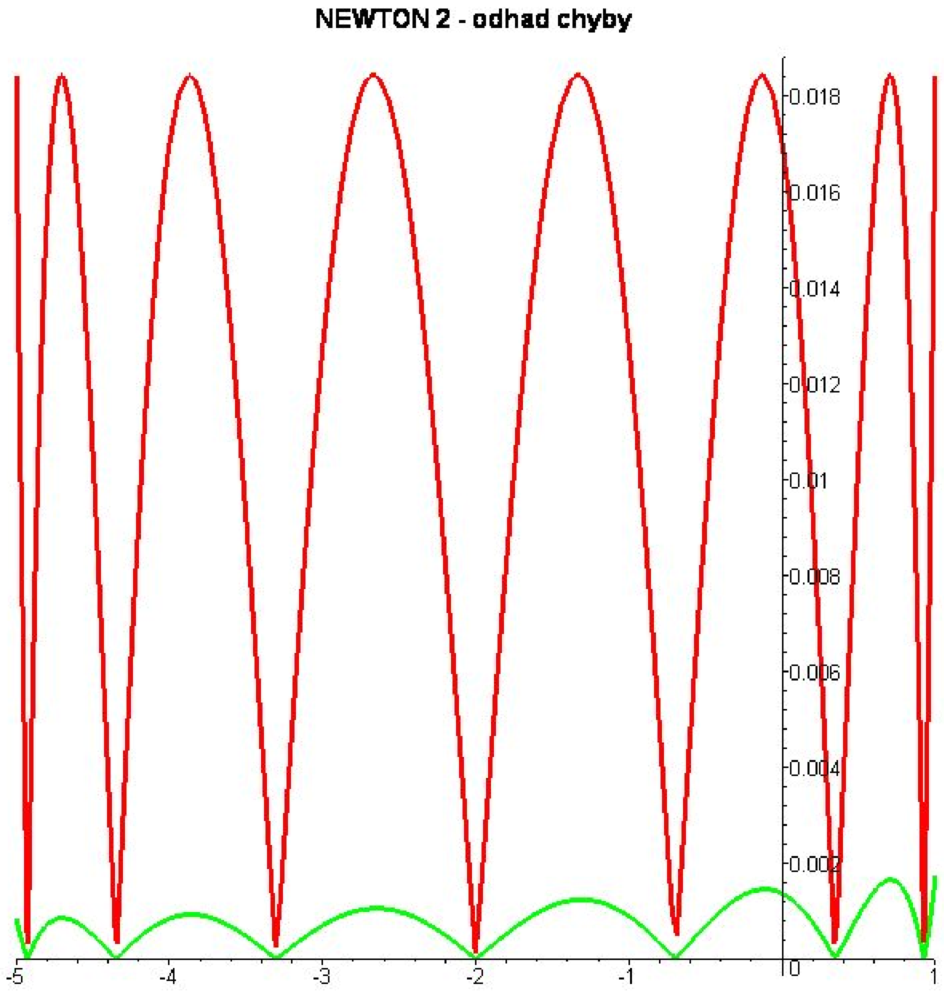
Recommendation for minimization of error: cosine distribution of nodes
On interval $[-1, 1]$, choose nodes

$$z_i = \cos \frac{\pi(i + \frac{1}{2})}{n}, \qquad i = 0, \ldots, n-1\,,$$

on a general interval $[a, b]$ choose

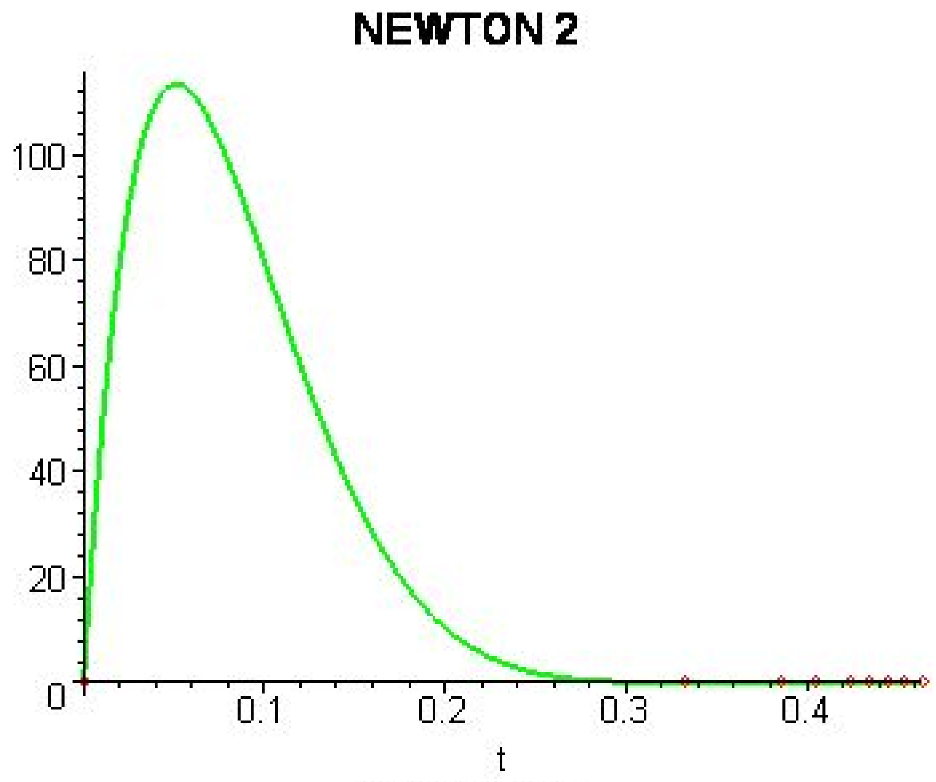
$$x_i = \frac{b+a}{2} + \frac{b-a}{2}\, z_i = \frac{b+a}{2} + \frac{b-a}{2} \cos \frac{\pi(i + \frac{1}{2})}{n}$$

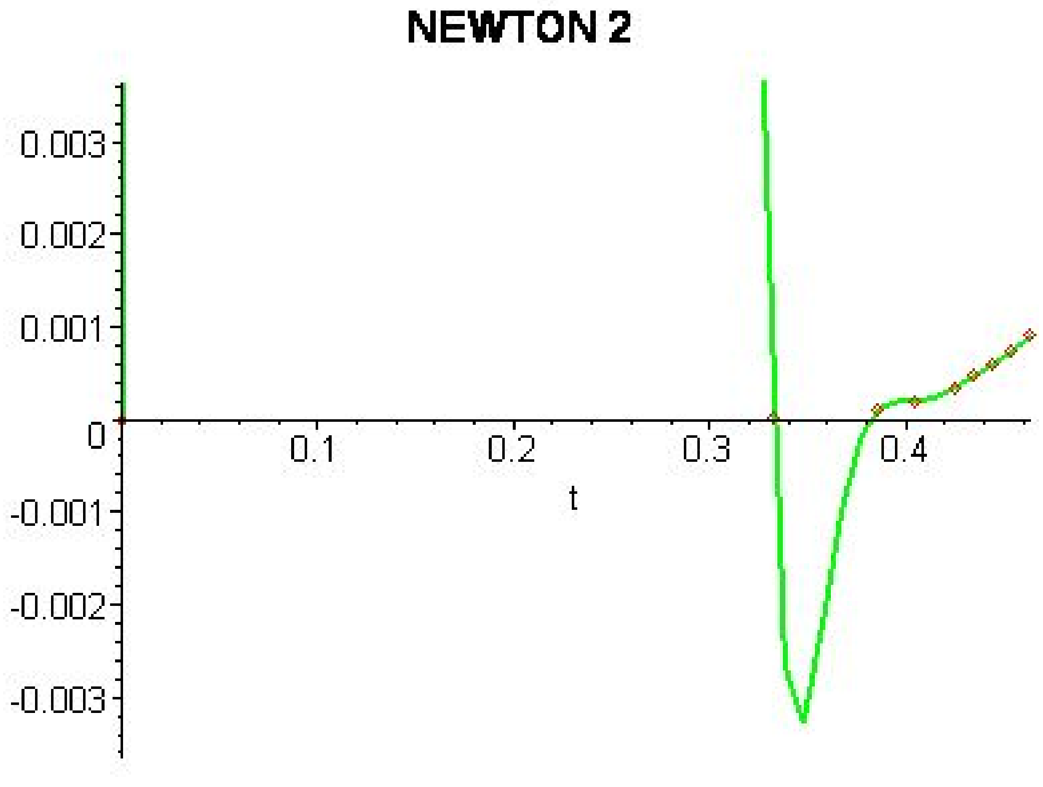
Approximation error of polynomial interpolation for a cosine distribution of nodes



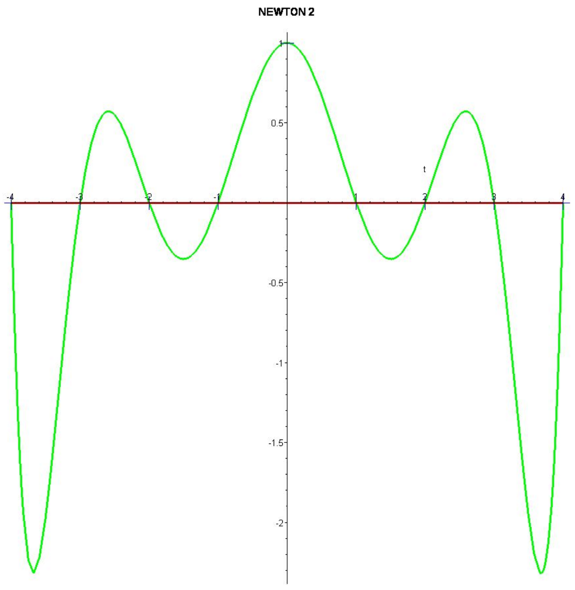
(Red: upper estimate of error; green: real error.)

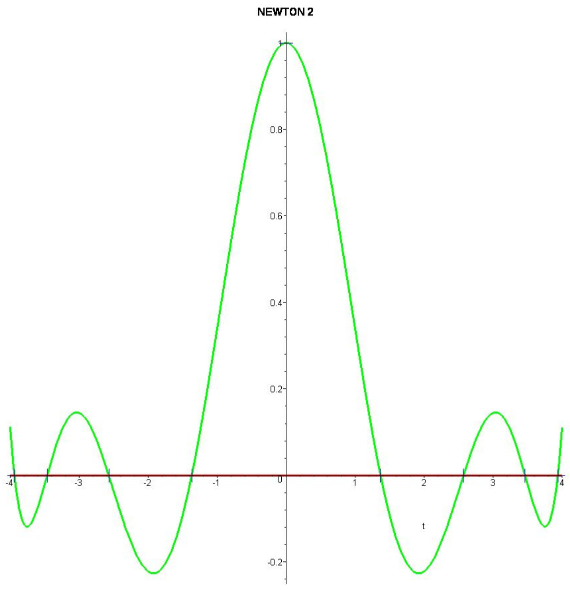
Motivating problem - polynomial interpolation

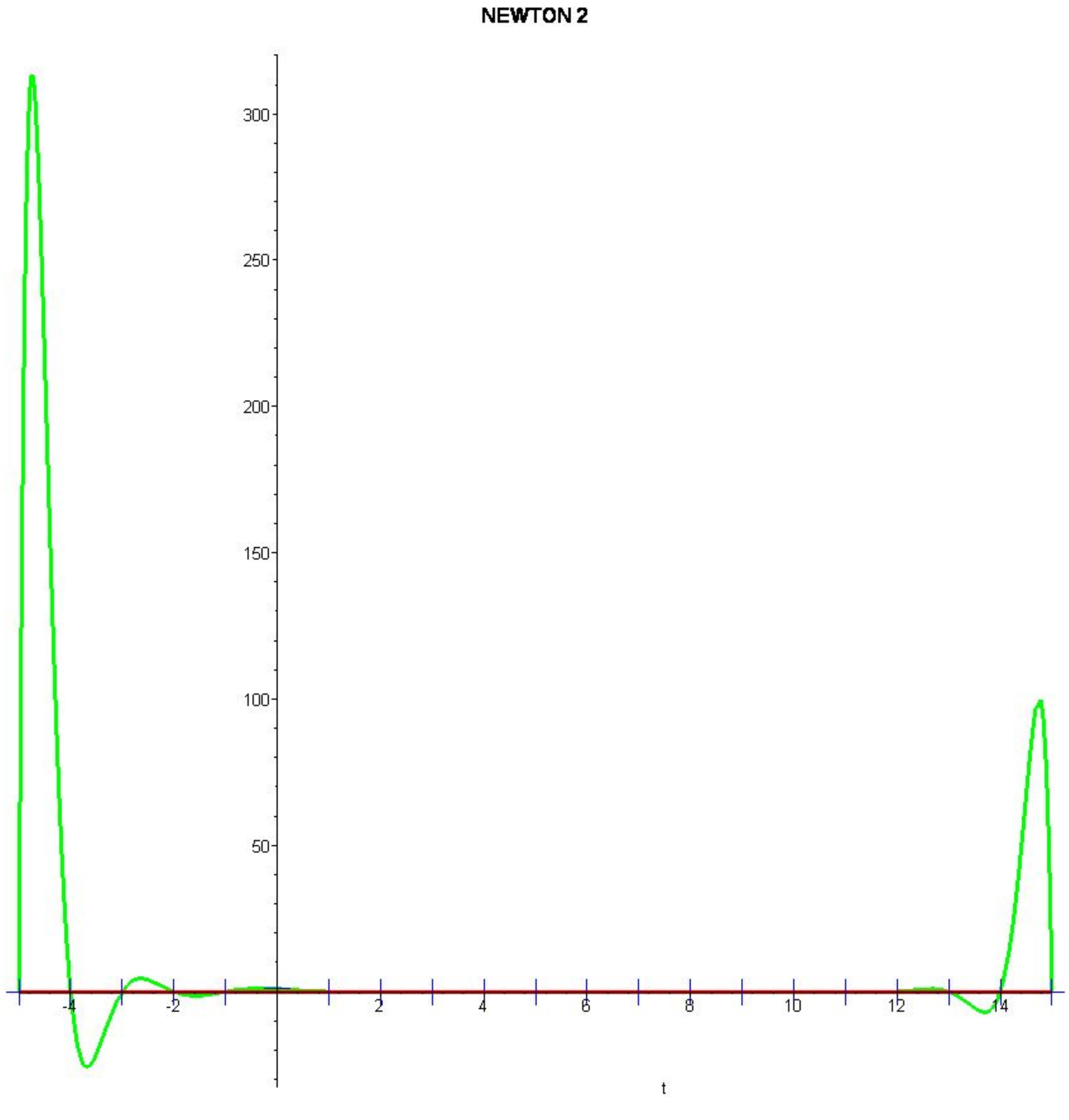


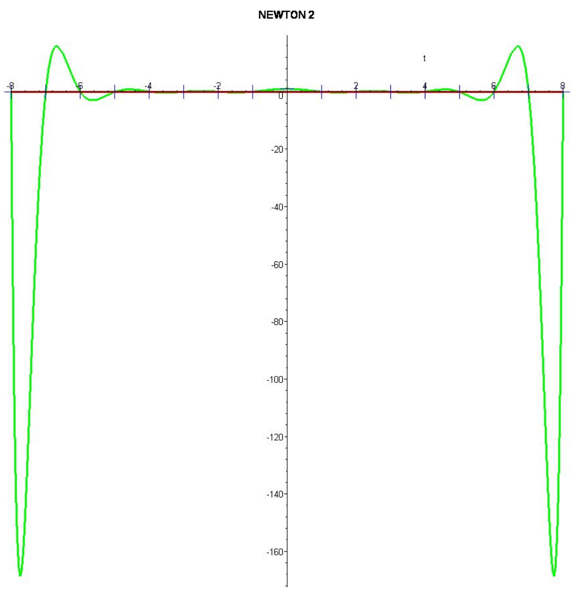


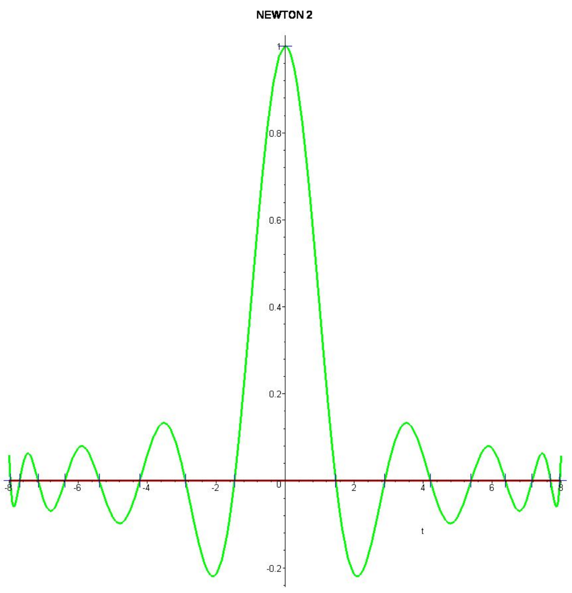
Dependence of interpolating polynomial on local changes


NEWTON 2
t


NEWTON 2
t


NEWTON 2
t





Hermite interpolating polynomial - example

Given:
$x_0 = 0, x_1 = 1,$
$y_{0,0}, y_{1,0}, y_{0,1}, y_{1,1} \in \mathbb{R}$
Task:
Find a polynomial φ of degree at most 3, such that

$$\varphi(x_0) = y_{0,0}\,, \quad \varphi(x_1) = y_{1,0}\,, \quad \varphi'(x_0) = y_{0,1}\,, \quad \varphi'(x_1) = y_{1,1}$$

As in the Lagrange construction of the interpolating polynomial, we first find polynomials $\eta, \varrho, \sigma, \tau$ of degree 3:

ψ	$\psi(0)$	$\psi(1)$	$\psi'(0)$	$\psi'(1)$
η	1	0	0	0
ϱ	0	1	0	0
σ	0	0	1	0
τ	0	0	0	1

Polynomial η has a double root 1, thus it is of the form

$$\eta(t) = (a_0 t + b_0)(t-1)^2\,,$$

where a_0, b_0 can be determined from the values at 0:

$$\begin{aligned} \eta(0) = b_0 &= 1 \\ \eta'(0) = a_0 - 2b_0 &= 0 \end{aligned}$$

$$a_0 = 2, \qquad b_0 = 1, \qquad \eta(t) = (2t+1)(t-1)^2$$

ϱ has a double root 0, thus it is of the form

$$\varrho(t) = (a_1 t + b_1)\, t^2\,,$$

where a_1, b_1 can be determined from the values at 1:

$$\begin{aligned} \varrho(1) = a_1 + b_1 &= 1 \\ \varrho'(1) = 3a_1 + 2b_1 &= 0 \end{aligned}$$

$$a_1 = -2, \qquad b_1 = 3, \qquad \varrho(t) = (-2t+3)t^2$$

σ has a double root 1 and a single root 0, thus it is of the form

$$\sigma(t) = c_0\, t\, (t-1)^2\,,$$

where c_0 can be determined from the derivative at 0:

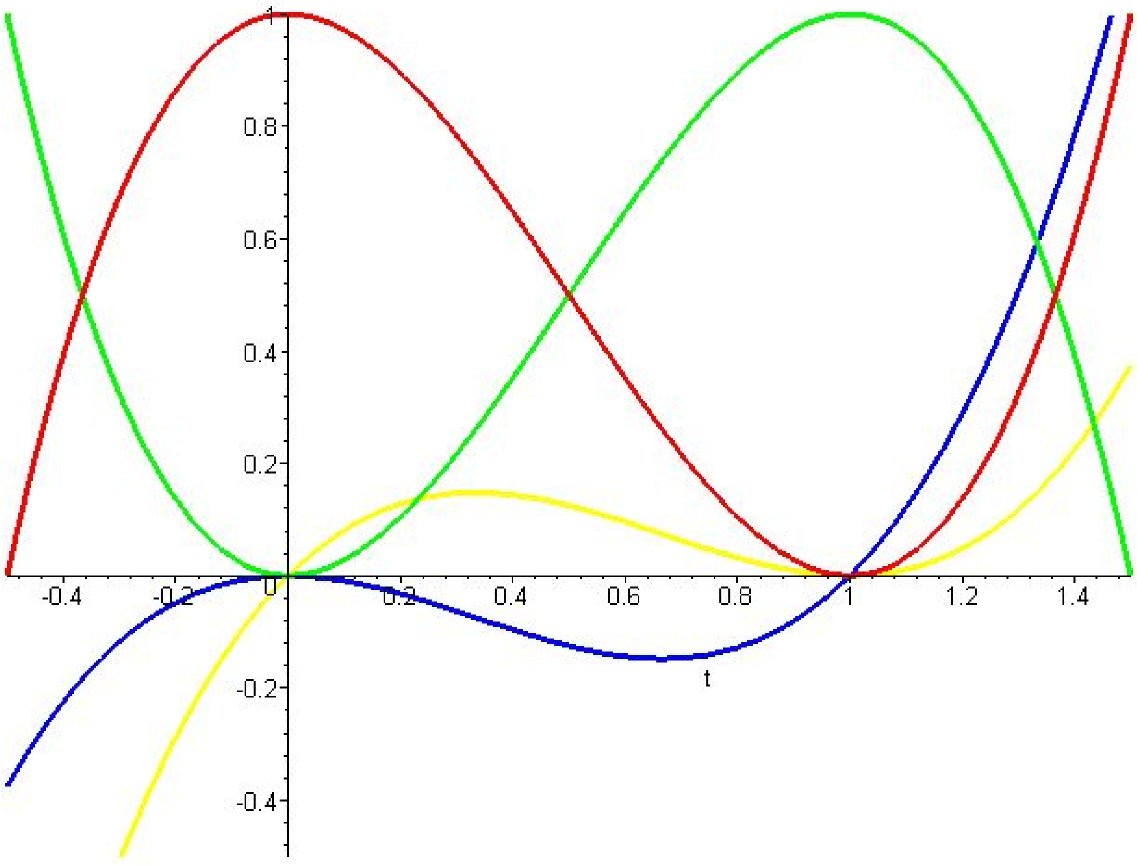
$$\sigma'(0) = c_0 = 1$$
$$\sigma(t) = t(t-1)^2$$

τ has a double root 0 and a single root 1, thus it is of the form

$$\tau(t) = c_1\, t^2 (t-1)\,,$$

where c_1 can be determined from the derivative at 1:

$$\tau'(1) = c_1 = 1$$
$$\tau(t) = t^2(t-1)$$



φ is a linear combination of $\eta, \varrho, \sigma, \tau$,

$$\varphi = y_{0,0}\, \eta + y_{1,0}\, \varrho + y_{0,1}\, \sigma + y_{1,1}\, \tau$$

Approximation by Taylor series (more exactly, by Taylor polynomial)

Special case of Hermite interpolating polynomial with one node, where the first n derivatives (including the 0's) are given.
Given:
Node x_0
n values $y_{0,0}, y_{0,1}, \dots, y_{0,n-1} \in \mathbb{R}$
Task:
Find a polynomial φ of degree less than n, such that

$$\varphi^{(j)}(x_0) = y_{0,j}\,, \qquad j = 0, \dots, n-1\,.$$

The solution is of the form

$$\varphi = \sum_{j<n} y_{0,j}\,\psi_j\,, \qquad \psi_j^{(i)}(x_0) = \delta_{ij}\,, \qquad \psi_j(t) = \frac{1}{j!}\,(t - x_0)^j$$

$$\varphi(t) = \sum_{j<n} \frac{y_{0,j}}{j!}\,(t - x_0)^j$$

If $y_{0,0}, y_{0,1}, \dots, y_{0,n-1}$ are the derivatives of some function f at x_0, i.e., $y_{0,j} = f^{(j)}(x_0)$, then φ is a finite Taylor series with center x_0:

$$\varphi(t) = \sum_{j<n} \frac{f^{(j)}(x_0)}{j!}\,(t - x_0)^j\,.$$

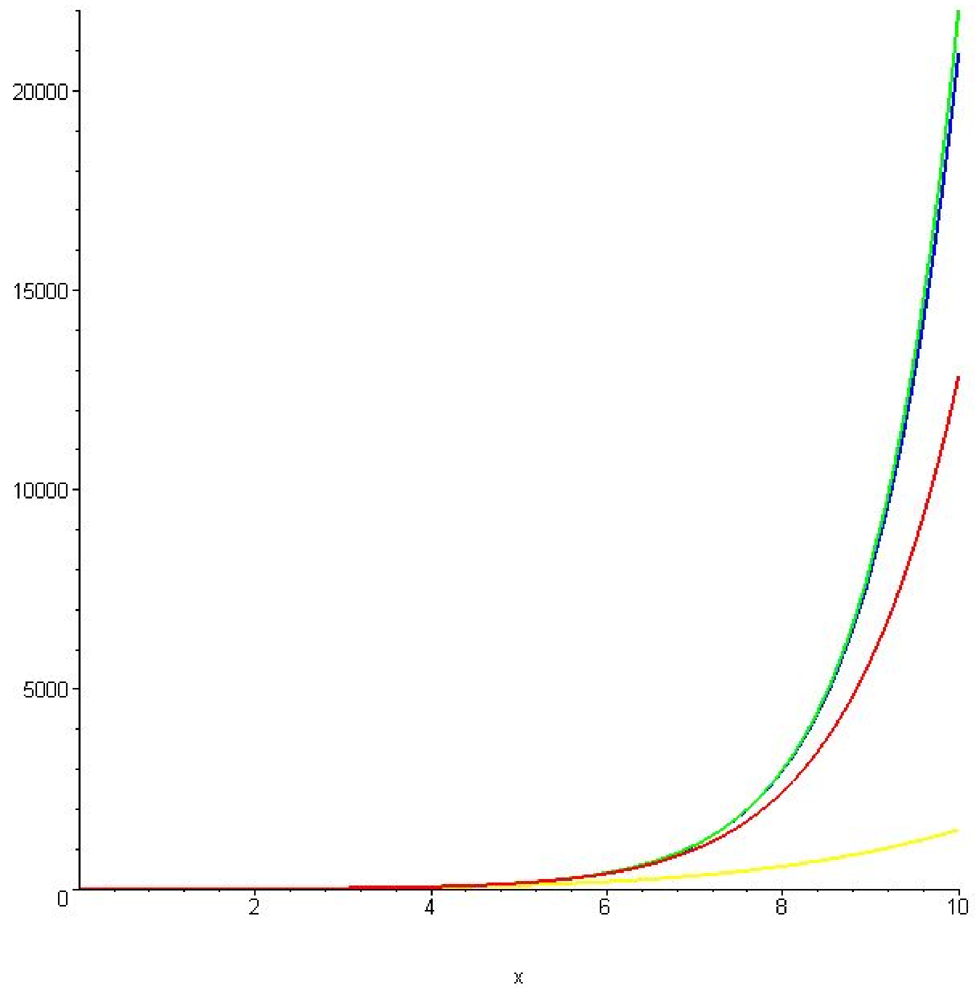
If f has a continuous nth derivative at interval $I(u, x_0)$, then

$$f(u) - \varphi(u) = \frac{f^{(n)}(\xi)}{n!}\,(u - x_0)^n\,,$$

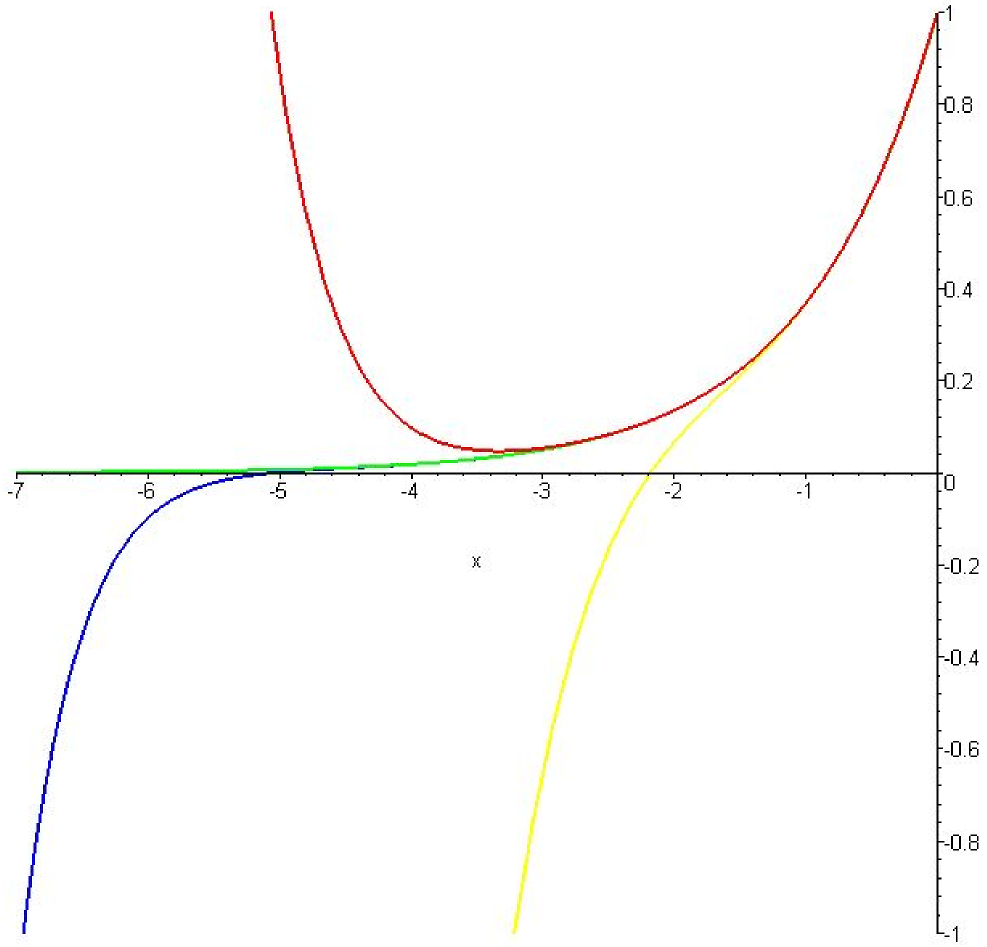
where $\xi \in I(u, x_0)$

$$|f(u) - \varphi(u)| \le \frac{M_n}{n!}\,|(u - x_0)^n|$$

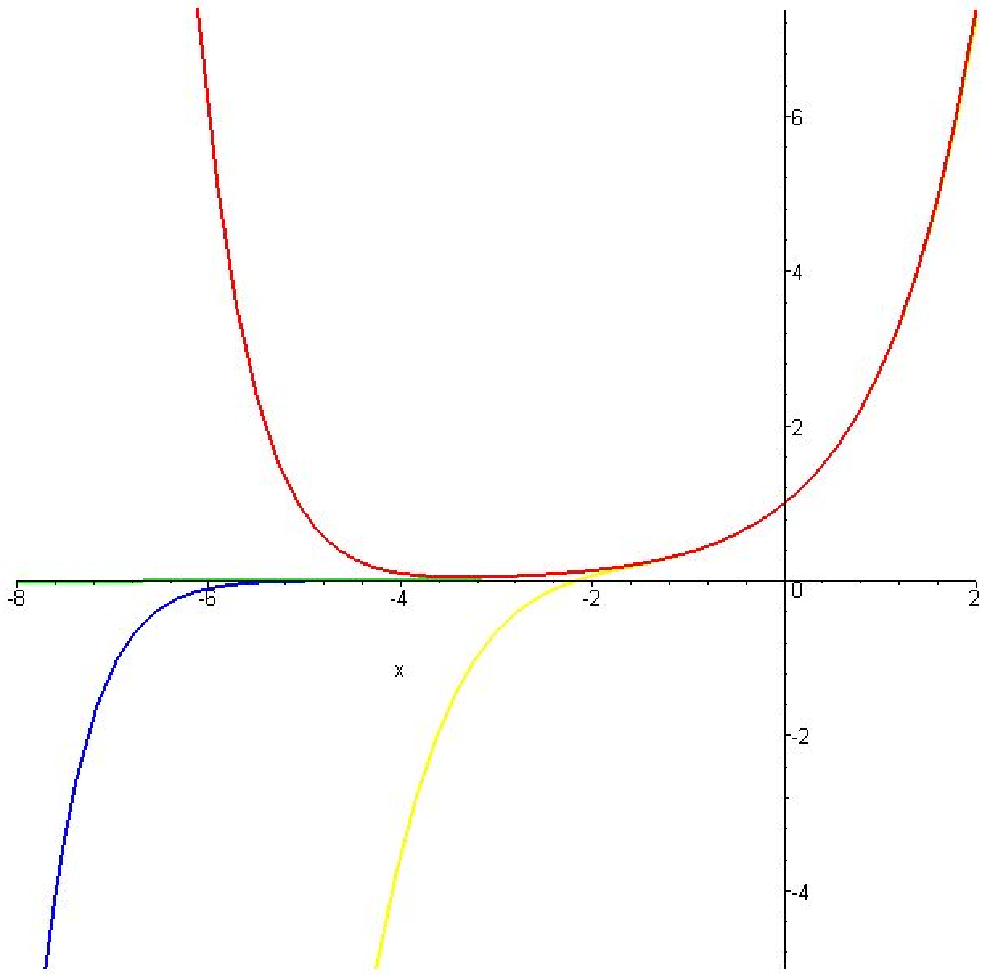
The only difference from the error of the interpolating polynomial is that polynomial $W(u)$ with roots $x_0, \dots, x_{n-1}$ is replaced by polynomial $(u - x_0)^n$ (of the same degree) with a single root x_0 of multiplicity n.
Approximation by Taylor polynomial is very precise in the neighborhood of x_0, but less precise in more distant points.



Approximations by Taylor polynomials of degrees 5, 10, and 15 for positive arguments.
Approximation by Taylor polynomial



Approximations by Taylor polynomials of degrees 5, 10, and 15 for negative arguments.
Approximation by Taylor polynomial



Approximations by Taylor polynomials of degrees 5, 10, and 15 for positive and negative arguments.

Spline interpolation

Disadvantage of polynomial interpolation: small change of input can cause big change of output at distant points (the influence is not localized).

Spline is a piecewise polynomial functionÃŋ. (It coincides with polynomials of small degree at particular intervals.)

The simplest case is **linear spline**, which is the approximation by a piecewise linear function.

Cubic spline

Given:
n nodes $x_0, \ldots, x_{n-1}$ **ordered increasingly**,
n values $y_0, \ldots, y_{n-1}$.
Task:
Find function φ, defined on interval $[x_0, x_{n-1}]$, such that:

- $\varphi(x_i) = y_i, \qquad i = 0, \ldots, n-1,$
- at each interval $[x_{i-1}, x_i], \qquad i = 1, \ldots, n-1$, function φ coincides with a polynomial φ_i of degree at most 3,
- the first two derivatives of φ are continuous at interval $[x_0, x_{n-1}]$.

(We shall find it necessary to make this task more precise.)

It suffices to ensure continuity of derivatives at $n-1$ nodes $x_1, \ldots, x_{n-2}$:

$$\begin{aligned} \varphi_i'(x_i) &= \varphi_{i+1}'(x_i), \qquad i = 1, \ldots, n-2\,, \\ \varphi_i''(x_i) &= \varphi_{i+1}''(x_i), \qquad i = 1, \ldots, n-2\,. \end{aligned}$$

Suppose that values $\varphi'(x_i) = \varphi_i'(x_i) = \varphi_{i+1}'(x_i)$, $i = 1, \ldots, n-2$ are known. Then φ' is continuous.

Polynomials φ_i, $i = 1, \ldots, n-1$ can be found just as the Hermite interpolating polynomial in the previous example; the only difference is that the nodes $0, 1$ are replaced by x_{i-1}, x_i.
The general case is obtained by an easy linear transformation of coordinates:

$$t = x_{i-1} + (x_i - x_{i-1})\, u\,, \qquad u = \frac{t - x_{i-1}}{x_i - x_{i-1}}\,.$$

On interval $[x_{i-1}, x_i]$, $i = 1, \ldots, n-1$, we get

$$\varphi_i(t) = y_{i-1}\eta_i(t) + y_i\varrho_i(t) + \varphi'(x_{i-1})\sigma_i(t) + \varphi'(x_i)\tau_i(t)\,,$$

where $\eta_i, \varrho_i, \sigma_i, \tau_i$ are polynomials of degree at most 3 (computed as the polynomials $\eta, \varrho, \sigma, \tau$ in the example of Hermite interpolating polynomial).
It remains to find $\varphi'(x_i)$, $i = 0, \ldots, n-1$:

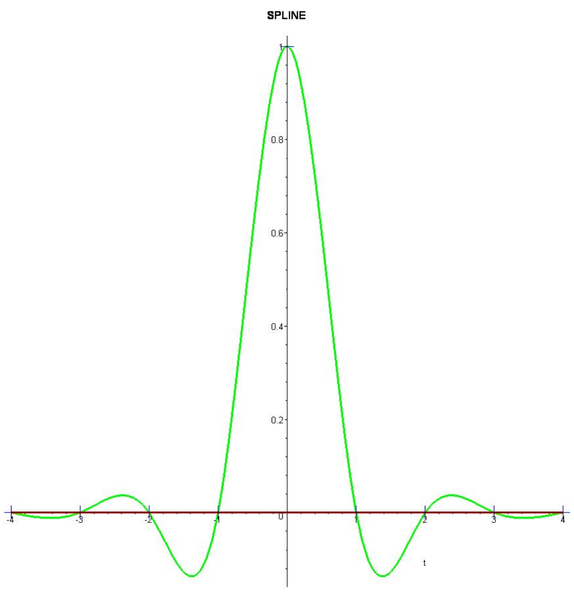
$$\begin{aligned} & y_{i-1}\eta_i''(x_i) + y_i\varrho_i''(x_i) + \varphi'(x_{i-1})\sigma_i''(x_i) + \varphi'(x_i)\tau_i''(x_i) = \\ = \; & y_i\eta_{i+1}''(x_i) + y_{i+1}\varrho_{i+1}''(x_i) + \varphi'(x_i)\sigma_{i+1}''(x_i) + \varphi'(x_{i+1})\tau_{i+1}''(x_i)\,, \end{aligned}$$

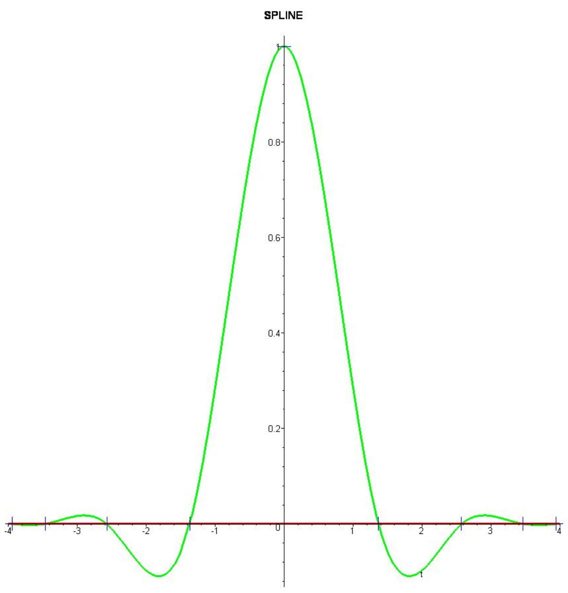
where $i = 1, \ldots, n-2$
(system of $n-2$ linear equations with n unknowns).
There remain 2 undefined parameters. The usual choice $\varphi''(x_0) = \varphi''(x_{n-1}) = 0$ leads to so-called **natural spline**. This choice influences the result only at the endpoints of the interval $[x_0, x_{n-1}]$.
The algorithm consists of 2 parts:
1. Computation of coefficients $\varphi'(x_i)$, $i = 0, \ldots, n-1$.
2. Evaluation of the spline.

The matrix of the system of linear equations is **three-diagonal** (its particular properties allow more effective solutions).
Splins cannot be used for extrapolation!

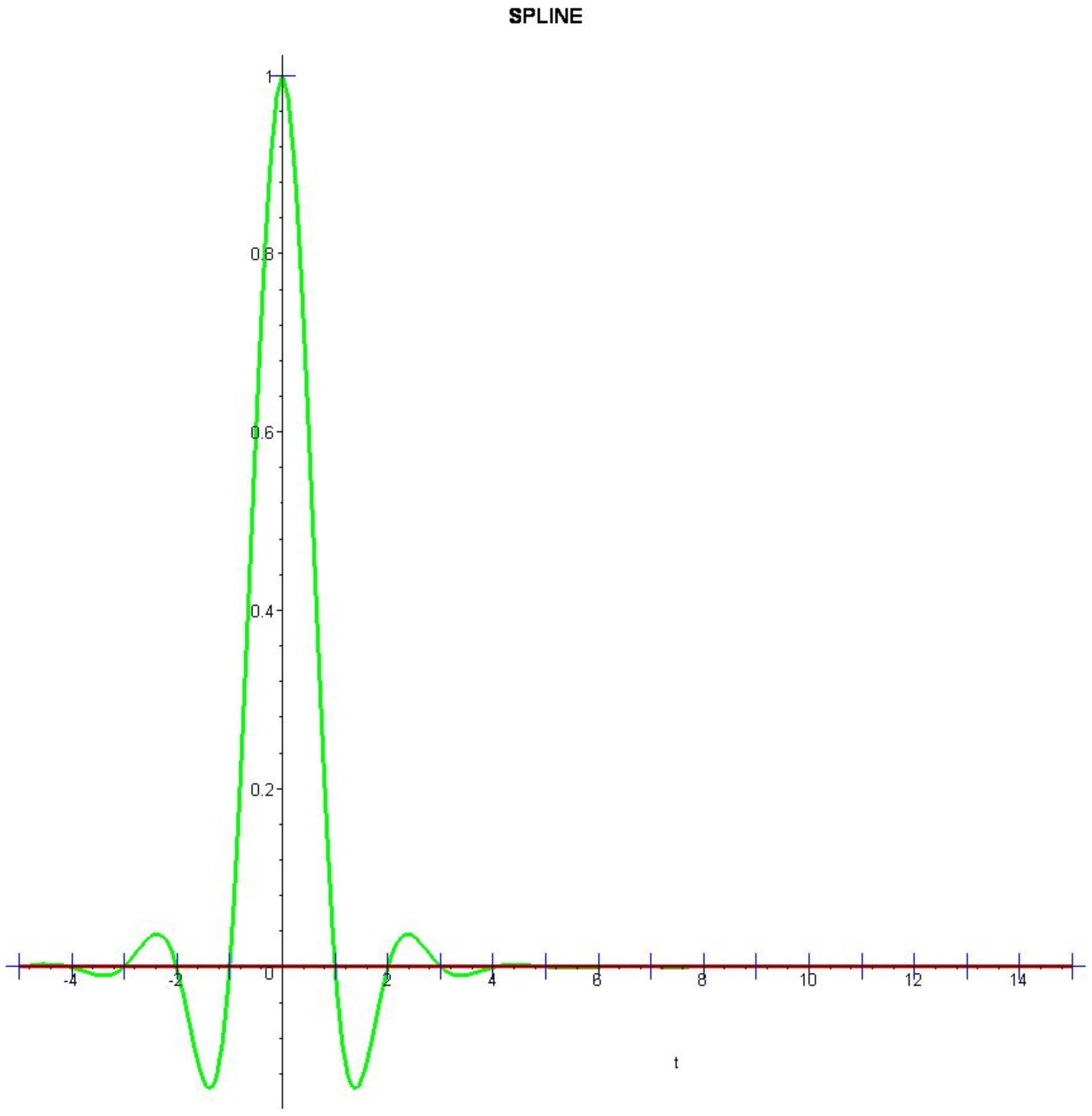
Still many implementations allow it.
Comment: The choice of nodes has smaller influence than in polynomial interpolation.

Influence of local changes on a spline

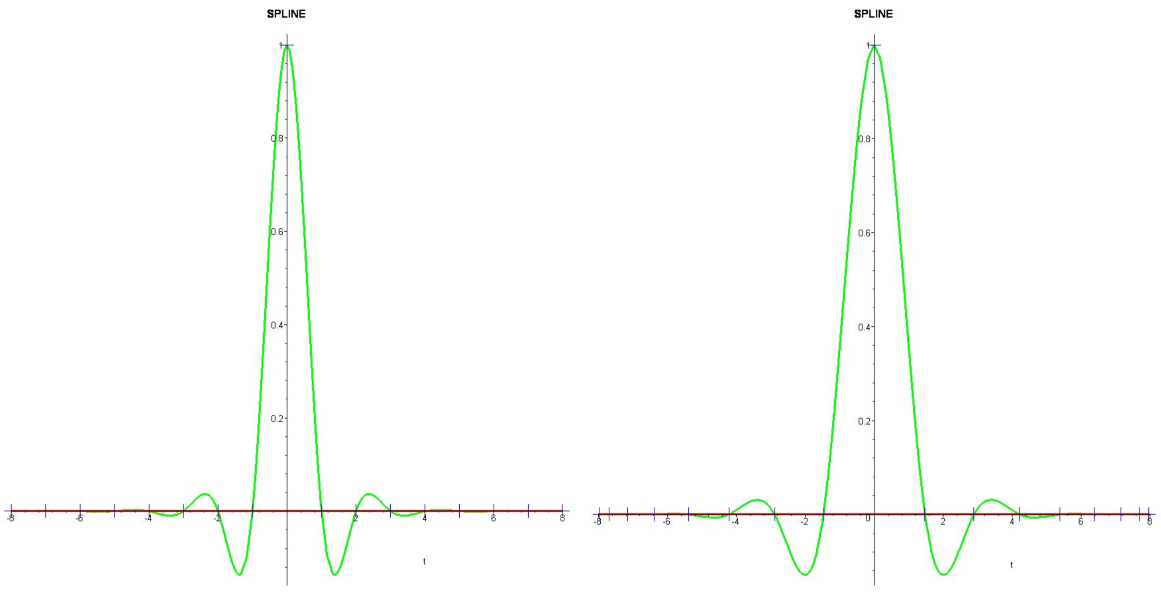




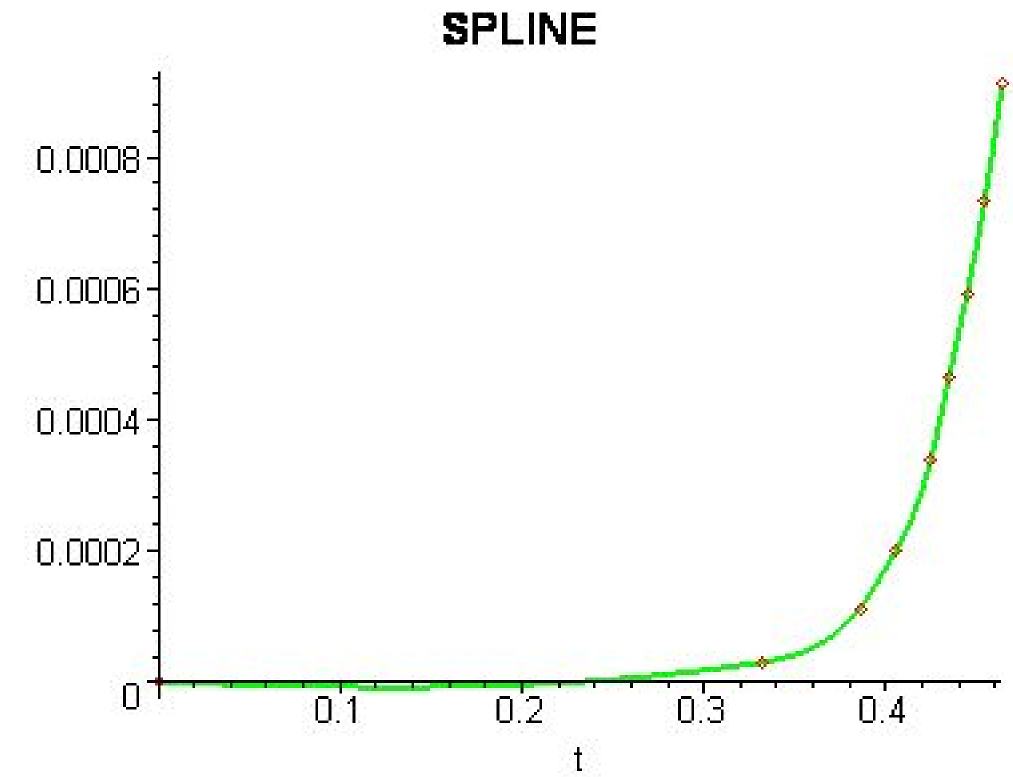
Influence of local changes on a spline

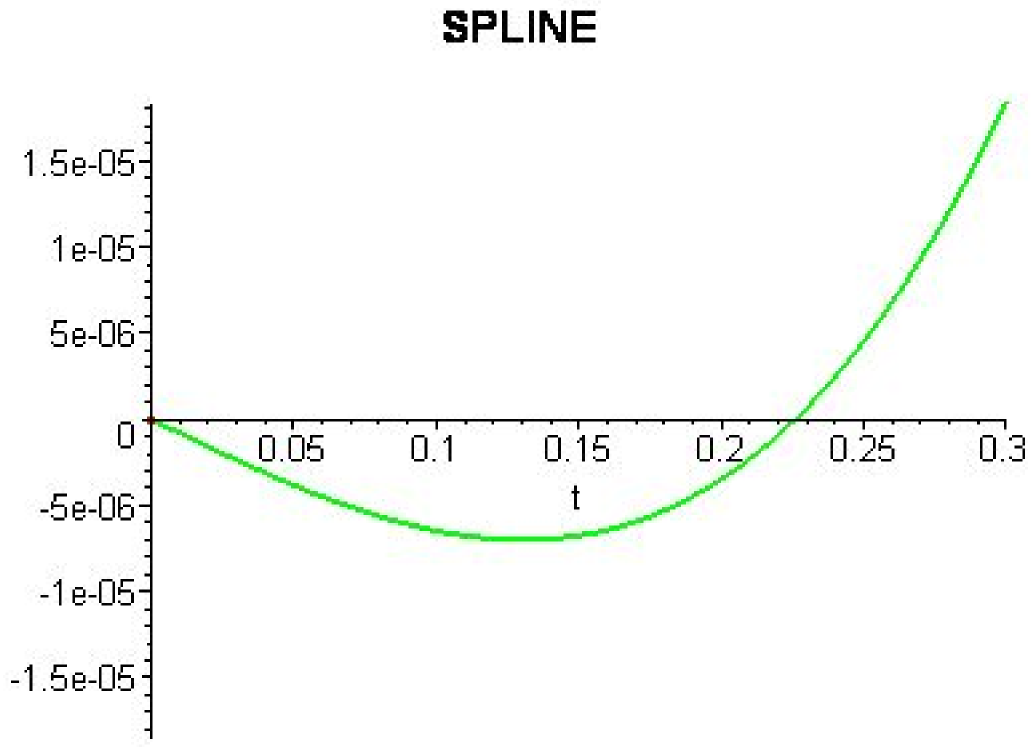


Influence of local changes on a spline



Motivating problem - spline





Given:
n nodes $x_0, \ldots, x_{n-1}$
n values $y_0, \ldots, y_{n-1}$
k functions $\varphi_0, \ldots, \varphi_{k-1}$, $k \leq n$, defined at least at the nodes.
Task:
Find coefficients $c_0, \ldots, c_{k-1} \in \mathbb{R}$ of a linear combination of functions φ_j

$$\varphi = \sum_{j<k} c_j \, \varphi_j$$

which minimizes

$$H_2 = \sum_{i<n} (\varphi(x_i) - y_i)^2 = \sum_{i<n} \left(\sum_{j<k} c_j \, \varphi_j(x_i) - y_i \right)^2$$

Modified criteria

$$H_{2w} = \sum_{i<n} w_i \, (\varphi(x_i) - y_i)^2$$

can be solved analogously,

$$H_1 = \sum_{i<n} |\varphi(x_i) - y_i|$$

not used, the solution is not unique,

$$H_0 = \max_{i<n} |\varphi(x_i) - y_i|$$

is more difficult to solve (Chebyshev approximation).

Solution of approximation by LSM

We introduce an inner (=scalar) product on $\mathbb{R}^n$, defined for $\vec{u} = (u_0, \ldots, u_{n-1})$, $\vec{v} = (v_0, \ldots, v_{n-1})$ by

$$\vec{u} \cdot \vec{v} = \sum_{i<n} u_i \cdot v_i \, .$$

We have to approximate vector $\vec{y}$ by a linear combination $\vec{\varphi} = \sum_{j<k} c_j \, \vec{\varphi}_j$,
the criterion is $H_2 = (\vec{\varphi} - \vec{y}) \cdot (\vec{\varphi} - \vec{y}) = \|\vec{\varphi} - \vec{y}\|^2$.
Solution: The orthogonal projection; it satisfies the system of equations (for $m = 0, \ldots, k-1$)

$$\begin{aligned} (\vec{\varphi} - \vec{y}) &\perp \vec{\varphi}_m, \\ (\vec{\varphi} - \vec{y}) \cdot \vec{\varphi}_m &= 0, \\ \vec{\varphi} \cdot \vec{\varphi}_m &= \vec{y} \cdot \vec{\varphi}_m. \end{aligned}$$

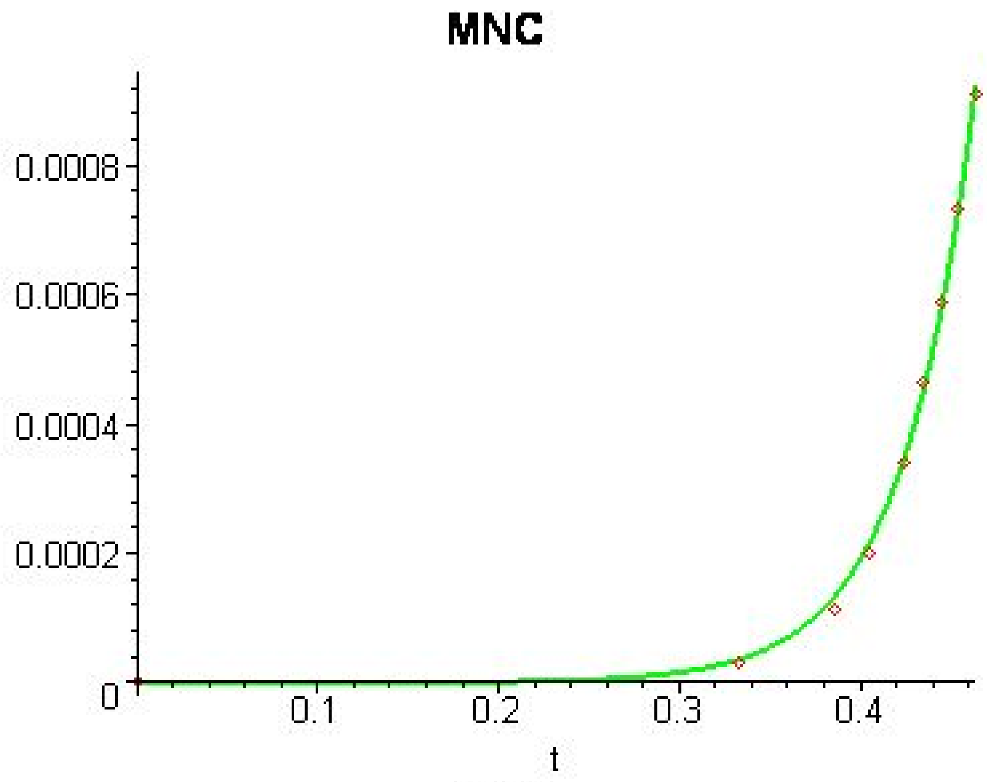
Vectors $\vec{\varphi}$ and $\vec{y}$ have the same inner products with vectors $\vec{\varphi}_m$.

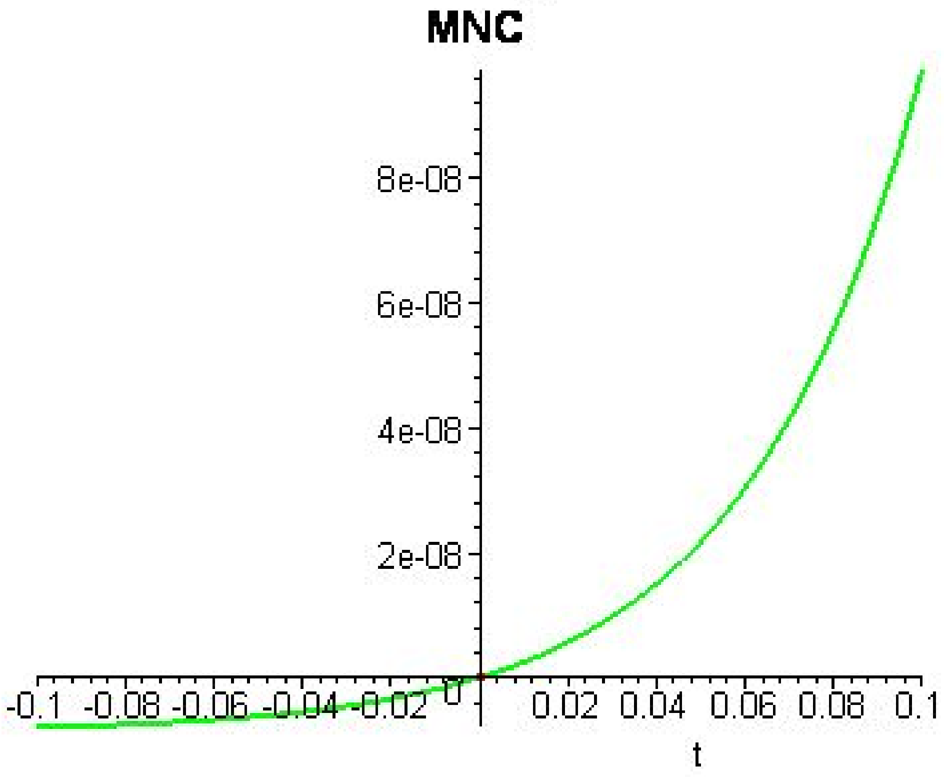
$$\begin{aligned} \left(\sum_{j<k} c_j \, \vec{\varphi}_j \right) \cdot \vec{\varphi}_m &= \vec{y} \cdot \vec{\varphi}_m, \\ \sum_{j<k} c_j (\vec{\varphi}_j \cdot \vec{\varphi}_m) &= \vec{y} \cdot \vec{\varphi}_m, \qquad m = 0, \ldots, k-1; \end{aligned}$$

system of linear equations for unknowns $c_0, \ldots, c_{k-1}$ - **system of normal equations**.
Special case: Approximation by a polynomial of degree less than k; we may choose $\varphi_j(t) = t^j$,

$$\vec{\varphi}_j \cdot \vec{\varphi}_m = \sum_{i<n} x_i^j \cdot x_i^m = \sum_{i<n} x_i^{j+m} \, .$$

Motivating problem - LSM





Approximation of V-A characteristic of a diode by a linear combination of a constant and two exponencials with appropriate bases.
In linear subspace $P = \langle \vec{\varphi}_0, \ldots, \vec{\varphi}_{k-1} \rangle$, we find an orthogonal basis $(\vec{\psi}_0, \ldots, \vec{\psi}_{k-1})$,

$$\vec{\psi}_j \cdot \vec{\psi}_m = 0 \text{ for } j \neq m\,.$$

Orthogonality depends not only on functions $\varphi_0, \ldots, \varphi_{k-1}$, but also on the choice of nodes!
We look for a solution in the form $\varphi = \sum\limits_{j<k} d_j\, \psi_j$, where d_j, $j = 0, \ldots, k-1$ are coordinates w.r.t. a new basis.

The matrix of the system of normal equations is diagonal:

$$\begin{aligned} d_j \left(\vec{\psi}_j \cdot \vec{\psi}_j\right) &= \vec{y} \cdot \vec{\psi}_j\,, \qquad j = 0, \dots, k-1, \\ d_j = \frac{\vec{y} \cdot \vec{\psi}_j}{\vec{\psi}_j \cdot \vec{\psi}_j} &= \frac{\vec{y} \cdot \vec{\psi}_j}{\|\vec{\psi}_j\|^2}\,, \qquad j = 0, \dots, k-1. \end{aligned}$$

Moreover, vectors $\vec{\psi}_j$ can be chosen unit, then also the denominator is 1.
Constructs an orthogonal basis

$$\vec{\psi}_0 = \vec{\varphi}_0$$

$$\vec{\psi}_1 = \vec{\varphi}_1 + \alpha_{10}\vec{\psi}_0$$

$$\begin{aligned} \vec{\psi}_1 \cdot \vec{\psi}_0 = 0 \quad &\Rightarrow \quad \vec{\varphi}_1 \cdot \vec{\psi}_0 + \alpha_{10}\vec{\psi}_0 \cdot \vec{\psi}_0 = 0 \\ &\Rightarrow \quad \alpha_{10} = \frac{-\vec{\varphi}_1 \cdot \vec{\psi}_0}{\vec{\psi}_0 \cdot \vec{\psi}_0} \end{aligned}$$

$$\vec{\psi}_2 = \vec{\varphi}_2 + \alpha_{20}\vec{\psi}_0 + \alpha_{21}\vec{\psi}_1$$

$$\begin{aligned} \vec{\psi}_2 \cdot \vec{\psi}_0 = 0 \quad &\Rightarrow \quad \vec{\varphi}_2 \cdot \vec{\psi}_0 + \alpha_{20}\vec{\psi}_0 \cdot \vec{\psi}_0 + \alpha_{21}\underbrace{\vec{\psi}_1 \cdot \vec{\psi}_0}_{0} = 0 \\ &\Rightarrow \quad \alpha_{20} = \frac{-\vec{\varphi}_2 \cdot \vec{\psi}_0}{\vec{\psi}_0 \cdot \vec{\psi}_0} \end{aligned}$$

$$\begin{aligned} \vec{\psi}_2 \cdot \vec{\psi}_1 = 0 \quad &\Rightarrow \quad \vec{\varphi}_2 \cdot \vec{\psi}_1 + \alpha_{20}\underbrace{\vec{\psi}_0 \cdot \vec{\psi}_1}_{0} + \alpha_{21}\vec{\psi}_1 \cdot \vec{\psi}_1 = 0 \\ &\Rightarrow \quad \alpha_{21} = \frac{-\vec{\varphi}_2 \cdot \vec{\psi}_1}{\vec{\psi}_1 \cdot \vec{\psi}_1} \end{aligned}$$

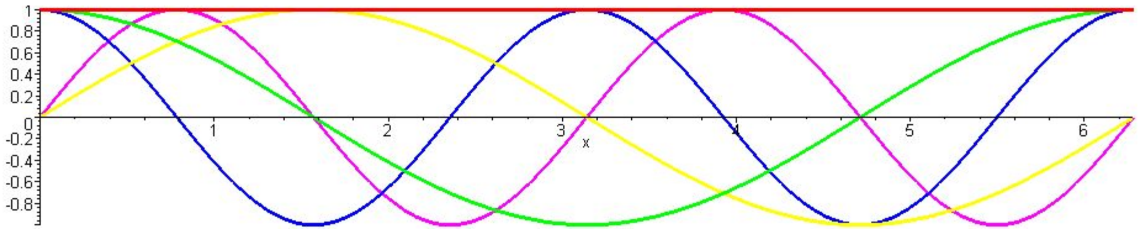
$$\dots$$

$$\vec{\psi}_j = \vec{\varphi}_j + \sum_{m<j} \alpha_{jm}\vec{\psi}_m$$

$$\begin{aligned} \forall p, p < j : \vec{\psi}_j \cdot \vec{\psi}_p &= 0 = \vec{\varphi}_j \cdot \vec{\psi}_p + \sum_{m<j} \alpha_{jm}\vec{\psi}_m \cdot \vec{\psi}_p = \vec{\varphi}_j \cdot \vec{\psi}_p + \alpha_{jp}\vec{\psi}_p \cdot \vec{\psi}_p \\ &\Rightarrow \quad \alpha_{jp} = \frac{-\vec{\varphi}_j \cdot \vec{\psi}_p}{\vec{\psi}_p \cdot \vec{\psi}_p} \end{aligned}$$

Approximation by LSM with the choice of approximating functions

$$1, \quad \cos 2\pi\frac{t}{T}, \quad \sin 2\pi\frac{t}{T}, \quad \cos 4\pi\frac{t}{T}, \quad \sin 4\pi\frac{t}{T}, \quad \dots$$



For equidistant nodes at an interval of length T,

$$x_i = a + i\frac{T}{n}, \qquad i = 0, \dots, n-1,$$

(at most n) vectors $\vec{\varphi}_j$ are orthogonal.
Given:
Bounded interval I,
continuous function f na I,
$k \in \mathbb{N}$
Task:
Find a polynomial φ of degree less than k minimizing

$$H_0 = \max_{t \in I} |\varphi(t) - f(t)|$$

There are methods solving this task, but their complexity is much higher;
instead of that, we usually apply modified LSM:
For simplicity, we take interval $I = \langle -1, 1 \rangle$; generalization to any interval $\langle a, b \rangle$ is done by a linear transformation

$$x = \frac{b+a}{2} + \frac{b-a}{2} z\,,$$

the inverse transformation is

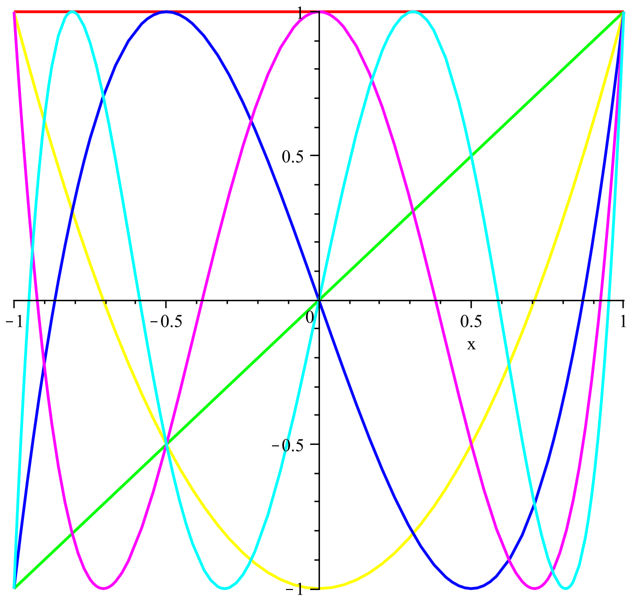
$$z = \frac{x - \frac{b+a}{2}}{\frac{b-a}{2}}\,.$$

We choose $n \geq k$ nodes $x_0, \ldots, x_{n-1} \in \langle -1, 1 \rangle$ with cosine distribution:

$$z_i = \cos \frac{\pi(i + \frac{1}{2})}{n}\,, \qquad i = 0, \ldots, n-1.$$

For a basis of the linear space of all polynomials of degree less than k, we take **Chebyshev polynomials**:

Chebyshev polynomials



$$\varphi_j(t) = \cos(j \arccos t), \qquad j = 0, \ldots, k-1, \quad k < n.$$

They can be computed by a recurrent formula

$$\begin{aligned} \varphi_0(t) &= 1, \\ \varphi_1(t) &= t, \\ \varphi_j(t) &= 2t\varphi_{j-1}(t) - \varphi_{j-2}(t), \qquad j \geq 2 \end{aligned}$$

The range of functions φ_j, $j \geq 1$, at interval I is $\langle -1, 1 \rangle$.

The vectors $\vec{\varphi}_0, \dots, \vec{\varphi}_{k-1}$ are orthogonal,

$$\vec{\varphi}_j \cdot \vec{\varphi}_m = \begin{cases} 0 & \text{if } j \neq m, \\ \frac{n}{2} & \text{if } j = m > 0, \\ n & \text{if } j = m = 0. \end{cases}$$

For $k = n$, we obtain the interpolating polynomial (with the recommended distribution of nodes). The solution for $k < n$ differs by ignoring the summands of higher order. Coefficients $c_k, \dots, c_{n-1}$ are *usually* small (they depend on higher derivatives of the approximated function!). An upper estimate of the error at nodes is

$$|\varphi(x_i) - f(x_i)| \leq \sum_{j=k}^{n-1} |c_j|.$$

Remarks on Chebyshev approximation

- We do not optimize the criterion H_0, but the result is not much different from the optimal solution.
- We cannot say much about the error at other points than the nodes; nevertheless, the method can be recommended.
- The recurrent formula is used not only for computation of Chebyshev polynomials, but also to their direct evaluation at given points.
- It is not recommended to expand the result to the standard form $\varphi(t) = \sum_{j<k} b_j \, t^j$.
- Chebyshev approximation can be generalized to an approximation by a product of a **known** function and an **unknown** polynomial.