# Solution of nonlinear equations

# Nonlinear equations

# Nonlinear equations

Example. $x^2 = 4 \sin x$. Find the real solutions.

Example. $x = \cos x$. Find the real solutions.

Example. $x^5 - 4x^4 + x^3 - x^2 + 4x - 4 = 0$. Find the real solutions. There is no solution formula that computes the roots from the coefficients.

**Problem:** We do not know whether the equation is solvable and how many solution does the equation have.

# Separation of the roots

**Thm. 94.** (Bolzano) If a continuous function satisfies the conditions $f(a) \cdot f(b) < 0$ $(a < b)$, then there exists a constant $c \in (a, b)$ such that $f(c) = 0$.

Rmk. We calculate the function values at certain points, and if the values have different sign at neighbouring points then there is root between these points.

Rmk. If the function is strictly monotone on a certain interval and there is a root in the interval, then the root is unique.

Rmk. It can be helpful if we draw the graphs of the functions. E.g. we draw the graphs of the left and the right hand side functions, and fix the interval which the intersection located in.

# Polynomials

# Evaluating polynomials

Horner's scheme (William George Horner (1786-1837, British))

$$a_n x^n + \ldots + a_1 x + a_0 = (\ldots((a_n x + a_{n-1})x + a_{n-2})\ldots)x + a_0$$

Rmk. There are altogether $n$ additions in the formula. In 1954, Ostrowski proved that we need at least $n$ additions to evaluate a polynomial.

Rmk. Victor Pan proved a similar theorem for the number of the multiplications in 1966.

**Thm. 95.** The roots of the polynomial $p(x) = a_n x^n + \ldots + a_1 x + a_0$ $(a_n, a_0 \neq 0)$ are located in the two circle rings centred in the origin with radius $R = 1 + A/|a_n|$ and $r = 1/(1 + B/|a_0|)$, where

$$A = \max\{|a_{n-1}|, \ldots, |a_0|\},\ B = \max\{|a_n|, \ldots, |a_1|\}.$$

Rmk. In case of $p(x) = x^5 - 4x^4 + x^3 - x^2 + 4x - 4$ we have $1/2 \leq |x_k| \leq 5$.

# Conditioning of root finding

# Conditioning

Let us consider the properly posed problem

$$f(x) = d.$$

Let us suppose that the solution function $G(d)$ is differentiable. Then

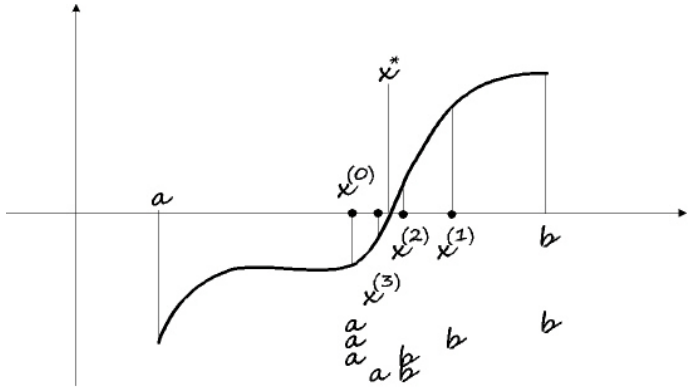
$$\kappa(d) = \left| \frac{G'(d)d}{G(d)} \right| = \left| \frac{d}{f'(G(d))G(d)} \right|, \ \kappa_{abs}(d) = \frac{1}{|f'(G(d))|}.$$

If $|f'(x)|$ is small then the problem is ill-conditioned, and if it is large then the problem is well-conditioned.

# Bisection method

# Bisection method

$f(x) = 0 \longrightarrow$ Find the root $x^\star$.




$x_k \to x^\star$

# Bisection method

Bisection method, $a < b$ and $f$ are given, $f(a) < 0 < f(b)$.

**for** $k := 1 : k_{\max}$ **do**
  $x := a + (b - a)/2$
  $f := f(x)$
  **if** $f = 0$ **then**
    end
  **else**
    **if** $f > 0$ **then**
      $b = x$
    **else**
      $a = x$
    **end if**
  **end if**
**end for**

# Bisection method

Rmk. Convergence order cannot be defined. But it is true the estimation

$$|e_k| \leq \frac{b-a}{2^{k+1}}.$$

This shows that we can expect one digit improvement after 3 steps.

Rmk. When we use only mantissas with two digits then we compute $(0.67 + 0.69)/2 = 1.36/2 \approx 1.4/2 = 0.7$, which is not between the two numbers. But $0.67 + (0.69 - 0.67)/2 = 0.67 + 0.02/2 = 0.67 + 0.01 = 0.68$.

Rmk. If the function has more than one roots then the method will surely find one of them.

Rmk. Other stopping conditions:

$$\frac{|x_k - x_{k-1}|}{|x_{k-1}|} \leq tol., \ |f(x_k)| \leq tol.$$

# Newton's method

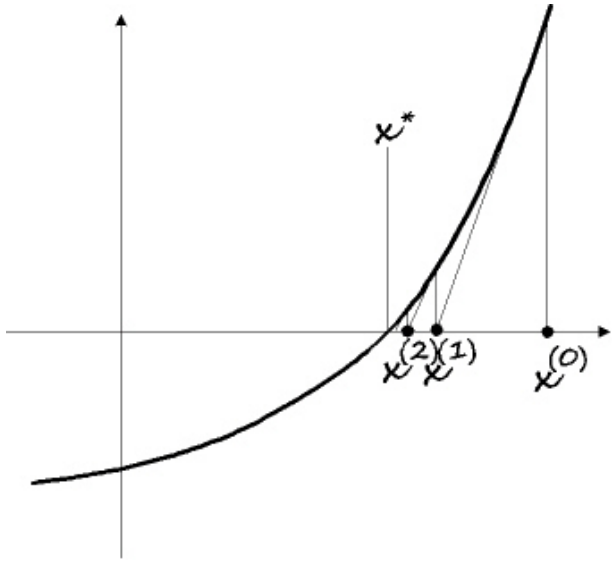
# Newton's method

Newton (1669), Raphson (1690)

Newton's method, $x_0$ and $f$ are given.

$x := x_0$
**for** $k := 1 : k_{\max}$ **do**
  $x := x - \frac{1}{f'(x)} f(x)$
  **if** $f(x) = 0$ **then**
    end
  **end if**
**end for**

# Newton's method



**Thm. 96.** (Monotone convergence theorem) Let us suppose that $f \in C^2$ and that the first and the second derivatives of the function do not have roots in the closed interval determined by the points $x^\star$ and $x_0$, moreover $f(x_0) \cdot f''(x_0) > 0$. Then the sequence $\{x_k\}$ generated by the Newton's method tends to $x^\star$ monotonically.

Proof: Let $x_0 > x^\star$ és $f(x_0) > 0, f''(x_0) > 0$ $(f'(x) > 0)$. We can see from the iteration

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

that $x_{k+1} \leq x_k$, that is the sequence is monotonically decreasing. It follows from the strict convexity that $x_k \geq x^\star$. Thus the sequence is convergent. Let us denote the limit with $\bar{x}^\star$.

Then

$$\underbrace{x_{k+1}}_{\to \bar{x}^\star} = \underbrace{x_k}_{\to \bar{x}^\star} - \frac{\overbrace{f(x_k)}^{\to f(\bar{x}^\star)}}{\underbrace{f'(x_k)}_{\to f'(\bar{x}^\star)}},$$

which implies that $\bar{x}^\star = x^\star$. ■

**Thm. 97.** Under the conditions of the previous theorem, the convergence of $\{x_k\}$ is of second order, moreover if $|f'(x)| \geq m_1 > 0$ and $|f''(x)| \leq M_2 < \infty$ in the interval determined by the points $x_0$ and $x^\star$ with appropriately chosen constants $m_1$ and $M_2$, then it is valid the estimation

$$|e_{k+1}| \leq \frac{M_2}{2m_1}|e_k|^2.$$

# Newton's method

Proof: Let us use Taylor's expansion around the point $x_k$:

$$0 = f(x^\star) = f(x_k) + f'(x_k)(x^\star - x_k) + \frac{1}{2} f''(\xi)(x^\star - x_k)^2,$$

where $\xi$ falls between $x^\star$ and $x_k$. From the reordering of the Newton's iteration:

$$0 = f(x_k) + f'(x_k)(x_{k+1} - x_k).$$

After subtraction:

$$0 = f'(x_k)(x_{k+1)} - x^\star) - \frac{1}{2} f''(\xi)(x^\star - x_k)^2.$$

Finally

$$|f'(x_k)| \cdot |e_{k+1}| = \frac{1}{2} |f''(\xi)| \cdot |e_k|^2.$$

# Newton's method

Thus

$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k|^2} = \frac{f''(x^\star)}{2f'(x^\star)},$$

which shows that the order of the convergence is second order, indeed. Moreover

$$|e_{k+1}| = \frac{|f''(\xi)|}{2|f'(x_k)|} \cdot |e_k|^2 \leq \frac{M_2}{2m_1}|e_k|^2. \blacksquare$$

Rmk. Newton's method can be applied combined with the bisection method. First we approaches the root with the bisection method in order to fulfil the conditions of the above theorems, then we switch to Newton's method in order to accelerate the convergence.

# A simple error estimation

Let us use Taylor's expansion around the point $x_k$:

$$0 = f(x^\star) = f(x_k) + f'(\xi)(x^\star - x_k),$$

where $\xi$ is between the points $x_k$ and $x^\star$.

Thus

$$|x^\star - x_k| = \frac{|f(x_k)|}{|f'(\xi)|} \leq \frac{|f(x_k)|}{m_1}.$$

# Fixed point iterations

# Fixed point iterations

**Thm. 98.** Let us suppose that the root $x^\star \in [a,b]$ of the function $f$ is a fixed point of the function $F : [a,b] \to \mathbb{R}$. Let us suppose that $F$ is a contraction with contraction coefficient $q$. Then the iteration $x_{k+1} = F(x_k)$ converges from arbitrary initial point $x_0 \in [a,b]$ to the unique solution of the equation $f(x) = 0$. Moreover

$$|x_k - x^\star| \leq \frac{q^k}{1-q}|x_1 - x_0|$$

Proof: The corollary of Banach's fixed point theorem. ■

Rmk. In certain cases $F$ can be given as $F(x) = x - g(x) \cdot f(x)$, where $g$ is a sufficiently chosen number that guarantees the contraction of $F$.

Rmk. The contraction property can be guaranteed supposing that $F$ is continuous on $[a, b]$ and differentiable in $(a, b)$, moreover there exists a number $0 \leq q < 1$, for which we have $|F'(x)| \leq q$, $\forall x \in (a, b)$ (Lagrange's mean value theorem).

**Thm. 99.** If, in the previous theorem, $F$ is continuously differentiable at least $r$ times and

$$F'(x^\star) = \ldots = F^{(r-1)}(x^\star) = 0$$

and $F^{(r)}(x^\star) \neq 0$, then the convergence order of the sequence $\{x_k\}$ is $r$ and it is valid the estimation

$$|e_{k+1}| \leq \frac{M_r}{r!}|e_k|^r,$$

where $M_r$ is an upper bound for the absolute value of the $r$th derivative of the function.

Proof: From the Taylor expansion around the point $x^\star$, we have

$$F(x_k) = F(x^\star) + \frac{F^{(r)}(\xi)}{r!}(x_k - x^\star)^r,$$

where $\xi$ is between the numbers $x_k$ and $x^\star$. That is

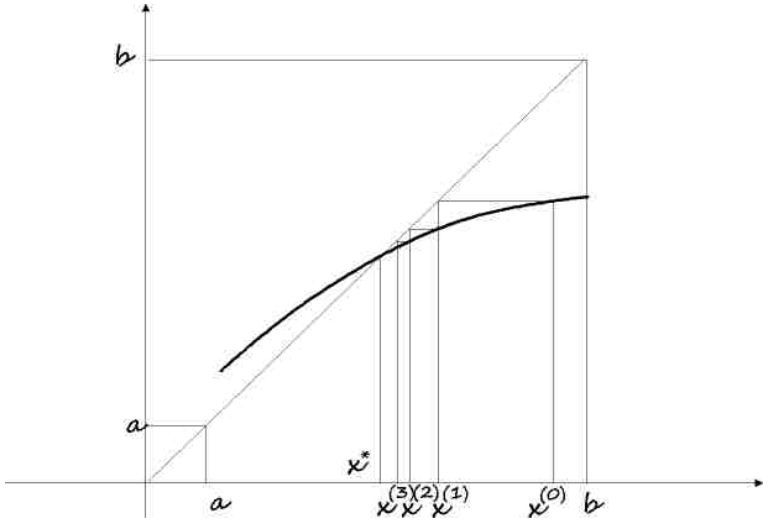
$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k|^r} = \frac{F^{(r)}(x^\star)}{r!}$$

that shows the $r$th order convergence of the method and the required estimation

$$|e_{k+1}| \leq K|e_k|^r. \blacksquare$$

Rmk. Newton's method can be written also in a fixed iteration form with the choice $g(x) = 1/f'(x)$. Its second order convergence could be proven also with the previous theorem.

# Fixed point iterations



# Systems of nonlinear equations

# Newton's method for systems of nonlinear equations

Solve the nonlinear system for the solution $\overline{\mathbf{x}}^\star \in \mathbb{R}^n$

$$\overline{\mathbf{f}}(\overline{\mathbf{x}}) = \mathbf{0},\ \overline{\mathbf{f}} : \mathbb{R}^n \to \mathbb{R}^n.$$

Example. Find the solution of the system

$$\begin{aligned} x^2 + y - 5 &= 0 \\ x + y^2 - 3 &= 0 \end{aligned}$$

Let us approximate $\overline{\mathbf{f}}$ around the point $\overline{\mathbf{x}}^\star$ with its second order Taylor expansion

$$\overline{\mathbf{f}}(\overline{\mathbf{x}}) \approx \underbrace{\overline{\mathbf{f}}(\overline{\mathbf{x}}^\star)}_{\mathbf{0}} + \overline{\mathbf{f}}'(\overline{\mathbf{x}}^\star)(\overline{\mathbf{x}} - \overline{\mathbf{x}}^\star),$$

where $\overline{\mathbf{f}}'(\overline{\mathbf{x}}^\star)$ is the Jacobian of the function $\overline{\mathbf{f}}$ at the solution $\overline{\mathbf{x}}^\star$. From this, we can approximate the solution as

$$\overline{\mathbf{x}}^\star \approx \overline{\mathbf{x}} - \left[\overline{\mathbf{f}}'(\overline{\mathbf{x}}^\star)\right]^{-1} \overline{\mathbf{f}}(\overline{\mathbf{x}}) \approx \overline{\mathbf{x}} - \left[\overline{\mathbf{f}}'(\overline{\mathbf{x}})\right]^{-1} \overline{\mathbf{f}}(\overline{\mathbf{x}}).$$

# Newton's method for systems of nonlinear equations

Using this approximation recursively, we arrive at an iterative method, the so-called Newton's method

$$\overline{\mathbf{x}}_{k+1} = \overline{\mathbf{x}}_k - \left[\overline{\mathbf{f}}'(\overline{\mathbf{x}}_k)\right]^{-1} \overline{\mathbf{f}}(\overline{\mathbf{x}}_k).$$

(We solve the system $\overline{\mathbf{f}}'(\overline{\mathbf{x}}_k)\overline{\mathbf{y}} = \overline{\mathbf{f}}(\overline{\mathbf{x}}_k)$ for $\overline{\mathbf{y}}$ then we update as $\overline{\mathbf{x}}_{k+1} = \overline{\mathbf{x}}_k - \overline{\mathbf{y}}$.)

Example.

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} - \begin{bmatrix} 2x_k & 1 \\ 1 & 2y_k \end{bmatrix}^{-1} \begin{bmatrix} x_k^2 + y_k - 5 \\ x_k + y_k^2 - 3 \end{bmatrix}$$

# Newton's method for systems of nonlinear equations

**Thm. 100.** Let us suppose that $\overline{\mathbf{f}}$ is continuously differentiable in a neighbourhood of $\overline{\mathbf{x}}^\star$, moreover let the Jacobians be bounded and Lipschitz continuous here. Then, when we start the Newton's iteration sufficiently close to $\overline{\mathbf{x}}^\star$, it will converge to $\overline{\mathbf{x}}^\star$ quadratically.

Rmk. The solution of a nonlinear system may be obtained also by fixed point iteration. If the equation $\overline{\mathbf{f}}(\overline{\mathbf{x}}) = \mathbf{0}$ is equivalent with the equation $\overline{\mathbf{x}} = \mathbf{F}(\overline{\mathbf{x}})$ with a suitably chosen function $\mathbf{F}$, and the iteration $\overline{\mathbf{x}}_{k+1} = \mathbf{F}(\overline{\mathbf{x}}_k)$ converges to $\overline{\mathbf{x}}^\star$, then $\overline{\mathbf{x}}^\star$ is the solution of $\overline{\mathbf{f}}(\overline{\mathbf{x}}) = \mathbf{0}$.

# Relations between root-finding and minimization

Solve $\overline{\mathbf{f}}(\overline{\mathbf{x}}) = \mathbf{0} \Longrightarrow$ find the minimum of the multivariable function $\|\overline{\mathbf{f}}(\overline{\mathbf{x}})\|$

Find the minimum of the multivariable function $f(\overline{\mathbf{x}}) \Longrightarrow$ solve $\nabla f(\overline{\mathbf{x}}) = \mathbf{0}$

**Thm. 101.** Let us suppose that in a neighbourhood of $\overline{\mathbf{x}}^\star$ the multivariable function $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable. If the conditions

$$\nabla f(\overline{\mathbf{x}}^\star) = \mathbf{0}, \ \nabla^2 f(\overline{\mathbf{x}}^\star) \text{ is s.p.d.}$$

are fulfilled, where $\nabla^2 f(\overline{\mathbf{x}}^\star)$ denotes the Hessian of the function $f$ at the point $\overline{\mathbf{x}}^\star$, then $\overline{\mathbf{x}}^\star$ is a local minimizer of the function $f$.

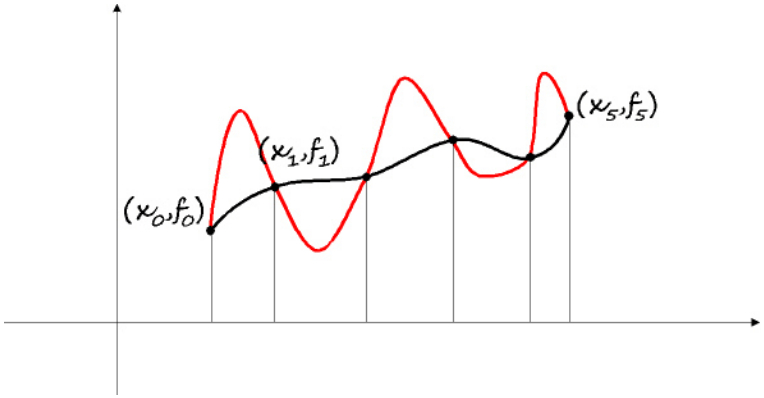
The possible local minimizer $\overline{\mathbf{x}}^\star$ may be found with the Newton's method applied to the equation $\nabla f(\overline{\mathbf{x}}) = \mathbf{0}$ as follows:

$$\overline{\mathbf{x}}_{k+1} = \overline{\mathbf{x}}_k - \left[\nabla^2 f(\overline{\mathbf{x}}_k)\right]^{-1} \nabla f(\overline{\mathbf{x}}_k).$$

# Interpolation with polynomials

# The interpolation problem

# The problem to solve



# The problem to solve

Let us suppose that we know the values of a function $f$ only at $n+1$ distinct points (the so-called nodes) ($(x_i, f_i)$ pairs ($i = 0, \ldots, n$), $x_i \neq x_j$, ha $i \neq j$).

**Problem:**

- Let us calculate the values of the function at other points;
- Let us calculate the derivative of the function;
- Let us calculate the extremizers of the function;
- Let us calculate its definite integral!

**Solution:** We give a functions $\phi$ with the properties $\phi(x_i) = f_i$ and we use this function in the calculation instead of the original (unknown) function. The functions $\phi$ are generally chosen to be polynomials, trigonometric polynomials (sin, cos) or piecewise polynomials.

# Lagrange interpolation

# Interpolation with polynomials

**Thm. 102.** For all fixed $n+1$ nodes, there exists a unique polynomial $L_n$ with degree at most $n$ such that $L_n(x_i) = f_i$.

Proof: Let us choose the required polynomial to be $L_n(x) = \sum_{k=0}^{n} a_k x^k$. In order to satisfy the interpolation property, the following equalities must be valid:

$$L_n(x_i) = \sum_{k=0}^{n} a_k x_i^k = f_i \ (i = 0, \ldots, n).$$

This is a SLAE. Its coefficient matrix is a so-called Vandermonde matrix. Because $x_i \neq x_j$, if $i \neq j$, its determinant is not zero. Thus, the SLAE can be solved uniquely for the coefficients. ■

# Interpolation with polynomials – Lagrangian form


Joseph-Louis Lagrange, 1736-1813, Italian (Giuseppe Lodovico Lagrangia)

**Def. 103.** For the fixed nodes $x_0, \ldots, x_n$, the polynomial

$$l_k(x) = \frac{(x - x_0) \ldots (x - x_{k-1})(x - x_{k+1}) \ldots (x - x_n)}{(x_k - x_0) \ldots (x_k - x_{k-1})(x_k - x_{k+1}) \ldots (x_k - x_n)}$$

$(k = 0, \ldots, n)$ is called the $k$th (it belongs to the point $x_k$) characteristic Lagrange polynomial.

# Interpolation with polynomials – Lagrangian form

Trivially we have

$$l_k(x_i) = \begin{cases} 1, & \text{if } i = k, \\ 0, & \text{if } i \neq k. \end{cases}$$

Rmk. With the notation $w_{n+1}(x) = (x - x_0) \dots (x - x_n)$ (so-called nodal polynomial) the $k$th characteristic Lagrange polynomial can be written in the form

$$l_k(x) = \frac{w_{n+1}(x)}{(x - x_k) \cdot w'_{n+1}(x_k)}.$$

**Lagrange form of the interpolation polynomial:**

$$L_n(x) = \sum_{k=0}^{n} f_k l_k(x).$$

This polynomial trivially has degree at most $n$ and its graph goes through the given points.

# Interpolation with polynomials – Lagrangian form

Example. Find the interpolation polynomial to the points $(0, 2)$, $(1, 1)$ and $(3, 5)$!

The characteristic Lagrange polynomials are:

$$l_0(x) = \frac{(x-1)(x-3)}{(0-1)(0-3)} = \frac{1}{3}(x-1)(x-3),$$

$$l_1(x) = \frac{(x-0)(x-3)}{(1-0)(1-3)} = \frac{-1}{2}x(x-3),$$

$$l_2(x) = \frac{(x-0)(x-1)}{(3-0)(3-1)} = \frac{1}{6}x(x-1),$$

thus the interpolation polynomial is

$$p_2(x) = 2l_0(x) + 1l_1(x) + 5l_2(x) = x^2 - 2x + 2.$$

# Conditioning

# Conditioning

Let

$$L_n(x) = \sum_{k=0}^{n} f_k l_k(x), \quad \tilde{L}_n(x) = \sum_{k=0}^{n} \tilde{f}_k l_k(x),$$

where we use the approximate values $\tilde{f}_k$ instead of the original values $f_k$.

$$\|L_n - \tilde{L}_n\|_{C[a,b]} \leq \max_{k=0,\ldots,n} \{|f_k - \tilde{f}_k|\} \underbrace{\left\| \sum_{k=0}^{n} |l_k| \right\|_{C[a,b]}}_{\Lambda_n}.$$

$\Lambda_n$ is the so-called Lebesgue constant.

**Thm. 104.** [Erdős, 1961] There is a constant $K > 0$ such that

$$\Lambda_n \geq \frac{2}{\pi} \ln(n+1) - K$$

for all arbitrary choice of nodal points.

Rmk. For small $n$ values the interpolation problem is well-conditioned and for large values it is ill-conditioned.

# Interpolation error

# Interpolation error

**Thm. 105.** [Cauchy, 1840] Let the function $f \in C^{n+1}$ and the nodal points $x_0, \ldots, x_n$ be given. Let us fix a point $x$ and denote the interval determined by the nodal points and the point $x$ by $I_x$. Let us denote the interpolation polynomial of $f$ determined by the nodal points by $L_n f$. Then

$$E_n(x) := f(x) - (L_n f)(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} w_{n+1}(x).$$

Proof: If $x$ is a nodal point, then the statement is trivial. Otherwise let

$$G(t) := E_n(t) - \frac{w_{n+1}(t)}{w_{n+1}(x)} E_n(x), \ t \in I_x,$$

which is a $C^{n+1}$ function on the interval $I_x$. This function has $n+2$ roots.

# Interpolation error

Then the function $G'(t)$ has at least $n+1$ roots, etc., and the function $G^{(n+1)}(t)$ has at least one root. Let us denote this root by $\xi_x$.

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

thus

$$G^{(n+1)}(\xi_x) = f^{(n+1)}(\xi_x) - \frac{(n+1)!}{w_{n+1}(x)} E_n(x) = 0,$$

hence

$$E_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} w_{n+1}(x). \blacksquare$$

# Interpolation error

**Thm. 106.** If $f \in C^{\infty}[a,b]$ and the nodal points $x_0^{(n)}, \ldots, x_n^{(n)}$ are chosen from the interval $[a,b]$ $(n = 1, 2, \ldots)$, moreover, if $\exists M > 0$ such that $\max_{x\in[a,b]}\{|f^{(n)}|\} \leq M^n$, then $\max_{x\in[a,b]}\{|f(x) - (L_n f)(x)|\} \to 0$ if $n \to \infty$.

Proof: We apply the previous theorem:

$$|E_n(x)| = \frac{|f^{(n+1)}(\xi_x)|}{(n+1)!}|w_{n+1}(x)| \leq \frac{M^{n+1}}{(n+1)!}(b-a)^{n+1} \to 0,$$

if $n \to \infty$, even independently of $x$. ■

Rmk. We will generally use the notation $M_n$ for an upper bound of $\max\{|f^{(n)}|\}$ on a predefined interval. Similarly, $m_n$ will denote a non-negative lower bound for $\min\{|f^{(n)}|\}$.

# Runge's example

Pl.: (Carl David Tolmé Runge, German, 1856–1927) Let us choose an equidistant partition of the interval $[-5, 5]$ and let us interpolate the function

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

in these points! Apparently, the interpolation polynomials do not tend to $f$. The difference is particularly emphasized at the two ends of the interval.

**Thm. 107.** Let $x$ be in the interval determined by the nodal points $x_0, \ldots, x_n$. Then the estimation

$$|w_{n+1}(x)| \leq \frac{n!}{4} h^{n+1}$$

is true for the nodal polynomial, where $h$ is the greatest difference between the adjacent points.

Proof: If $x$ is a nodal point, then the statement is trivially true.
Let $x \in (x_0, x_1)$. Then the maximizer of $|(x - x_0)(x - x_1)|$ is $x = (x_0 + x_1)/2$.

$$|(x - x_0)(x - x_1)| \leq \left| \frac{x_1 - x_0}{2} \right| \cdot \left| \frac{x_0 - x_1}{2} \right| \leq \frac{h^2}{4}.$$

Thus

$$|w_{n+1}(x)| \leq \frac{h^2}{4} \cdot 2h \cdot 3h \cdot \ldots \cdot nh = \frac{h^{n+1}}{4} n!.$$

# Interpolation error

Let $x \in (x_1, x_2)$. Then the maximizer of $|(x - x_1)(x - x_2)|$ is $x = (x_1 + x_2)/2$.

$$|(x - x_1)(x - x_2)| \leq \left| \frac{x_1 - x_2}{2} \right| \cdot \left| \frac{x_2 - x_1}{2} \right| \leq \frac{h^2}{4}.$$

Thus

$$|w_{n+1}(x)| \leq 2h \cdot \frac{h^2}{4} \cdot 2h \cdot 3h \cdot \ldots \cdot (n-1)h = \frac{h^{n+1}}{4} \frac{2n!}{n} \leq \frac{h^{n+1}}{4} n!.$$

Etc. ■

Rmk. The estimations for the inner sub-intervals are less then that for the outer sub-intervals. Thus we can expect that if we choose the nodal point denser close to the ends of the interval, then the interpolation error can be decreased.

Rmk. Independently of $x$, we have

$$|E_n(x)| \leq \frac{M_{n+1}}{4(n+1)} h^{n+1}.$$

# Chebyshev polynomials

# Chebyshev polynomials


Pafnuty Lvovich Chebyshev, Russian, 1821-1894

Let us consider the polynomials defined with the recursion

$$T_0(x) = 1,\ T_1(x) = x,\ T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

on the interval $[-1, 1]$.

Example. $T_2(x) = 2x^2 - 1$, $T_3(x) = 4x^3 - 3x$.

**Thm. 108.**

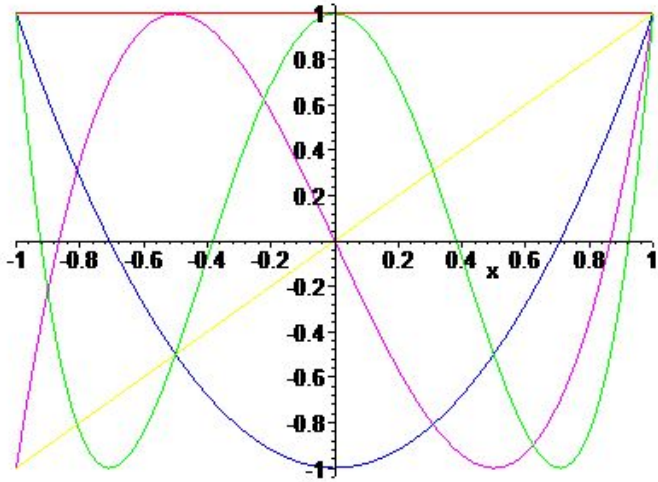
$$T_n(x) = \cos(n \cdot \arccos x).$$

Proof: The statement is trivial for $n = 0$ and $n = 1$. Let us assume that the statement is true for $n = k$. Then

$$\begin{aligned}
&2x\cos(k\arccos x) - \cos((k-1)\arccos x) \\
&= 2x\cos(k\arccos x) \\
&-(\cos(k\arccos x)x + \sin(k\arccos x)\sin(\arccos x)) \\
&= x\cos(k\arccos x) - \sin(k\arccos x)\sin(\arccos x) \\
&= \cos(\arccos x)\cos(k\arccos x) - \sin(k\arccos x)\sin(\arccos x) \\
&= \cos((k+1)\arccos x).
\end{aligned}$$

Thus the statement is true also for $k + 1$. ■

# Chebyshev polynomials



**Thm. 109.**

$$|T_n(x)| \leq 1,$$

moreover the leading coefficient of $T_n(x)$ is $2^{n-1}$.

Proof: Trivial. ■

**Thm. 110.** Let $\tilde{T}_n(x) = T_n(x)/2^{n-1}$, that is we norm the Chebyshev polynomial to leading coefficient 1. Then

$$\|\tilde{T}_n\|_{C[-1,1]} \leq \|p_n^{(1)}\|_{C[-1,1]},$$

where $p_n^{(1)}$ is an arbitrary polynomial with degree at most $n$ and normed to leading coefficient 1.

# Chebyshev polynomials

Proof: The extremizers of $T_n(x)$ are the points $t_k^{(n)} = \cos(k\pi/n)$ $(k = 0, \ldots, n)$. Indeed, $T_n(t_k^{(n)}) = \cos(n \arccos(t_k^{(n)})) = \cos(k\pi) = \pm 1$ (alternately). Thus, these points are the extremizers also of $\tilde{T}_n$.

We use reduction to absurdity. Thus, let us suppose that $\exists p_n^{(1)}$, such that

$$\|p_n^{(1)}\|_{C[-1,1]} < \|\tilde{T}_n\|_{C[-1,1]}.$$

Then the polynomial $q(x) = \tilde{T}_n(x) - p_n^{(1)}$ has degree at most $n-1$ and the sign of this polynomial is the same as that of the original polynomial. The polynomial $q(x)$ should change sign $n$ times, which contradicts to the fact that the polynomial has degree at most $n-1$. ■

# Chebyshev polynomials

Rmk.

$$|E_n(x)| = \frac{|f^{(n+1)}(\xi_x)|}{(n+1)!} |\underbrace{(x-x_0)(x-x_1)\dots(x-x_n)}_{=\tilde{T}_{n+1}(x)}|$$

Let us choose the nodal points to be the roots of the polynomial $T_{n+1}(x)$, that is the values

$$z_k = \cos\left(\frac{(2k+1)\pi}{2(n+1)}\right),\ k = 0, \dots, n!$$

In this case we have

$$|E_n(x)| \leq \frac{M_{n+1}}{(n+1)!}\frac{1}{2^n}$$

independently of $x$.

# Newton interpolation

# Newton form of the interpolation polynomial

Let the nodes $(x_i, f_i)$ $(i = 0, \ldots, n)$ be given. Let us search for the interpolation polynomial in the so-called Newton form:

$$\begin{aligned} p_n(x) = & \, c_0 \\ & + c_1(x - x_0) \\ & + c_2(x - x_0)(x - x_1) \\ & + \ldots \\ & + c_n(x - x_0) \ldots (x - x_{n-1}). \end{aligned}$$

Rmk. This is a polynomial of degree at most $n$. Because the terms are linearly independent, all polynomials with degree at most $n$ can be uniquely written in this form. Thus the coefficients $c_k$ $(k = 0, \ldots, n)$ are uniquely determined.

# Newton's divided differences

**Def. 111.** Let be given a function $f$ and the nodal points $y_0, \ldots, y_k$. Then the uniquely defined leading coefficient of the interpolation polynomial defined by the points $(y_0, f(y_0)), \ldots, (y_k, f(y_k))$ is called Newton's divided difference of order $k$. Notation: $[y_0, \ldots, y_k]f$.

Rmk. Trivially, we have $[y_i]f = f(y_i)$.

Rmk. $[y_0, \ldots, y_k]f$ is uniquely defined and does not depend on the order of the nodal points $y_0, \ldots, y_k$.

**Thm. 112.** If $L_{k-1}$ is the interpolation polynomial defined by the points $(x_0, f_0), \ldots, (x_{k-1}, f_{k-1})$ and $L_k$ is the interpolation polynomial defined by the points $(x_0, f_0), \ldots, (x_k, f_k)$, then the relation

$$L_k(x) = L_{k-1}(x) + [x_0, \ldots, x_k]f \cdot (x - x_0) \ldots (x - x_{k-1})$$

is true.

# Newton's divided differences

Proof: $L_k - L_{k-1}$ is a polynomial of degree at most $k$, and it takes zero value at the points $x_0, \ldots, x_{k-1}$. Moreover its leading coefficient is the same as that of $L_k$: $[x_0, \ldots, x_k]f$. These conditions determine the polynomial

$$L_k(x) - L_{k-1}(x) = [x_0, \ldots, x_k]f \cdot (x - x_0) \ldots (x - x_{k-1})$$

uniquely, which gives the statement of the theorem. ■

Corollary: Based on the previous theorem, the $c_k$ coefficients of the Newton form of the interpolation polynomial can be calculated as $c_k = [x_0, \ldots, x_k]f$.

**Thm. 113.** The Newton's divided differences fulfil the recursion formula

$$[x_0, \ldots, x_k]f = \frac{[x_1, \ldots, x_k]f - [x_0, \ldots, x_{k-1}]f}{x_k - x_0}.$$

Proof: Let us denote the interpolation polynomial defined by the points $(x_1, f_1), \ldots, (x_k, f_k)$ by $q_{k-1}$. Then

$$L_k(x) = \frac{x - x_0}{x_k - x_0} q_{k-1}(x) + \frac{x_k - x}{x_k - x_0} L_{k-1}(x).$$

Indeed, this is a polynomial of degree at most $k$ and $L_k(x_i) = f_i$ $(i = 0, \ldots, k)$. The statement of the theorem follows from the comparison of the leading coefficients. We have

$$\text{leading coef. of } L_k = \frac{\text{leading coef. of } q_{k-1}}{x_k - x_0} - \frac{\text{leading coef. of } L_{k-1}}{x_k - x_0},$$

that is

$$[x_0, \ldots, x_k]f = \frac{[x_1, \ldots, x_k]f - [x_0, \ldots, x_{k-1}]f}{x_k - x_0}.$$

# Newton form of the interpolation polynomial

**Calculation of the coefficients $c_k$:**

By the definition we have: $[x_i]f = f_i$ $(i = 0, \ldots, n)$. According to the recursion formula:

$$[x_0, x_1]f = \frac{[x_1]f - [x_0]f}{x_1 - x_0}, \; [x_1, x_2]f = \frac{[x_2]f - [x_1]f}{x_2 - x_1},$$

$$[x_0, x_1, x_2]f = \frac{[x_1, x_2]f - [x_0, x_1]f}{x_2 - x_0}, \text{ etc.}$$

Example. Find the interpolation polynomials to the points (0,2), (1,1) és (3,5)! We construct a so-called Newton's divided difference table:

| $x_i$ | $f_i = [x_i]f$ | $[.,.]f$ | $[.,.,.]f$ |
|---|---|---|---|
| 0 | $2 = c_0$ | | |
| | | $-1 = c_1$ | |
| 1 | 1 | | $1 = c_2$ |
| | | 2 | |
| 3 | 5 | | |

# Newton form of the interpolation polynomial

Thus the interpolation polynomial has the form:

$$2 + (-1)(x - 0) + 1(x - 0)(x - 1) = x^2 - 2x + 2.$$

For the calculation of the substitution value at a fixed point $x$ we can use a Horner's scheme like rewriting:

$$2 + (-1)(x - 0) + 1(x - 0)(x - 1) = (1(x - 1) + (-1))(x - 0) + 2.$$

Generally:

$$\begin{aligned} L_n(x) = (([x_0, \ldots, x_n]f \cdot (x - x_{n-1}) \\ +[x_0, \ldots, x_{n-1}]f) \cdot (x - x_{n-2}) \\ +[x_0, \ldots, x_{n-2}]f) \cdot (x - x_{n-3}) \ldots + [x_0]f. \end{aligned}$$

# Newton form of the interpolation polynomial

Addition of new nodes is easy: the new table:

| $x_i$ | $f_i = [x_i]f$ | $[.,.]f$ | $[.,.,.]f$ | $[.,.,.,.]f$ |
|---|---|---|---|---|
| $0$ | $2 = c_0$ | | | |
| | | $-1 = c_1$ | | |
| $1$ | $1$ | | $1 = c_2$ | |
| | | $2$ | | $1/2 = c_3$ |
| $3$ | $5$ | | $1/2$ | |
| | | $1$ | | |
| $-1$ | $1$ | | | |

Thus the interpolation polynomial:

$$2 + (-1)(x-0) + 1(x-0)(x-1) + 1/2(x-0)(x-1)(x-3)$$

$$= x^3/2 - x^2 - x/2 + 2.$$

# Comparison of Lagrange's and Newton's formulas

**Lagrange**

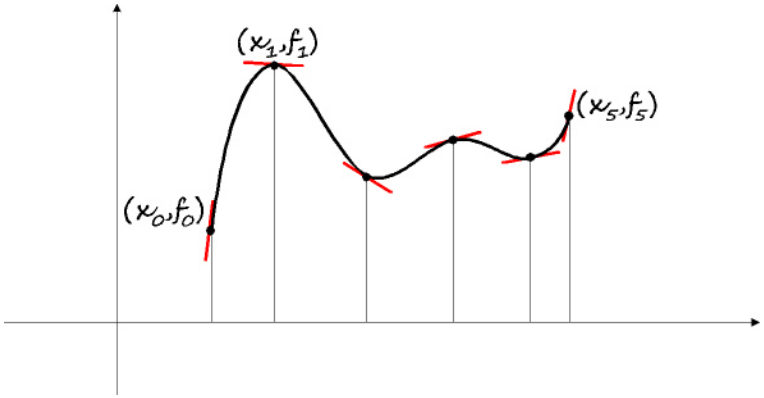
- Less accurate.
- The calculation of $p_n(x)$ for a fixed $x$ costs $4n^2$ flop.
- Addition of new nodes is complicated.
- The characteristic Lagrange polynomials $l_k(x)$ are independent of the values $f_k$. Thus, if these values change, then the new interpolation polynomial can be obtained easily.

**Newton**

- More accurate.
- $3n^2/2$ flop is the calculation of the divided differences and additional $3n$ flop is the calculation of the function values..
- Addition of new nodes is easy.
- When the function values change, the polynomial must be newly calculated.

# Hermite interpolation

# Hermite interpolation



# Hermite interpolation

Let the different nodal points $x_0, \ldots, x_n$ be given together with the function and derivative values

$$f_0^{(0)}, f_0^{(1)}, \ldots, f_0^{(m_0)}; \ldots; f_n^{(0)}, f_n^{(1)}, \ldots, f_n^{(m_n)}.$$

We would like to find the polynomial $p(x)$ that satisfies the conditions

$$p^{(i)}(x_k) = f_k^{(i)}, \ k = 0, \ldots, n; \ i = 0, \ldots, m_k.$$

We have altogether $m_0 + 1 + m_1 + 1 + \ldots m_n + 1 = n + 1 + \sum_{k=0}^{n} m_k =: N$ data. Thus, we can expect that a polynomial with degree at most $N - 1$ will be sufficient.

**Thm. 114.** There exists a unique polynomial $H_{N-1}$ with degree at most $N - 1$ that satisfies the conditions

$$H_{N-1}^{(i)}(x_k) = f_k^{(i)}, \ k = 0, \ldots, n; \ i = 0, \ldots, m_k.$$

# Hermite interpolation

Proof: Let $H_{N-1}(x) = a_0 + a_1 x + \ldots + a_{N-1} x^{N-1}$. Then we have to solve the SLAE:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^{N-1} \\ 0 & 1 & 2x_0 & \ldots & (N-1)x_0^{N-2} \\ \vdots & \vdots & \vdots & \ldots & \vdots \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} f_0^{(0)} \\ f_0^{(1)} \\ f_0^{(2)} \\ \vdots \end{bmatrix}$$

We have here $N$ equations and $N$ unknowns, and the coefficient matrix is non-singular. Indeed, if a non-zero vector existed such that its product with the matrix is a non-zero vector, then the polynomial $H_{N-1}$ would have $N$ roots, which is impossible. ■

Hermite–Fejér interpolation polynomial: At each point only the function value and the derivative are given ($m_k = 1$, $k = 0, \ldots, n$). The the interpolation polynomial has degree at most $2n + 1$.

# Hermite–Fejér interpolation

**Construction of the interpolation polynomial with divided differences:**

$$[x_0, x_1]f = \frac{f(x_0)}{(x_0 - x_1)} + \frac{f(x_1)}{(x_1 - x_0)}$$

Let $x_1 = x_0 + h$ and suppose that $h \to 0$. Then

$$\lim_{h\to 0}[x_0, x_0 + h]f = \lim_{h\to 0}\left(-\frac{f(x_0)}{h} + \frac{f(x_0 + h)}{h}\right) = f'(x_0).$$

Example. $x_0 = 0$, $x_1 = 1$, $f_0^{(0)} = 0$, $f_0^{(1)} = 0$, $f_1^{(0)} = 1$ és $f_1^{(1)} = 3$.

| $x_i$ | $f_i = [x_i]f$ | $[.,.]f$ | $[.,.,.]f$ | $[.,.,.,.]f$ |
|---|---|---|---|---|
| 0 | $0 = c_0$ | | | |
| | | $0 = c_1$ | | |
| 0 | 0 | | $1 = c_2$ | |
| | | 1 | | $1 = c_3$ |
| 1 | 1 | | 2 | |
| | | 3 | | |
| 1 | 1 | | | |

$$H_3(x) = x^3$$

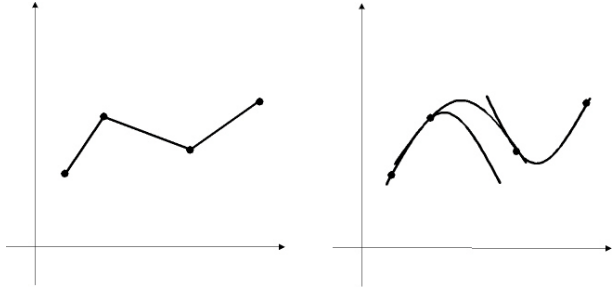
# Spline interpolation

# Spline interpolation - first and second degree splines

Spline = thin and flat bendable wood or metal strip used to draw curves.

When in an interpolation problem the nodes are given, then Chebyshev nodes cannot be used in order to decrease the interpolation error. In this case we generally interpolate with piecewise polynomials of lower degree. (The points that determine the sub-interals are called knots.)

Example. First and second degree splines



First-degree splines : interpolation error $= M_2 h^2/8$ ($h$ is the maximum step-size).

# Spline interpolation - cubic splines

Cubic splines. Let us construct a function $s$ defined on the whole interval $[x_0, x_n]$ that possesses the following properties:

- $s(x_k) = f_k$ $(k = 0, \ldots, n)$,
- $g$, $g'$, $g''$ are continuous,
- $s|_{[x_{i-1}, x_i]}$ is an at most cubic polynomial $(i = 1, \ldots, n)$.

The number of data: $4n$.
The number of the conditions: $2n + 2(n-1) = 4n - 2$.

We may choose two parameters arbitrarily:
a) natural cubic spline: $s''(x_0) = s''(x_n) = 0$.
b) clamped cubic spline: the values $s'(x_0)$ and $s'(x_n)$ are fixed.

**Thm. 115.** There is a unique function $s$ that satisfies the above conditions.

# Construction of the natural cubic splines

For the sake of simplicity let $x_k - x_{k-1} = h$ for all $k = 1, \ldots, n$. Let us consider the polynomial $s_k$ that interpolates on the $k$th sub-interval.
Let

$$s_k(x_{k-1}) = f_{k-1},\ s_k(x_k) = f_k, s_k'(x_{k-1}) = d_{k-1},\ s_k'(x_k) = d_k,$$

where $d_{k-1}$ and $d_k$ are the for now unknown derivatives $s'(x_{k-1})$ and $s'(x_k)$. Let us apply the Hermite–Fejér interpolation:

| $x_i$ | $f_i = [x_i]f$ | $[.,.]f$ | $[.,.,.]f$ | $[.,.,.,.]f$ |
|---|---|---|---|---|
| $x_{k-1}$ | $f_{k-1}$ | | | |
| | | $d_{k-1}$ | | |
| $x_{k-1}$ | $f_{k-1}$ | | $\frac{f_k - f_{k-1}}{h^2} - \frac{d_{k-1}}{h}$ | |
| | | $\frac{f_k - f_{k-1}}{h}$ | | $\left(-2\frac{f_k - f_{k-1}}{h^2} + \frac{d_{k-1}+d_k}{h}\right)/h$ |
| $x_k$ | $f_k$ | | $\frac{d_k}{h} - \frac{f_k - f_{k-1}}{h^2}$ | |
| | | $d_k$ | | |
| $x_k$ | $f_k$ | | | |

# Construction of the natural cubic splines

The polynomial $s_k$ and its second derivatives can be obtained. For these we can set $n+1$ equations: $n-1$ equations in the inner points and 2 equations in the end points. In this way we arrive at the SLAE:

$$\frac{h}{3}\begin{bmatrix} 2 & 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & \dots & 0 & 0 & 0 \\ \vdots & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_n \end{bmatrix} = \begin{bmatrix} f_1 - f_0 \\ f_2 - f_0 \\ f_3 - f_1 \\ \vdots \\ f_n - f_{n-1} \end{bmatrix}$$

With the solution of the system for the derivatives $d_k$, the polynomials $s_k$ can be obtained with Hermite–Fejér interpolation.

# Construction of the natural cubic splines

Example. Let us determine the natural cubic interplation of the points (0,1), (1,2) and (2,0)! Az egyenletrendszer

$$\frac{1}{3}\begin{bmatrix} 2 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 2 \end{bmatrix}\begin{bmatrix} d_0 \\ d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ -2 \end{bmatrix}.$$

We obtain that $d_0 = 7/4$, $d_1 = -1/2$, $d_2 = -11/4$ and the cubic polynomials that belong to the sub-intervals:

$$s_1(x) = -\frac{3}{4}x^3 + \frac{7}{4}x + 1,\ s_2(x) = \frac{3}{4}x^3 - \frac{9}{2}x^2 + \frac{25}{4}x - \frac{1}{2}.$$

# Construction of the clamped cubic splines

The SLAE can be obtained similarly to the previous case. Now $d_0$ and $d_n$ are fixed, and modify the system according to this fact.

$$\frac{h}{3}\begin{bmatrix} 4 & 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & \dots & 0 & 0 & 0 \\ \vdots & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} d_1 \\ \vdots \\ d_{n-1} \end{bmatrix} = \begin{bmatrix} f_2 - f_0 - d_0 h/3 \\ f_3 - f_1 \\ \vdots \\ f_{n-1} - f_{n-3} \\ f_n - f_{n-2} - d_n h/3 \end{bmatrix}$$

With the solution of the system for the derivatives $d_k$, the polynomials $s_k$ can be obtained with Hermite–Fejér interpolation.

# Construction of the clamped cubic splines

Example. Let us determine the claped cubic spline interpolation to the points (0,1), (1,2) and (2,0), if $s'(0) = 0$ and $s'(2) = 1$!
Thus, $d_0 = 0$ and $d_2 = 1$. The "SLAE" symplifies to

$$\frac{1}{3}4d_1 = -1 - \frac{1}{3}0 - \frac{1}{3}1,$$

which gives $d_1 = -1$.

The cubic polynomials that belong to the sub-intervals are:

$$s_1(x) = -3x^3 + 4x^2 + 1, \; s_2(x) = 4x^3 - 17x^2 + 21x - 6.$$

# Properties of cubic splines

## Error estimate for cubic splines

**Thm. 116.** Let $f \in C^4[x_0, x_n]$ and let $s$ be the cubic spline interpolating $f$ on an equidistant mesh (with stepsize $h$) $x_0 < x_1 < \ldots < x_n$. Then

$$\|f^{(r)} - s^{(r)}\|_{C[x_0,x_n]} \leq C_r h^{4-r} \|f^{(4)}\|_{C[x_0,x_n]}, \ r = 0, 1, 2, 3,$$

where $C_0 = 5/384$, $C_1 = 1/24$, $C_2 = 3/8$ and $C_3 = 1$.

## Minimum norm property of cubic splines

**Thm. 117.** Let $f \in C^2[a, b]$ and let $s$ be the cubic spline interpolating $f$. Then

$$\int_{x_0}^{x_n} |s''(x)|^2 \, \mathrm{d}x \leq \int_{x_0}^{x_n} |f''(x)|^2 \, \mathrm{d}x,$$

where equality holds iff $f = s$.

# Trigonometric interpolation

# Interpolating trigonometric polynomials

# Trigonometric polynomials

When we know that the data are the values of a periodic function, then it is advisable to interpolate with trigonometric functions instead of polynomials. .

Let us suppose that we know the values $(f_k)$ of a $2\pi$ periodic function at the points $x_k = 2\pi k/(n+1) \in [0, 2\pi)$ $(k = 0, \ldots, n)$, where $n$ is a positive natural number. Let us search for the interpolating function in the form

$$t_m(x) = a_0 + \sum_{j=1}^{m} (a_j \cos(jx) + b_j \sin(jx)),$$

which has to satisfy the equalities $t_m(x_k) = f_k$ $(k = 0, \ldots, n)$. $t_m$ is called trigonometric polynomial of $m$th degree.

# Trigonometric polynomials

Thus we have $2m+1$ coefficients and $n+1$ equations.

- If $n$ is even, then we can expect that a polynomial with degree $m = n/2$ will be suitable.
- If $n$ is odd, then introduce the notation $m = (n+1)/2$. Then we have $n+2$ coefficients and $n+1$ equations, that is the system is underdetermined. The term with the coefficient $b_m$ has the following values at the nodes:

$$b_m \sin(mx_k) = b_m \sin\left(\frac{n+1}{2}\frac{2\pi k}{n+1}\right) = b_m \sin(\pi k) = 0.$$

Hence, the value of $b_m$ can be chosen to be zero. We say that in this case the trigonometric polynomial (in the case if $n$ is odd) is balanced.

# Interpolation with trigonometric polynomials

**Thm. 118.** Let us suppose that the $f_k$ values $(k = 0, \ldots, n)$ are given at the nodes $x_k = 2\pi k/(n+1)$. Let us suppose that $n$ is odd. Then there exists a unique balanced trigonometric polynomial of degree $m = (n+1)/2$ denoted by $t_m$ that satisfies the interpolation condition $t_m(x_k) = f_k$ $(k = 0, \ldots, n)$.

Proof: We will construct the polynomial. We work with complex numbers. Using the equality $e^{\mathtt{i}\phi} = \cos\phi + \mathtt{i}\sin\phi$ we obtain that

$$e^{\mathtt{i}jx} = \cos(jx) + \mathtt{i}\sin(jx),\ e^{-\mathtt{i}jx} = \cos(jx) - \mathtt{i}\sin(jx),$$

which results in the formulas

$$\cos(jx) = \frac{e^{\mathtt{i}jx} + e^{-\mathtt{i}jx}}{2}, \quad \sin(jx) = \frac{e^{\mathtt{i}jx} - e^{-\mathtt{i}jx}}{2\mathtt{i}}.$$

# Interpolation with trigonometric polynomials

After back substitution to the original polynomial $t_m$ and with the use of the interpolation property we obtain that

$$f_k = t_m(x_k) = a_0 + \sum_{j=1}^{m} \left( a_j \frac{e^{\mathtt{i}jx_k} + e^{-\mathtt{i}jx_k}}{2} + b_j \frac{e^{\mathtt{i}jx_k} - e^{-\mathtt{i}jx_k}}{2\mathtt{i}} \right)$$

$$= \overbrace{a_0}^{=:c_0} + \sum_{j=1}^{m-1} \left( \overbrace{\frac{a_j - b_j\mathtt{i}}{2}}^{=:c_j} e^{\mathtt{i}jx_k} + \overbrace{\frac{a_j + b_j\mathtt{i}}{2}}^{=:c_{2m-j}} e^{-\mathtt{i}jx_k} \right)$$

$$+ \underbrace{\overbrace{\frac{a_m}{2}}^{=:c_m} e^{\mathtt{i}mx_k} + \overbrace{\frac{a_m}{2}}^{=:c_m} e^{-\mathtt{i}mx_k}}_{c_m e^{\mathtt{i}mx_k}}$$

$$= \sum_{j=0}^{n} c_j e^{\mathtt{i}jx_k}, \quad k = 0, \ldots, n.$$

# Interpolation with trigonometric polynomials

We applied the equality

$$e^{\mathtt{i}mx_k} = e^{-\mathtt{i}mx_k} = (-1)^k.$$

The original real coefficients can be calculated with the complex coefficients $c_j$:

$$a_0 = c_0, \quad a_m = c_m, \quad a_j = c_j + c_{2m-j} \ (j = 1, \ldots, m-1),$$

$$b_j = \mathtt{i}(c_j - c_{2m-j}) \ (j = 1, \ldots, m-1).$$

Because $f_k \in \mathbb{R}$, taking the complex conjugate of both sides we arrive at the form

$$f_k = \overline{f}_k = \sum_{j=0}^{n} \overline{c}_j e^{-\mathtt{i}jx_k}, \quad k = 0, \ldots, n,$$

that is $c_0, \ c_m \in \mathbb{R}$ és $c_{2m-j} = \overline{c}_j$, thus $a_j = 2\mathrm{Re}(c_j)$ és $b_j = -2\mathrm{Im}(c_j)$ $(j = 1, \ldots, m-1)$.

# Interpolation with trigonometric polynomials

Let us introduce the notation

$$w = e^{-\mathtt{i}2\pi/(n+1)}.$$

$w$ is a $(n+1)$th root of unity, because $w^{n+1} = 1$. Moreover,

$$e^{-\mathtt{i}jx_k} = w^{jk}$$

and using this notation we have to solve the SLAE

$$f_k = \sum_{j=0}^{n} c_j w^{-jk}, \ k = 0, \ldots, n$$

for the coefficients $c_j$. We show that this SLAE always has a unique solution, which fact shows that the trigonometric interpolation polynomial is unique.

# Interpolation with trigonometric polynomials

With the notations

$$\overline{\mathbf{f}}_{n+1} = [f_0, \ldots, f_n]^T,\ \overline{\mathbf{c}}_{n+1} = [c_0, \ldots, c_n]^T,$$

$$\mathbf{F}_{n+1} \in \mathbb{R}^{(n+1)\times(n+1)},\ (\mathbf{F}_{n+1})_{jk} = w^{jk}$$

the SLAE can be written in the form

$$\overline{\mathbf{f}}_{n+1} = \mathbf{F}_{n+1}^H \overline{\mathbf{c}}_{n+1}.$$

Lemma. $\mathbf{F}_{n+1}\mathbf{F}_{n+1}^H = (n+1)\mathbf{I}_{n+1}$

Proof:

$$(\mathbf{F}_{n+1}\mathbf{F}_{n+1}^H)_{kj} = \sum_{s=0}^{n} w^{ks} w^{-js} = \sum_{s=0}^{n} w^{s(k-j)} =$$

$$= \begin{cases} \text{n+1}, & \text{if } j = k, \\ \frac{(w^{k-j})^{n+1}-1}{w^{k-j}-1} = 0, & \text{if } j \neq k. \end{cases} \blacksquare$$

# Interpolation with trigonometric polynomials

Let us return to the proof of the theorem. Let us multiply both sides of the SLAE

$$\overline{\mathbf{f}}_{n+1} = \mathbf{F}_{n+1}^{H}\overline{\mathbf{c}}_{n+1}$$

by the matrix $\mathbf{F}_{n+1}$. We obtain

$$\mathbf{F}_{n+1}\overline{\mathbf{f}}_{n+1} = \mathbf{F}_{n+1}\mathbf{F}_{n+1}^{H}\overline{\mathbf{c}}_{n+1},$$

that is the coefficients $c_j$ can be calculated uniquely as

$$\overline{\mathbf{c}}_{n+1} = \frac{1}{n+1}\mathbf{F}_{n+1}\overline{\mathbf{f}}_{n+1}.\blacksquare$$

Let us introduce the notation $\hat{\mathbf{f}}_{n+1} := (n+1)\overline{\mathbf{c}}_{n+1}$.

# Interpolation with trigonometric polynomials

**Fourier analysis (Discrete Fourier Transform - DFT):** We calculate the $c_j$ complex Fourier coefficients from the data

$$\hat{f}_j = (n+1)c_j = \sum_{k=0}^{n} f_k w^{kj}, \quad j = 0, \ldots, n.$$

**Fourier synthesis (Inverse Discrete Fourier Transform - IDFT):** We calculate the nodal function values by the help of the Fourier coefficients $c_j$.

$$\frac{1}{n+1} \sum_{j=0}^{n} \hat{f}_j w^{-jk} = f_k, \quad k = 0, \ldots, n.$$

# Interpolation with trigonometric polynomials

Rmk. If the function values $f_k$ are real then $c_{2m-j} = \overline{c}_j$ $(j = 1, \ldots, m-1)$, that is these coefficients are complex conjugate of each other, $a_0 = c_0$ and $a_m = c_m$ are real values. Thus $a_j = 2\mathrm{Re}(c_j)$ and $b_j = -2\mathrm{Im}(c_j)$. From this, we obtain

$$a_0 = \frac{1}{n+1}\sum_{k=0}^{n} f_k, \quad a_m = \frac{1}{n+1}\sum_{k=0}^{n} f_k \cos(mx_k),$$

$$a_j = \frac{2}{n+1}\sum_{k=0}^{n} f_k \cos(jx_k) \quad (j = 1, \ldots, m-1),$$

$$b_j = \frac{2}{n+1}\sum_{k=0}^{n} f_k \sin(jx_k) \quad (j = 1, \ldots, m-1).$$

# Interpolation with trigonometric polynomials

When the number of nodes is odd, then a similar theorem can be proven. The proof is also similar. .

**Thm. 119.** Let us suppose that the function values $f_k$ $(k = 0, \ldots, n)$ are given at the nodes $x_k = 2\pi k/(n+1)$. Let us suppose that $n$ is even. Then, there exists a unique trigonometric polynomial $t_m$ with degree $m = n/2$ such that $t_m(x_k) = f_k$ $(k = 0, \ldots, n)$.

Corollary: In this case the real discrete Fourier coefficients can be calculated as follows

$$a_0 = \frac{1}{n+1} \sum_{k=0}^{n} f_k,$$

$$a_j = \frac{2}{n+1} \sum_{k=0}^{n} f_k \cos(jx_k) \quad (j = 1, \ldots, m),$$

$$b_j = \frac{2}{n+1} \sum_{k=0}^{n} f_k \sin(jx_k) \quad (j = 1, \ldots, m).$$

# Interpolation with trigonometric polynomials

Rmk. Let $f$ be a $2\pi$ periodic function. Let us search the function in the so-called Fourier series form

$$f(x) = \alpha_0 + \sum_{j=1}^{\infty} (\alpha_j \cos(jx) + \beta_j \sin(jx)).$$

Then it can be shown that

$$\alpha_0 = \frac{1}{2\pi} \int_0^{2\pi} f(x)\,\mathrm{d}x,$$

$$\alpha_j = \frac{1}{\pi} \int_0^{2\pi} f(x)\cos(jx)\,\mathrm{d}x$$

$$\beta_j = \frac{1}{\pi} \int_0^{2\pi} f(x)\sin(jx)\,\mathrm{d}x.$$

Let us notice that the discrete Fourier coefficients are the approximations of the integrals above.

# Interpolation with trigonometric polynomials

Example. $\overline{\mathbf{f}} = [0, 1, 4, 9]^T$, $n = 3$, $m = (n+1)/2 = 2$, $w = e^{-2i\pi/4} = -i$.

$$\hat{\overline{\mathbf{f}}}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & w & w^2 & w^3 \\ 1 & w^2 & w^4 & w^6 \\ 1 & w^3 & w^6 & w^9 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 4 \\ 9 \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -\mathtt{i} & -1 & \mathtt{i} \\ 1 & -1 & 1 & -1 \\ 1 & \mathtt{i} & -1 & -\mathtt{i} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 4 \\ 9 \end{bmatrix} = \begin{bmatrix} 14 \\ -4+8\mathtt{i} \\ -6 \\ -4-8\mathtt{i} \end{bmatrix}.$$

Thus $a_0 = 14/4 = 7/2$, $a_1 = -8/4 = -2$, $b_1 = -16/4 = -4$, $a_2 = -6/4 = -3/2$.

# Fast Fourier transform

# Fast Fourier transform (FFT)

The procedure was given already by Gauss in the early 1800s, but his work has been forgotten. After the advent of the computers the method was newly rediscoverd. James W. Cooley (IBM), John W. Tukey (Princeton), 1965.

$$\hat{f}_j = \sum_{k=0}^{n} f_k w^{kj}, \quad j = 0, \ldots, n.$$

The calculation of the discrete Fourier coefficients requires approximately $(n+1)^2$ complex multiplications, provided that the powers of $w$ have been computed already (each coefficient requires $n+1$ multiplications).

**How could we determine these coefficient with much less effort using the special form of the elements of the matrix.**

# Fast Fourier transform (FFT)

Example. In the previous problem we need to calculate the multiplication

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -\mathtt{i} & -1 & \mathtt{i} \\ 1 & -1 & 1 & -1 \\ 1 & \mathtt{i} & -1 & -\mathtt{i} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 4 \\ 9 \end{bmatrix}.$$

Let us swap the columns of the matrix in order to put the odd numbered columns to the "left part" of the matrix!

$$\left[\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & -1 & -\mathtt{i} & \mathtt{i} \\ \hline 1 & 1 & -1 & -1 \\ 1 & -1 & \mathtt{i} & -\mathtt{i} \end{array}\right] \left[\begin{array}{c} 0 \\ 4 \\ \hline 1 \\ 9 \end{array}\right]$$

Here the two blocks on the left hand side is $\mathbf{F}_2$, the lower right block is the opposite of the upper right one, and the upper right block is

$$\begin{bmatrix} 1 & 0 \\ 0 & w \end{bmatrix} \mathbf{F}_2 = \begin{bmatrix} 1 & 0 \\ 0 & -\mathtt{i} \end{bmatrix} \mathbf{F}_2.$$

# Fast Fourier transform (FFT)

In fact, we have to calculate only the product of the matrix $\mathbf{F}_2$ with the vectors $[0,4]^T$ and $[1,9]^T$, moreover the elements of the last product must be multiplied with the powers of $w$ ($w^0, w^1, w^2, \ldots, w^{m-1}$), respectively.

General case: Let us suppose that $n+1$ is an even number. Then we need to perform the multiplication

$$\begin{bmatrix} \hat{f}_0 \\ \hat{f}_1 \\ \hat{f}_2 \\ \vdots \\ \hat{f}_{m-1} \\ \hat{f}_m \\ \hat{f}_{m+1} \\ \vdots \\ \hat{f}_n \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & w & w^2 & \ldots & w^n \\ 1 & w^2 & w^4 & \ldots & w^{2n} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & w^{m-1} & w^{2(m-1)} & \ldots & w^{(m-1)n} \\ 1 & w^m & w^{2m} & \ldots & w^{mn} \\ 1 & w^{(m+1)} & w^{2(m+1)} & \ldots & w^{(m+1)n} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & w^n & w^{2n} & \ldots & w^{n^2} \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}.$$

# Fast Fourier transform (FFT)

Let us change the odd numbered columns forward!
Then the elements of the vector $\overline{\mathbf{f}}$ will be also rearranged.

We obtain the product:

$$\left[\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & w^2 & \dots & w^{n-1} & w & w^3 & \dots & w^n \\ 1 & w^4 & \dots & w^{2(n-1)} & w^2 & w^6 & \dots & w^{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & w^{2(m-1)} & \dots & w^{(m-1)(n-1)} & w^{m-1} & w^{3(m-1)} & \dots & w^{(m-1)n} \\ \hline 1 & w^{2m} & \dots & w^{m(n-1)} & w^m & w^{3m} & \dots & w^{mn} \\ 1 & w^{2(m+1)} & \dots & w^{(m+1)(n-1)} & w^{(m+1)} & w^{3(m+1)} & \dots & w^{(m+1)n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & w^{2n} & \dots & w^{n(n-1)} & w^n & w^{3n} & \dots & w^{n^2} \end{array}\right] \left[\begin{array}{c} f_0 \\ f_2 \\ f_4 \\ \vdots \\ f_{n-1} \\ \hline f_1 \\ f_3 \\ \vdots \\ f_n \end{array}\right].$$

The upper left block is $\mathbf{F}_m$ because $w^2$ is an $m$th root of unity. The lower left block is also $\mathbf{F}_m$. This can be checked easily using the fact that $w$ is an $(n+1)$th root of unity. The upper right block can be written in the form $\mathbf{D}_m\mathbf{F}_m$ with the notation $\mathbf{D}_m = \mathrm{diag}(1, w, w^2, \dots, w^{m-1})$. The lower right block is the opposite of this.

# Fast Fourier transform (FFT)

When we partition the vector $\hat{\mathbf{f}}$ and the rearranged $\overline{\mathbf{f}}$ (denoted by $\tilde{\mathbf{f}}$) accordingly, the product can be written in the form

$$\begin{bmatrix} \hat{\mathbf{f}}_1 \\ \hat{\mathbf{f}}_2 \end{bmatrix} = \mathbf{F}_{n+1}\overline{\mathbf{f}} = \begin{bmatrix} \mathbf{I}_m & \mathbf{D}_m \\ \mathbf{I}_m & -\mathbf{D}_m \end{bmatrix} \begin{bmatrix} \mathbf{F}_m & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_m \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{f}}_1 \\ \tilde{\mathbf{f}}_2 \end{bmatrix}.$$

What can we win compared to the $(n+1)^2$ complex multiplication?
$\mathbf{F}_m\tilde{\mathbf{f}}_1$ and $\mathbf{F}_m\tilde{\mathbf{f}}_2$ require $((n+1)/2)^2$ complex multiplications each. The product of the diagonal matrix $\mathbf{D}_m$ and the vector $\mathbf{F}_m\tilde{\mathbf{f}}_2$ requires $(n+1)/2$ complex multiplicaitons. We do not need more multiplications. We must perform

$$2\left(\frac{n+1}{2}\right)^2 + \frac{n+1}{2}$$

complex multiplications.

# Fast Fourier transform (FFT)

The algorithm become really fast if we use the above procedure in the case of the half sized matrices, too. This can be done repeatably if $n+1$ is a power of 2.
Let $Q_l$ denote the number of complex multiplications of FFT when we use $2^l$ nodes. Then trivially

$$Q_l = 2Q_{l-1} + 2^{l-1}$$

and taking into the account that $Q_1 = 1$, we obtain with induction that

$$Q_l = l2^{l-1} = \frac{1}{2}(n+1)\log_2(n+1).$$

This is a significant drop in the number of operations:

| $n+1$ | $DFT$ | $FFT$ |
|---|---|---|
| $2^5 = 32$ | 1024 | 80 |
| $2^{10} = 1024$ | 1048576 | 5120 |
| $2^{20} = 1048576$ | 1099511627776 | 10485760 |