Numerical methods

Approximation of functions

OUTLINE

1. Approximation and interpolation
2. Polynomial interpolation
   a. Lagrange polynomial
   b. Newton polynomial
   c. The error of approximation for interpolating polynomial
   d. Optimal distribution of interpolation nodes
   e. Hermite interpolation
3. Spline interpolation
   a. Linear spline interpolation
   b. Hermite cubic spline
   c. Cubic spline
   d. Cubic natural spline

Approximation and interpolation

To approximate function $f(x)$ means

to substitute it by a function $\varphi(x)$,

which is in some sense close to function $f(x)$.

We will deal with two basic types of approximation:
interpolation and **least-square method**

Definition: **Interpolation** is such approximation,

in which the function $\varphi(x)$ goes exactly through given

points $[x_i, y_i]$, where $y_i = f(x_i)$.

Sometimes we also require that

functions f and φ have the same derivatives

in points x_i .

Approximation and interpolation

To approximate function $f(x)$ means

to substitute it by a function $\varphi(x)$,

which is in some sense close to function $f(x)$.

We will deal with two basic types of approximation:
interpolation and **least-square method**

Definition: **Least-square method** is such approximation,

in which $\varphi(x)$ is „interlaced“

between given points $[x_i, y_i]$ in such a way,

that the „distance“ between functions f and φ is

in some sense minimal.

Usually the function $\varphi(x)$ does not go through

points $[x_i, y_i]$.

For example, we use approximation $\varphi(x)$
to approximate calculation of values of function $f(x)$
during the plotting of graph $\varphi(x) \approx f(x)$.

In general, $\varphi(x)$ is used to solve problems,
in which it is practical and sometimes inevitable
to substitute function f
by its approximation φ.

Such an example is computation
of derivative or definite integral.

It is desirable that calculation of $\varphi(x)$ is “simple”.
Therefore $\varphi(x)$ is often seek in the polynomial form.

Interpolation

We chose interpolation function $\varphi(x)$
from a suitable class of functions.
We restrict ourselves to two
the most common cases:

1. $\varphi(x)$ is a polynomial function;

2. $\varphi(x)$ is a piece-wise polynomial,
   i.e. in general different on each subinterval

OUTLINE

Let suppose there are $n+1$ given points

$$x_0, x_1, \dots, x_n, \qquad x_i \neq x_j \quad \text{for} \quad i \neq j,$$

which we call **interpolation nodes**,

and in each node there is given value y_i.

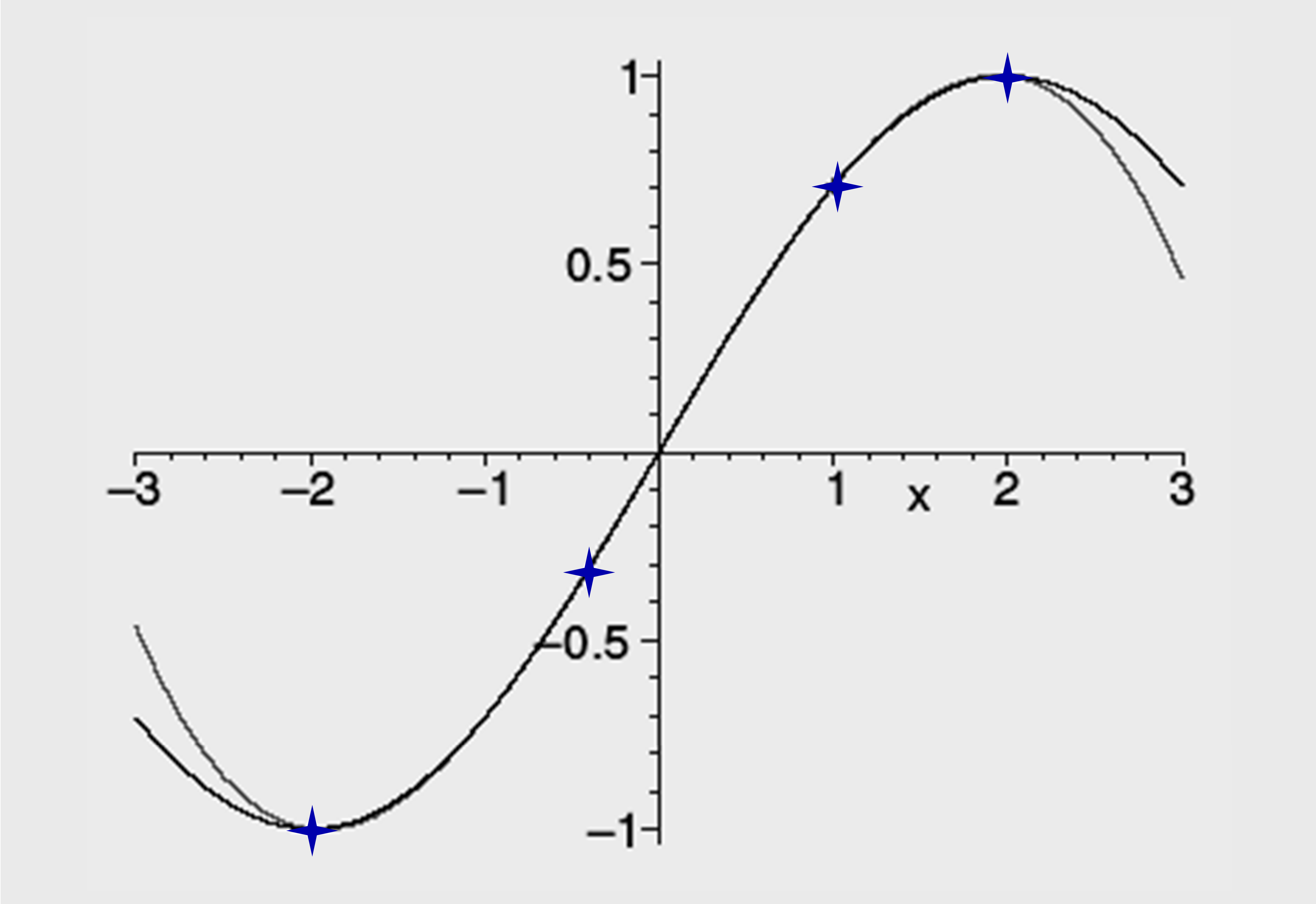
We are looking for **interpolation polynomial** $P_n(x)$

of degree of most n,

which satisfies **interpolation conditions**

$$P_n\left(x_i\right) = y_i, \quad i = 0, 1, \dots, n.$$

Polynomial interpolation



Unisolvence theorem

Lets there is given a set of points $[\, x_i, y_i \,]$, $i = 0, \ldots n$,

where no two x_i are the same.

Then **there exists a unique polynomial** P_n degree at most n such,

that $P_n(x_i) = y_i, \;\; i = 0, \ldots n$.

We prove the existence of interpolation polynomial
in such a way,
that we show its construction
for any mutually different nodal points.

The uniqueness of interpolation polynomial
can be proofed by contradiction.

Suppose that there are two at-most n degree polynomials
$P_n(x)$ and $R_n(x)$ such that
$P_n(x_i) = y_i,\; i = 0, \ldots n$ and also $R_n(x_i) = y_i,\; i = 0, \ldots n$.

We will show, that the two polynomials are equal.

Denote $Q_n(x) = P_n(x) - R_n(x)$.
We see, that $Q_n(x)$ is also at-most n degree polynomial
and moreover $Q_n(x_i) = 0,\; i = 0, \ldots ,n$.
We have at-most n degree polynomial, which has $n+1$ roots.
But this is possible only if $Q_n(x)$ is identically equal to zero, $Q_n(x) = 0$
and tehrefore $P_n(x) = R_n(x) \quad \forall x \in \Re$.

OUTLINE

1. Approximation and interpolation
2. Polynomial interpolation
   a. Lagrange polynomial
   b. Newton polynomial
   c. The error of approximation for interpolating polynomial
   d. Optimal distribution of interpolation nodes
   e. Hermite interpolation
3. Spline interpolation
   a. Linear spline interpolation
   b. Hermite cubic spline
   c. Cubic spline
   d. Cubic natural spline

Interpolation polynomial in Lagrange form is

$$P_n(x) = y_0 l_0(x) + y_1 l_1(x) + \cdots + y_n l_n(x) = \sum_{i=0}^{n} y_i l_i(x) \ ,$$

where $l_i(x)$ are **Lagrange basis polynomials** defined as

$$l_i(x) = \frac{(x-x_0)(x-x_1)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_0)(x_i-x_1)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)} \ .$$

It is easy to see that

$$l_i(x_k) = \begin{cases} 1 & \text{for } \ k = i, \\ 0 & \text{for } \ k \neq i, \end{cases} \qquad i,k = 0,1,\ldots,n \ ,$$

therefore interpolating conditions

$$P_n(x_k) = \sum_{i=0}^{n} y_i l_i(x_k) = y_k, \ \ k = 0,1,\ldots,n \ ,$$

are satisfied.

Example: Find the interpolation polynomial for data given in table

x_i	-1	1	2	3
y_i	-6	-2	-3	2

At first we obtain Lagrange basis polynomials

$$\ell_0(x) = \frac{(x-1)(x-2)(x-3)}{(-1-1)(-1-2)(-1-3)} = -\frac{1}{24}\left(x^3 - 6x^2 + 11x - 6\right),$$

$$\ell_1(x) = \frac{(x+1)(x-2)(x-3)}{(1+1)(1-2)(1-3)} = \frac{1}{4}\left(x^3 - 4x^2 + x + 6\right),$$

$$\ell_2(x) = \frac{(x+1)(x-1)(x-3)}{(2+1)(2-1)(2-3)} = -\frac{1}{3}\left(x^3 - 3x^2 - x + 3\right),$$

$$\ell_3(x) = \frac{(x+1)(x-1)(x-2)}{(3+1)(3-1)(3-2)} = \frac{1}{8}\left(x^3 - 2x^2 - x + 2\right)$$

Lagrange polynomial

x_i	-1	1	2	3
y_i	-6	-2	-3	2

$$\ell_0(x) = \frac{(x-1)(x-2)(x-3)}{(-1-1)(-1-2)(-1-3)} = -\frac{1}{24}\,(x^3 - 6x^2 + 11x - 6)\,,$$

$$\ell_1(x) = \frac{(x+1)(x-2)(x-3)}{(1+1)(1-2)(1-3)} = \frac{1}{4}\,(x^3 - 4x^2 + x + 6)\,,$$

$$\ell_2(x) = \frac{(x+1)(x-1)(x-3)}{(2+1)(2-1)(2-3)} = -\frac{1}{3}\,(x^3 - 3x^2 - x + 3)\,,$$

$$\ell_3(x) = \frac{(x+1)(x-1)(x-2)}{(3+1)(3-1)(3-2)} = \frac{1}{8}\,(x^3 - 2x^2 - x + 2)$$

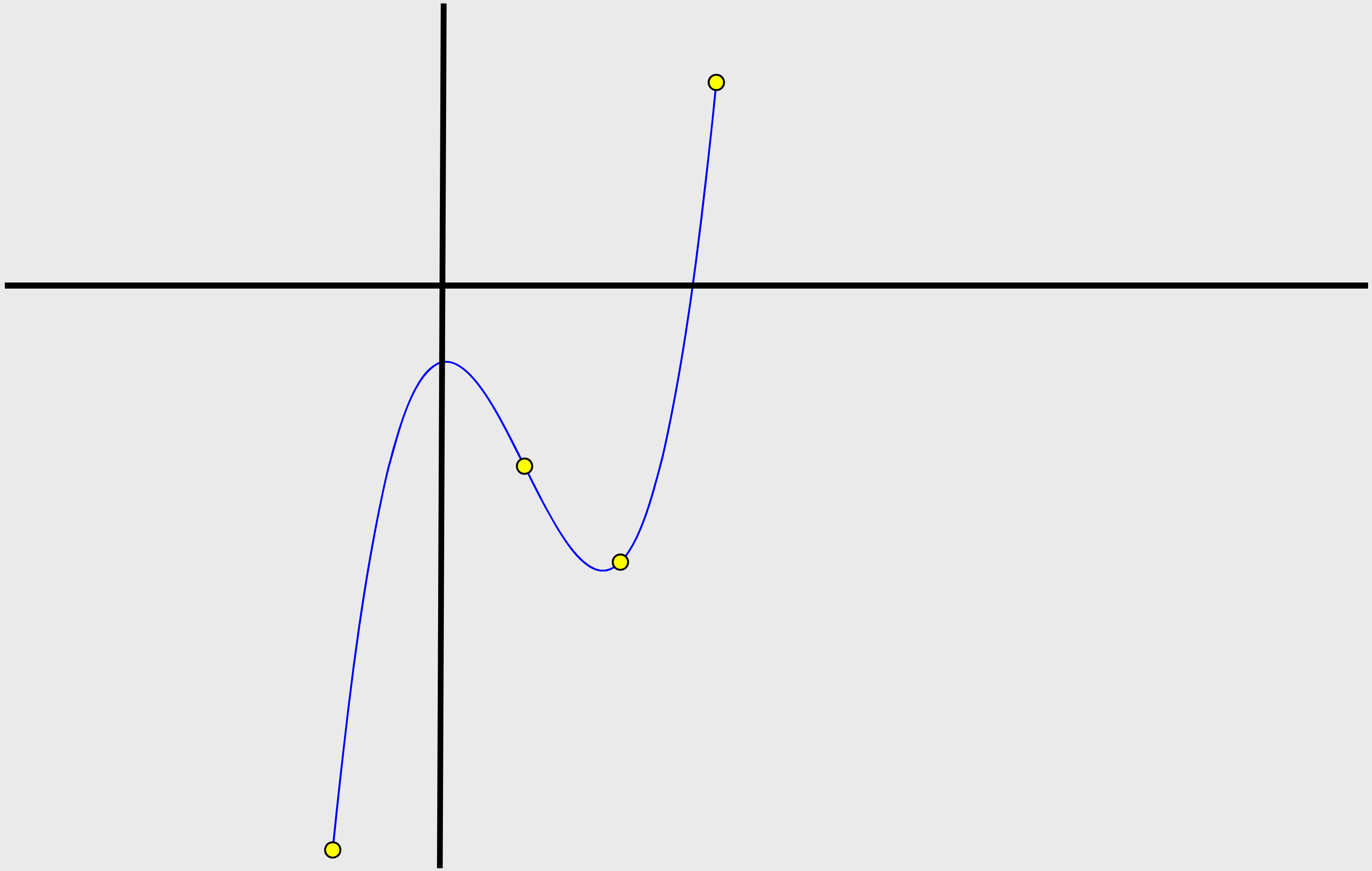
Then we construct the interpolation polynomial

$$P_3(x) = -6\cdot\ell_0(x) - 2\cdot\ell_1(x) - 3\cdot\ell_2(x) + 2\cdot\ell_3(x) = x^3 - 3x^2 + x - 1\,.$$

Lagrange polynomial

x_i	-1	1	2	3
y_i	-6	-2	-3	2

$$P_3(x) = -6 \cdot \ell_0(x) - 2 \cdot \ell_1(x) - 3 \cdot \ell_2(x) + 2 \cdot \ell_3(x) = x^3 - 3x^2 + x - 1\,.$$



Lagrange polynomial

The main advantage of Lagrange polynomial is
its elegant form.
Therefore it is mainly used in theoretical considerations.

It is not ideal for practical use
because it has two main drawbacks

- If we add another node x_{n+1},
  we have to recalculate all Lagrange basis polynomials

- The number of operations needed to calculate values $P_n(x^*)$
  is relatively high,
  it requires $2n^2+2n$ operations of multiplication and
  $2n^2+3n$ operations of addition

OUTLINE

Drawbacks of Lagrange polynomial are eliminated by **Newton polynomial**, which has a form

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)(x - x_1)\cdots(x - x_{n-1})\,.$$

Addition of another node x_{n+1} is easy, it is enough to add next term to the $P_n(x)$ because

$$P_{n+1}(x) = P_n(x) + a_{n+1}(x - x_0)(x - x_1)\cdots(x - x_n)\,.$$

Newton polynomial

The value $z=P_n(x^*)$ can be estimated using **Horner scheme**:

$$z := a_n$$

and then for $i = n-1, n-2, \ldots, 0$ we calculate

$$z := z\,(x^* - x_i) + a_i.$$

This significantly reduces the number of operations.

The coefficients a_i could be computed directly from interpolation conditions

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

There is, however, better way called **Divided-Difference method**.

Newton polynomial

At first define **divided differences:**

$$P[x_i] := y_i\,,$$

$$P[x_i, x_{i+1}] := (P[x_{i+1}] - P[x_i])/(x_{i+1} - x_i)\,,$$

$$P[x_i, x_{i+1}, x_{i+2}] := (P[x_{i+1}, x_{i+2}] - P[x_i, x_{i+1}])/(x_{i+2} - x_i),$$

and for $3 \le k \le n$:

$$P[x_i, x_{i+1}, \ldots, x_{i+k-1}, x_{i+k}] := (P[x_{i+1}, \ldots, x_{i+k}] - P[x_i, \ldots, x_{i+k-1}])/(x_{i+k} - x_i)\,.$$

It is possible to show that

$$a_i = P[x_0, x_1, \ldots, x_i]\,,$$

So the **Newton polynomial** is

$$P_n(x) = P[x_0] + P[x_0,x_1](x-x_0) + P[x_0,x_1,x_2](x-x_0)(x-x_1) + \ldots$$
$$+ \ P[x_0,x_1,\ldots,x_n](x-x_0)(x-x_1)\ldots(x-x_{n-1}).$$

If we denote $P_{ik} = P[x_{i-k},\ldots,x_i]$, then a_i=P_{ii} and the **algorithms of divided-difference method** will be

For i=0,1,...,n do P_{i0}:=y_i.

For k=1,2,...,n do:

for i=k, k+1, ..., n do:

$$P_{ik} := (P_{i,k-1} - P_{i-1,k-1})/(x_i - x_{i-k})$$

end of cycle i ,

end of cycle k .

The calculation could be written in table,
which is filled-in by columns.

$$
\begin{array}{c|cccccc}
x_0 & P_{00} & & & & & \\
x_1 & P_{10} & P_{11} & & & & \\
x_2 & P_{20} & P_{21} & P_{22} & & & \\
x_3 & P_{30} & P_{31} & P_{32} & P_{33} & & \\
\vdots & \vdots & \vdots & \vdots & & \ddots & \\
x_n & P_{n0} & P_{n1} & P_{n2} & \dots & P_{n,n-1} & P_{nn}
\end{array}
$$

Newton polynomial

Computation of coefficients $a_i = P_{ii}$

and

the follow-up computation of $z=P_n(x^*)$ by Horner scheme

requires

$\frac{1}{2}n^2 + \frac{3}{2}n$ operations of multiplication and

$n^2 + 3n$ operations of addition.

It is much less than using Lagrange polynomial $P_n(x^*)$

($2n^2+2n$ operations of multiplication and

$2n^2+3n$ operations of addition)

Example: Construct Newton polynomial for the same data as in previous example

x_i	-1	1	2	3
y_i	-6	-2	-3	2

Progress of computation is stored in table:

x_i	P_{i0}	P_{i1}	P_{i2}	P_{i3}				
-1	-6				$\Longrightarrow$	a_0	$=$	-6
1	-2	2			$\Longrightarrow$	a_1	$=$	2
2	-3	-1	-1		$\Longrightarrow$	a_2	$=$	-1
3	2	5	3	1	$\Longrightarrow$	a_3	$=$	1,

Newton polynomial

x_i	-1	1	2	3
y_i	-6	-2	-3	2

$$\begin{aligned} a_0 &= -6 \\ a_1 &= 2 \\ a_2 &= -1 \\ a_3 &= 1 \end{aligned}$$

$$P_3(x) = -6 + 2 \cdot (x+1) + (-1) \cdot (x+1)(x-1) + 1 \cdot (x+1)(x-1)(x-2)\,.$$

We calculate the value of polynomial at point $x^* = 0{,}5$
using Horner scheme:
$z := a_n$
and then for $i = n$-1, n-2, ... , 0 we calculate
$z := z\,(x^* - x_i) + a_i$, etc.

$$P_3(0{,}5) = ((1 \cdot (0{,}5 - 2) - 1) \cdot (0{,}5 - 1) + 2) \cdot (0{,}5 + 1) - 6 = -1{,}125\,.$$

If we add another node $x_4 = 0$ with prescribed value $y_4 = 2$, then it is enough to add one more line to the table

x_i	P_{i0}	P_{i1}	P_{i2}	P_{i3}
-1	-6			
1	-2	2		
2	-3	-1	-1	
3	2	5	3	1

x_4	P_{40}	P_{41}	P_{42}	P_{43}	P_{44}
0	2	0	$2{,}5$	$0{,}5$	$-0{,}5$

$\Longrightarrow \quad a_4 \quad = \quad -0{,}5$

and then $P_4(x) = P_3(x) + (-0{,}5) \cdot (x+1)(x-1)(x-2)(x-3)$
where

$$P_3(x) = -6 + 2 \cdot (x+1) + (-1) \cdot (x+1)(x-1) + 1 \cdot (x+1)(x-1)(x-2)\,.$$

OUTLINE

1. Approximation and interpolation
2. Polynomial interpolation
   a. Lagrange polynomial
   b. Newton polynomial
   c. The error of approximation for interpolating polynomial
   d. Optimal distribution of interpolation nodes
   e. Hermite interpolation
3. Spline interpolation
   a. Linear spline interpolation
   b. Hermite cubic spline
   c. Cubic spline
   d. Cubic natural spline

Notation

Symbol $C\langle a,b\rangle$ denotes a set of all continuous functions on interval $\langle a,b\rangle$.

Symbol $C^k\langle a,b\rangle$ denotes a set of all functions, which are continuous together with its derivatives up to the order k on interval $\langle a,b\rangle$.

For $k = 0$ obviously $C^0\langle a,b\rangle \equiv C\langle a,b\rangle$.

The error of approximation for interpolating polynomial

Let suppose that y_i are not arbitrary,

but they are values of function f in the nodes, $y_i = f(x_i)$.

Then we want to evaluate the error

$$E_n(x^*) := f(x^*) - P_n(x^*)$$

at chosen point x^*.

For $x^* = x_i$ is $E_n(x_i) = 0$.

What is the error out of nodes?

Theorem: Let x^* is and arbitrary point,
$\langle a,b \rangle$ is any interval which contains all interpolation nodes x_i and also the examined point x^* and
let $f \in C^{n+1}\langle a,b \rangle$.

Then for the error $E_n(x^*)$ holds

$$E_n(x^*) = f(x^*) - P_n(x^*) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x^* - x_0)(x^* - x_1)\ldots(x^* - x_n) \ ,$$

where $\xi = \xi(x^*)$ is some point from the interval $\langle a,b \rangle$.

By $\xi = \xi(x^*)$ we want to stress that,
the position of point ξ depends not only on function f and interpolation P_n,
but also on the chosen point x^*.

Notes: (For the simplicity we will consider that

$x_0 < x_1 < \cdots < x_n$.)

1. If M_{n+1} is such a constant that $|f^{(n+1)}(x)| \leq M_{n+1}$ for each $x \in \langle a, b \rangle$, then

$$\left|E_n\left(x^*\right)\right| \leq \frac{M_{n+1}}{(n+1)!} \max_{x \in \langle a, b \rangle} |\omega_{n+1}(x)|,$$

where $\omega_{n+1}(x) = (x - x_0)(x - x_1) \ldots (x - x_n)$.

This estimation is often pessimistic.

2. If function $f(x)$ has derivatives of all orders bounded by the same constant, then for large enough n is the error arbitrary small.

Example: For $f(x) = \sin x$ we can take $M_{n+1} = 1$, therefore

$$|E_n(x)| \leq \frac{(b-a)^{n+1}}{(n+1)!}.$$

It is possible to show that for $n \to \infty$,

$$\frac{(b-a)^{n+1}}{(n+1)!} \to 0$$

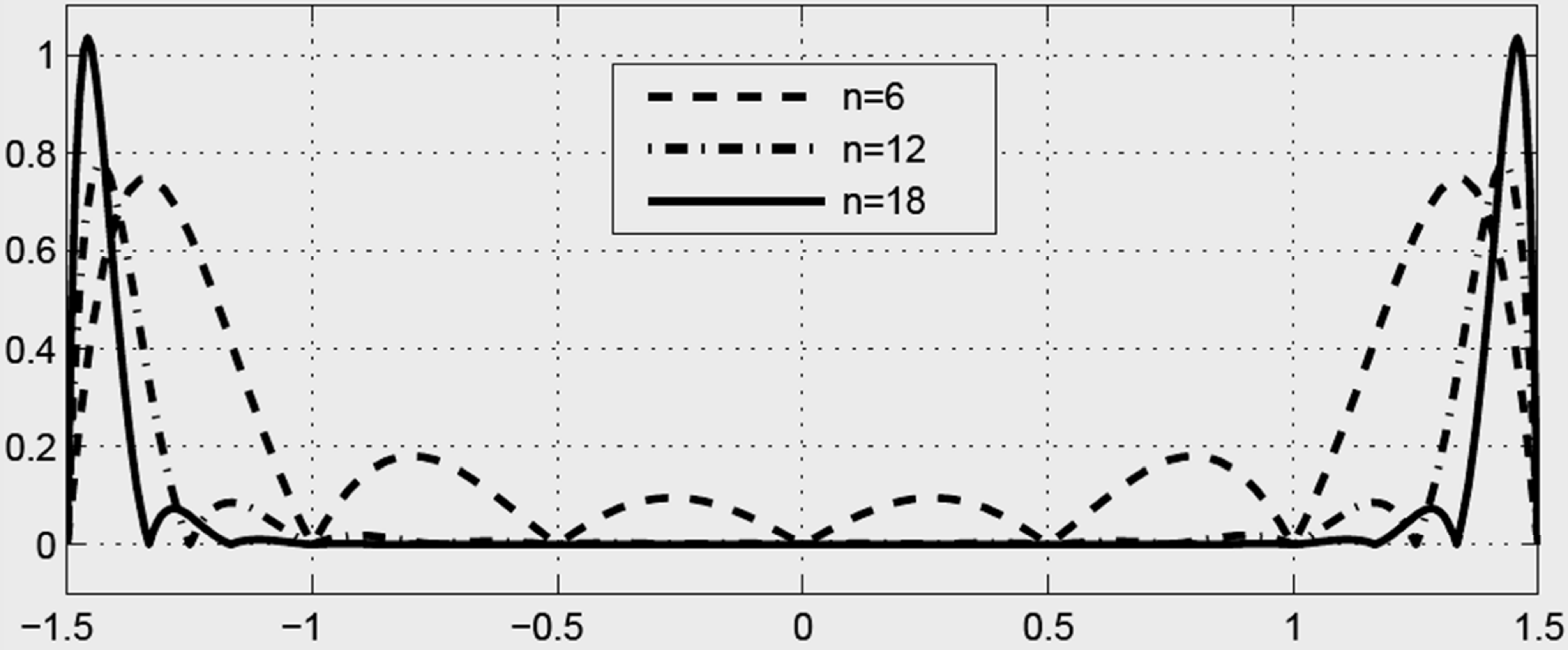
so $P_n(x) \to f(x)$ for each x from any interval $\langle a, b \rangle$.

The error of approximation for interpolating polynomial

3. If interpolation polynomial is used for calculation of values of interpolated function outside interval $\langle x_0, x_n \rangle$, we say that we do **extrapolation**.
   In such a case the error could be large, because the value $|\omega_{n+1}(x)|$ quickly grows, when x retreats form x_0 to the left or from x_n to the right.

4. $\omega_{n+1}(x)$ can achieve large values also inside the interval $\langle x_0, x_n \rangle$, mainly if nodes x_i are deployed equidistantly, i.e. if $x_i = x_0 + ih$ where h is fixed step.

The error of approximation for interpolating polynomial

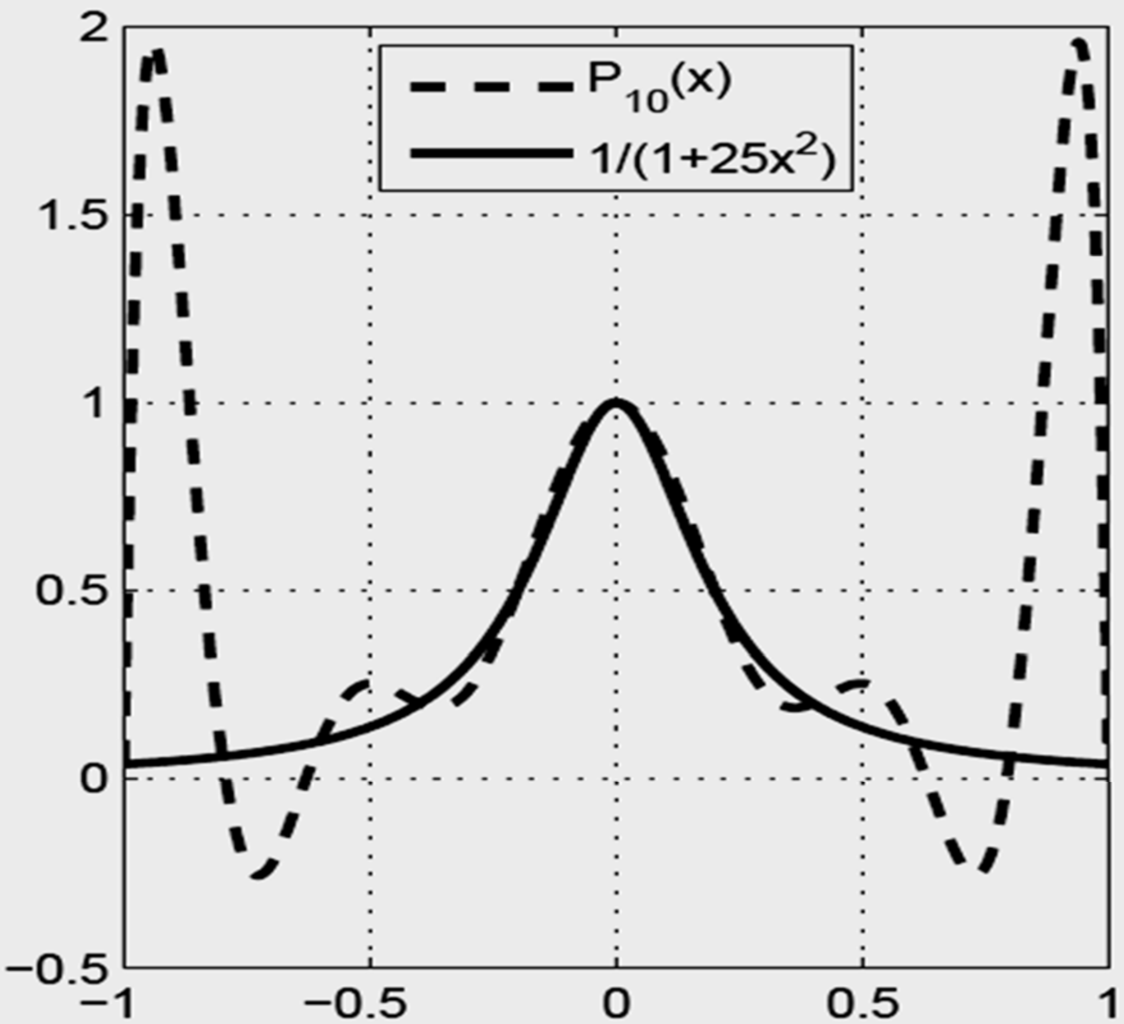


Graph of function $|\omega_{n+1}(x)|$

The error of approximation for interpolating polynomial

Example: Construct the interpolation polynomial for function

$f(x) = \dfrac{1}{1+25x^2}$ using equidistantly positioned nodes on $\langle -1, 1 \rangle$.



This is so called
Runge's phenomenon
and Runge function,
which demonstrates that
the larger number of nodes,
the larger interpolation error.

Therefore
it is advisable
not to use
high degree
interpolation polynomials
with equidistant nodes.

OUTLINE

The Runge's phenomenon can be mitigated
by appropriate distribution of nodes.

Definition: **Normalized polynomial** of degree n has form

$$P_n(x) = x^n + a_1 x^{n-1} + \cdots + a_n$$

Theorem: Among all normalized polynomial of the degree n
just polynomial

$$\tilde{T}_n(z) = \frac{1}{2^{n-1}} \cos(n \arccos z)$$

on the interval $\langle -1, 1 \rangle$
is less deviated from zero.

Polynomials $\tilde{T}_n(z)$ are called **Chebyshev polynomials of the first kind**.

Optimal distribution of interpolation nodes

Chebyshev polynomials could seem to be a trigonometric,
but due to trigonometric identities it is possible to write also this form

$$\tilde{T}_0(z) = 1$$
$$\tilde{T}_1(z) = z$$
$$\tilde{T}_2(z) = 2z^2 - 1$$
$$\tilde{T}_3(z) = 4z^3 - 3z$$
$$\tilde{T}_4(z) = 8z^4 - 8z^2 + 1$$
$$\cdots$$
$$\tilde{T}_{n+1}(z) = 2z\tilde{T}_n(z) - \tilde{T}_{n-1}(z) \qquad n \geq 1.$$

Optimal interpolation nodes are Chebyshev nodes,
i.e. the roots of Chebyshev polynomials of the first kind.

Optimal distribution of interpolation nodes

Let suppose that we are looking for optimal distribution of nodes on interval $\langle a, b\rangle$.

We transform the interval $\langle -1,1\rangle$ into interval $\langle a, b\rangle$

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

Roots of Chebyshev polynomial of degree $n+1$

$$\cos\left[(n+1)\arccos z_i\right] = 0 \quad \Rightarrow \quad (n+1)\arccos z_i = \frac{2i+1}{2}\pi$$

$$z_i = \cos\left(\frac{2i+1}{2n+2}\pi\right) \quad \Leftarrow \quad \arccos z_i = \frac{2i+1}{2(n+1)}\pi$$

then **optimal distribution of interpolation nodes** is

$$x_i = \frac{b-a}{2}\cos\left(\frac{2i+1}{2n+2}\pi\right) + \frac{b+a}{2} \qquad i = 0,1,\ldots,n.$$

OUTLINE

1. Approximation and interpolation
2. Polynomial interpolation
    a. Lagrange polynomial
    b. Newton polynomial
    c. The error of approximation for interpolating polynomial
    d. Optimal distribution of interpolation nodes
    e. Hermite interpolation
3. Spline interpolation
    a. Linear spline interpolation
    b. Hermite cubic spline
    c. Cubic spline
    d. Cubic natural spline

Hermite interpolation

Up till now we deal with interpolation,
in which the interpolating polynomial was given by prescribed values
$P_n(x_i) = y_i$ in nodes x_i.

If we prescribe also derivatives if interpolated function,
we say about **Hermite interpolation.**

Let suppose that
in each node x_i we have $\alpha_i + 1$ numbers $y_i^{(0)}, y_i^{(1)}, \ldots, y_i^{(\alpha_i)}$.

Denote

$$\alpha = n + \sum_{i=0}^{n} \alpha_i \quad .$$

Then **Hermite interpolation polynomial** $P_\alpha(x)$ is
polynomial at most of degree α, which holds interpolation conditions

$$\frac{d^j}{dx^j} P_\alpha(x_i) = y_i^{(j)}, \qquad j = 0, 1, \ldots, \alpha_i, \quad i = 0, 1, \ldots, n.$$

It is possible to proof that there is unique such polynomial.

If

$$y_i^{(j)} = \frac{\mathrm{d}^j}{\mathrm{d}x^j} f(x_i)\,, \qquad j = 0, 1, \ldots, \alpha_i\,, \quad i = 0, 1, \ldots, n\,,$$

we say that $P_\alpha(x)$ is Hermite interpolation polynomial of function $f\,(x)$.

Let $\langle a, b\rangle$ is interval containing all nodes.
If $f \in C^{\alpha+1}\langle a, b\rangle$, then
for the error of Hermite interpolation in point $\bar{x} \in \langle a, b\rangle$ holds

$$f(\bar{x}) - P_\alpha(\bar{x}) = \frac{f^{(\alpha+1)}(\xi)}{(\alpha+1)!}(\bar{x} - x_0)^{\alpha_0+1}(\bar{x} - x_1)^{\alpha_1+1} \ldots (\bar{x} - x_n)^{\alpha_n+1}\,,$$

where $\xi = \xi(\bar{x})$ is some point from interval $\langle a, b\rangle$.

It is not advisable to use the Hermite polynomial of higher degree,
because the error between nodes could be significant.

The formula for calculation of coefficients of Hermite polynomial
is complicated,

we show the calculation on example.

Example Construct Hermite polynomial for data from table

x_i	y_i	y_i'	y_i''
-1	2	-4	12
1	2	4	

So we have $x_0 = -1,\ \alpha_0 = 2,\ y_0^{(0)} = 2,\ y_0^{(1)} = -4,\ y_0^{(2)} = 12,$

$x_1 = \ 1,\ \alpha_1 = 1,\ y_1^{(0)} = 2,\ y_1^{(1)} = \ 4.$

Because we have prescribed 5 conditions,
we will seek for Hermite polynomial of degree $\alpha = 4$.

We will write it in the form of power series around that point,
in which there is the most prescribed conditions,
in our case around the point $x_0 = -1$.

$$P_4(x) = a + b(x+1) + c(x+1)^2 + d(x+1)^3 + e(x+1)^4 .$$

Hermite interpolation

x_i	y_i	y'_i	y''_i
-1	2	-4	12
1	2	4	

$$P_4(x) = a + b(x+1) + c(x+1)^2 + d(x+1)^3 + e(x+1)^4 .$$

Coefficients *a*, *b*, *c* could be easily obtained.

From the condition $P_4(-1) = 2$ we get $a = 2$.

Similarly from $P_4'(-1) = -4$ we get $b = 4$ and because $P_4''(-1) = 2c$, from condition $P_4''(-1) = 12$ we get $c = 6$.

Next

$$P_4(1) = 2 - 4 \cdot 2 + 6 \cdot 2^2 + d \cdot 2^3 + e \cdot 2^4 = 2 \qquad \Longrightarrow \qquad 8d + 16e = -16 ,$$

$$P_4'(1) = -4 + 2 \cdot 6 \cdot 2 + 3 \cdot d \cdot 2^2 + 4 \cdot e \cdot 2^3 = 4 \qquad \Longrightarrow \qquad 12d + 32e = -16 .$$

Solving the system we get $d = -4$, $e = 1$. Therefore

$$P_4(x) = 2 - 4(x+1) + 6(x+1)^2 - 4(x+1)^3 + (x+1)^4 = x^4 - 1 .$$

OUTLINE

1. Approximation and interpolation
2. Polynomial interpolation
   a. Lagrange polynomial
   b. Newton polynomial
   c. The error of approximation for interpolating polynomial
   d. Optimal distribution of interpolation nodes
   e. Hermite interpolation
3. Spline interpolation
   a. Linear spline interpolation
   b. Hermite cubic spline
   c. Cubic spline
   d. Cubic natural spline

Interpolation

We chose interpolation function $\varphi(x)$
from a suitable class of functions.
We restrict ourselves to two
the most common cases:

1. $\varphi(x)$ is a polynomial function;

2. $\varphi(x)$ is a piecewise polynomial,
   i.e. in general different on each subinterval

Spline interpolation

If we want to interpolate function $f\ (x)$
on the relatively long interval $\langle a,b \rangle$,
we have to request fulfilment of interpolation conditions
in a very large number of nodes.

If we use interpolation polynomials then
it has to be high degree
and this, as we already know,
usually leads to large errors between nodes.

This is therefore not right way to do.

The better way is to divide the interval $\langle a,b \rangle$ into
many small subintervals
an on each subinterval
construct an interpolation polynomial of low degree.

Spline interpolation

Suppose that

$$a = x_0 < x_1 < \cdots < x_{i-1} < x_i < x_{i+1} < \cdots < x_{n-1} < x_n = b$$

is **division** of interval $\langle a,b \rangle$.

In each node x_i there is prescribed value y_i of interpolant.

Denote the length of i-th interval $\langle x_{i-1},x_i \rangle$ as h_i
and the length of the longest interval as h, i.e.

$$h_i = x_i - x_{i-1}\,, \qquad i = 1, 2, \ldots, n\,,$$

$$\boxed{h = \max_{1 \le i \le n} h_i\,.}$$

Spline interpolation

We will denote searched piecewise interpolating polynomial
as $S(x)$
and we will call it **interpolating spline**.

The $S(x)$ is polynomial on each interval $\langle x_{i-1}, x_i \rangle$ and reference to the i-th interval is denoted by subscript i, i.e.

$S(x)$ is polynomial $S_i(x)$ on interval $\langle x_{i-1}, x_i \rangle$.

For the expression of polynomial $S_i(x)$
is good to use **local variable**

$$s = x - x_{i-1}\,.$$

We will also use the **first divided difference**

$$\delta_i = \frac{y_i - y_{i-1}}{x_i - x_{i-1}} = \frac{y_i - y_{i-1}}{h_i}\,.$$

OUTLINE

1. Approximation and interpolation
2. Polynomial interpolation
   a. Lagrange polynomial
   b. Newton polynomial
   c. The error of approximation for interpolating polynomial
   d. Optimal distribution of interpolation nodes
   e. Hermite interpolation
3. Spline interpolation
   a. Linear spline interpolation
   b. Hermite cubic spline
   c. Cubic spline
   d. Cubic natural spline

Linear spline is the easiest spline:
we connect each two neighboring points $[x_{i-1}, y_{i-1}]$ and $[x_i, y_i]$ by a line segment.

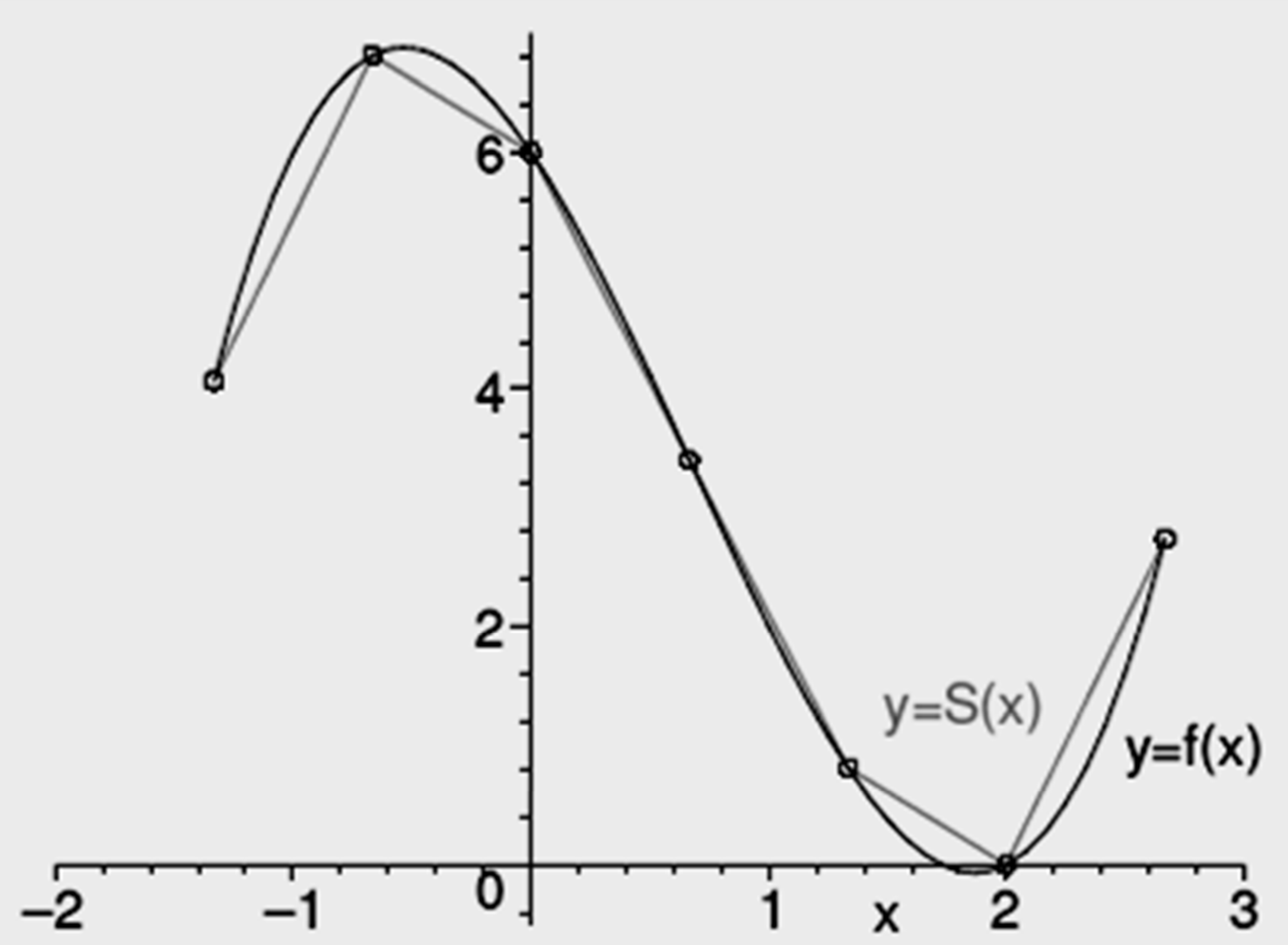
Then

$$S_i(x) = y_{i-1} + \frac{y_i - y_{i-1}}{x_i - x_{i-1}}(x - x_{i-1}) = y_{i-1} + s\delta_i$$

is linear interpolating polynomial passing through points $[x_{i-1}, y_{i-1}]$ and $[x_i, y_i]$.

Linear spline $S(x)$ is continuous function,
the derivative $S`(x)$ is however
in general discontinuous at interior nodes.

Linear spline interpolation



Linear spline interpolation

If $y_i = f(x_i),\ i = 0, 1, \ldots, n,$ and $f \in C^2\langle a, b\rangle$, then for the error of approximation it holds

$$|f(x) - S(x)| \leq Ch^2 ,$$

where $x \in \langle a, b\rangle$ is arbitrary and C is constant independent on h.

For a sufficiently large number of nodes it is possible to make the error arbitrary small.

Example: Drawing a graph on screen with resolution 1024 x 768 points.

Linear spline interpolation

More accurate interpolant could be constructed in such a way,
that we approximate function $f(x)$
on intervals $\langle x_0, x_k\rangle$, $\langle x_k, x_{2k}\rangle$, ...
using interpolating polynomials of degree
at most k, where $k > 1$.

The error of interpolation would be proportional to h^{k+1},
but derivatives in nodes $x_k, x_{2k}, \ldots$ would remain discontinuous.

The large k has no sense, because,
we would have large errors
between nodes and
we would have the same problem
as in the beginning.

OUTLINE

1. Approximation and interpolation
2. Polynomial interpolation
   a. Lagrange polynomial
   b. Newton polynomial
   c. The error of approximation for interpolating polynomial
   d. Optimal distribution of interpolation nodes
   e. Hermite interpolation
3. Spline interpolation
   a. Linear spline interpolation
   b. **Hermite cubic spline**
   c. Cubic spline
   d. Cubic natural spline

Hermite cubic spline

is a function $S(x)$, which

1. it is continuous on interval $\langle a, b\rangle$ together with its first derivative, i.e. $S \in C^1\langle a, b\rangle$,
2. it holds interpolating conditions

$$S(x_i) = y_i\,, \qquad S'(x_i) = d_i\,, \qquad i = 0, 1, \ldots, n\,,$$

   where y_i, d_i are given functional values and derivatives, respectively,
3. it is polynomial at most third degree on each interval $\langle x_{i-1}, x_i\rangle$, $i = 1, 2, \ldots, n$.

$S_i(x)$ is therefore cubic Hermite polynomial
uniquely defined by conditions

$$
\begin{aligned}
S_i(x_{i-1}) &= y_{i-1}\,, & S_i'(x_{i-1}) &= d_{i-1}\,,\\
S_i(x_i) &= y_i\,, & S_i'(x_i) &= d_i\,.
\end{aligned}
$$

It is easy to find, that the conditions are fulfilled for

$$
S_i(x) = y_{i-1} + s d_{i-1} + s^2\,\frac{3\delta_i - 2d_{i-1} - d_i}{h_i} + s^3\,\frac{d_{i-1} - 2\delta_i + d_i}{h_i^2}\,.
$$

Function $S(x)$ is continuous together with its first derivative,
the second derivative is in general discontinuous.

Hermite cubic spline

If $y_i = f(x_i)$, $d_i = f'(x_i)$, $i = 0, 1, \ldots, n$, and $f \in C^4\langle a, b\rangle$, then for the error of interpolation it holds

$$|f(x) - S(x)| \le Ch^4\,,$$

where $x \in \langle a, b\rangle$ is arbitrary and C is constant independent on h.

If the derivatives d_i are not provided, we have to calculate them using appropriate additional conditions.

Shape preserving Hermite cubic spline
is one possibility.

The derivatives d_i are chosen in such a way that $S(x)$ will have the same convexity as linear spline passing through points $[x_i, y_i]$.

In detail, if $L(x)$ is linear spline then we require:

1. if $L(x)$ has a local extrema at interior node, then the $S(x)$ has also local extrema;
2. if $L(x)$ is monotonous between two neighboring nodes, then also $S(x)$ is same way monotonous.

One of good implementation
could be find in MATLAB as function **`pchip`**.

The calculation of tangents d_i is made as follows:

1. <u>Interior nodes</u>

   If tangents δ_i and δ_{i+1} have opposite signs,
   or if some of them equals to zero, i.e. if

   $\delta_i \delta_{i+1} \leq 0\,,$ we set $d_i = 0\,.$

   Otherwise we estimate d_i as generalized
   harmonic average of tangents δ_i and δ_{i+1} as

   $$\frac{w_1 + w_2}{d_i} = \frac{w_1}{\delta_i} + \frac{w_2}{\delta_{i+1}}\,,$$

   where $w_1 = h_i + 2h_{i+1}\,,\ w_2 = 2h_i + h_{i+1}\,.$

One of good implementation
could be find in MATLAB as function **pchip**.

The calculation of tangents d_i is made as follows:

2. <u>Endpoints</u> x_0 and x_n.

   The easiest way is to set up $d_0 = \delta_1$, $d_n = \delta_n$.

   There is better approximation in the **pchip** algorithm based on quadratic interpolation (see https://www.mathworks.com/moler/interp.pdf)

OUTLINE

1. Approximation and interpolation
2. Polynomial interpolation
   a. Lagrange polynomial
   b. Newton polynomial
   c. The error of approximation for interpolating polynomial
   d. Optimal distribution of interpolation nodes
   e. Hermite interpolation
3. Spline interpolation
   a. Linear spline interpolation
   b. Hermite cubic spline
   c. Cubic spline
   d. Cubic natural spline

In the **cubic spline** we can determine
the tangents d_i at interior nodes in such a way,
that we require $S(x)$ to have continuous also
the second derivative

$$S_i''(x_i) = S_{i+1}''(x_i)\,, \qquad i = 1, 2, \ldots, n-1\,.$$

If we differentiate

$$S_i(x) = y_{i-1} + s d_{i-1} + s^2\,\frac{3\delta_i - 2d_{i-1} - d_i}{h_i} + s^3\,\frac{d_{i-1} - 2\delta_i + d_i}{h_i^2}\,.$$

we get

$$S_i''(x) = \frac{(6h_i - 12s)\delta_i + (6s - 4h_i)d_{i-1} + (6s - 2h_i)d_i}{h_i^2}\,.$$

Cubic spline

$$S_i''(x) = \frac{(6h_i - 12s)\delta_i + (6s - 4h_i)d_{i-1} + (6s - 2h_i)d_i}{h_i^2}\,.$$

For $x = x_i$ we have $s = h_i$, so

$$S_i''(x_i) = \frac{-6\delta_i + 2d_{i-1} + 4d_i}{h_i}\,.$$

For $x = x_{i-1}$ we have $s = 0$ and

$$S_i''(x_{i-1}) = \frac{6\delta_i - 4d_{i-1} - 2d_i}{h_i}\,.$$

If we advance the subscript i by 1 in the last formula we get

$$S_{i+1}''(x_i) = \frac{6\delta_{i+1} - 4d_i - 2d_{i+1}}{h_{i+1}}\,.$$

Cubic spline

Inserting into equation

$$S_i''(x_i) = S_{i+1}''(x_i)\,, \qquad i = 1, 2, \ldots, n-1\,.$$

we get

$$h_{i+1}d_{i-1} + 2(h_{i+1} + h_i)d_i + h_i d_{i+1} = 3(h_{i+1}\delta_i + h_i\delta_{i+1})\,, \qquad i = 1, 2, \ldots, n-1\,.$$

If we have the **boundary conditions** like

$$S'(a) = d_a\,, \quad S'(b) = d_b\,,$$

then we insert into the first equation $d_0 := d_a$ and term $h_2 d_a$ will go to the right-hand side

and into the last equation we insert $d_n := d_b$ and term $h_{n-1}d_b$ will go to the right-hand side.

Solving the system we obtain the rest tangents $d_i, i = 1, 2, \ldots, n-1.$

The coefficient matrix is tridiagonal, diagonally dominant, so we can solve it by modified GEM for tridiagonal matrices.

If

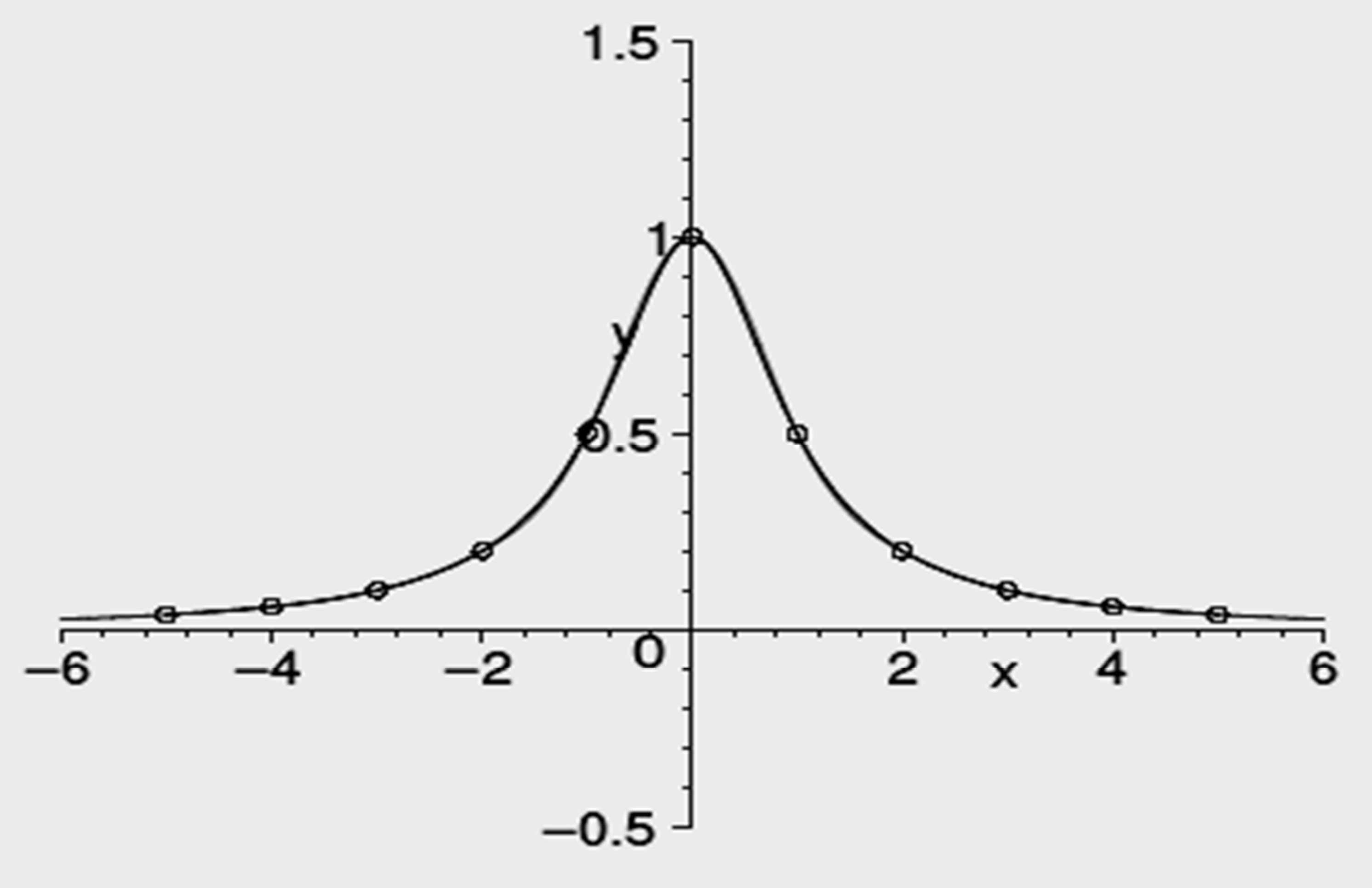
$$y_i = f(x_i),\ i = 0, 1, \ldots, n,\ d_0 = f'(x_0),\ d_n = f'(x_n),$$

and if $f \in C^4\langle a, b\rangle$,

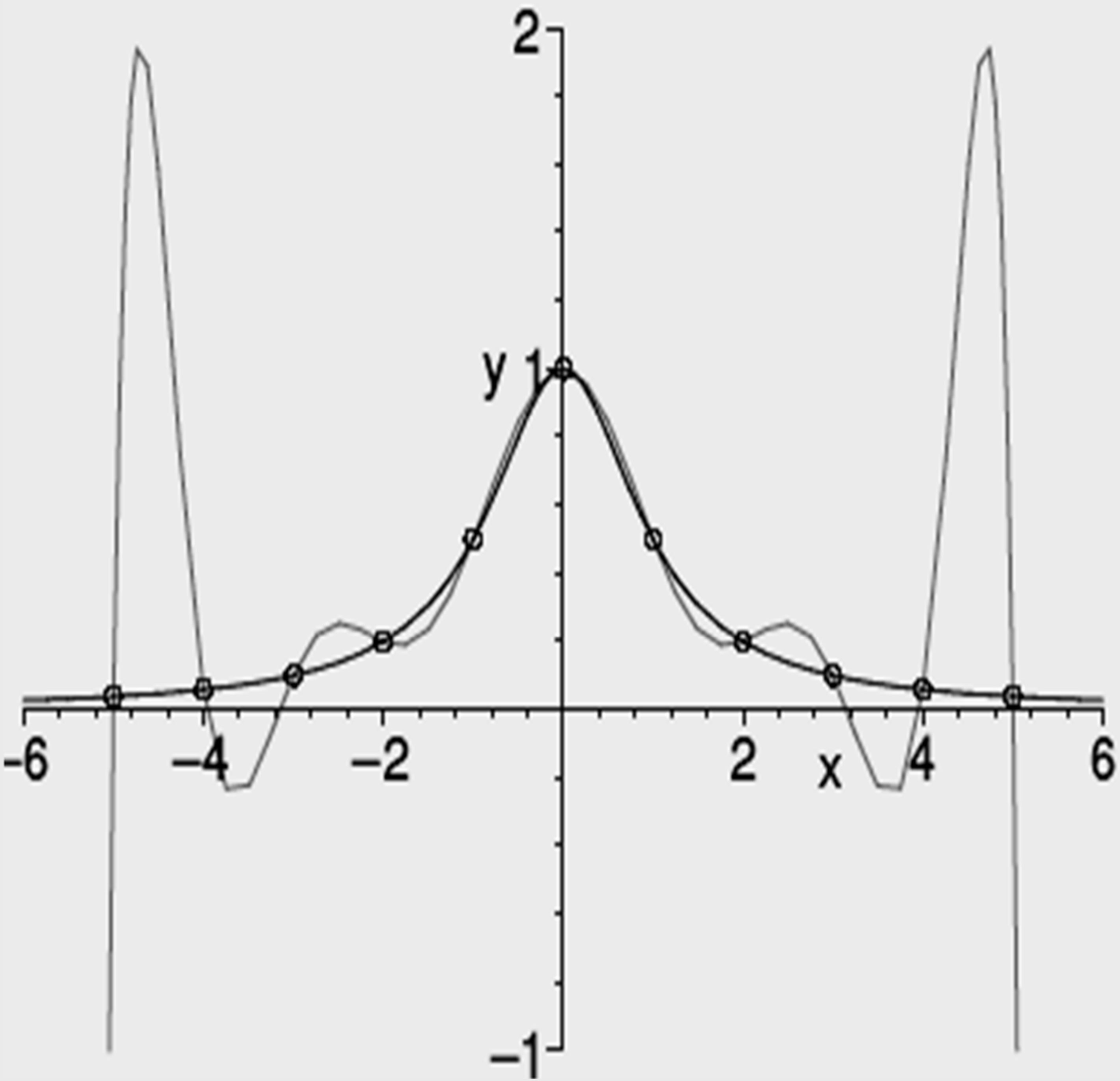
then for the error of interpolation it holds

$$|f(x) - S(x)| \leq Ch^4$$

Cubic spline



The error for interpolating polynomial



Cubic spline has interesting **extremal property.**

Denote

$$V = \{v \in C^2\langle a, b\rangle \,|\, v(x_i) = y_i, i = 0, 1, \ldots, n, v'(x_0) = d_0, v'(x_n) = d_n\}$$

the set of all functions,
which have continuous second derivative on interval $\langle a, b\rangle$,
pass through given points $[x_i, y_i]$, $i = 0, 1, \ldots, n$,
and in endpoints $a = x_0$ and $x_n = b$
have derivatives values d_0 and d_n.

Then $\int_a^b [v''(x)]^2 \,\mathrm{d}x$ for cubic spline $S(x)$ achieves minimal value on the set of all function V, i.e. it hold

$$\int_a^b [S''(x)]^2 \,\mathrm{d}x = \min_{v \in V} \int_a^b [v''(x)]^2 \,\mathrm{d}x \,.$$

Cubic spline

This property has interesting interpretation in mechanics.

It is known that
elastic energy of homogeneous isotropic rod,
which central line is described as
$$y = v(x),\ x \in \langle a, b \rangle,$$
has approximately value
$$E(v) = c \int_a^b [v''(x)]^2 \,\mathrm{d}x,$$
where c is a constant.

It also holds, that rod,
which is constrained on passing through
fixed interpolating points $[x_i, y_i]$ in such a way,
that it is only under normal stress to the rod,
take place with minimal energy.

Extremal property therefore claims,
that cubic spline approximates central line of such a rod..

If we do not know the tangents d_a and d_b
in endpoints of interval $\langle a, b \rangle$,
then we can use other boundary conditions.

Construction of cubic spline using the second derivatives.

We can easily check that cubic polynomial

$$S_i(x) = y_{i-1} + s\,\frac{6\delta_i - 2h_i M_{i-1} - h_i M_i}{6} + s^2\,\frac{M_{i-1}}{2} + s^3\,\frac{M_i - M_{i-1}}{6h_i} \qquad (1)$$

satisfies conditions

$$\begin{aligned} S_i(x_{i-1}) &= y_{i-1}\,, \quad & S_i''(x_{i-1}) &= M_{i-1}\,, \\ S_i(x_i) &= y_i\,, & S_i''(x_i) &= M_i\,. \end{aligned}$$

Function $S(x)$ defined on each interval $\langle x_{i-1}, x_i\rangle$ by eq. (1) therefore satisfies conditions

$$S(x_i) = y_i,\; S''(x_i) = M_i,\; i = 0, 1, \ldots, n.$$

$S(x)$ is continuous on interval $\langle a, b\rangle$ and it has there continuous also the second derivative.

Cubic spline

In order to obtain cubic spline,
function $S(x)$ has to have continuous also first derivative
on interval $\langle a, b \rangle$.
We require, that at interior points it holds

$$S_i'(x_i) = S_{i+1}'(x_i)\,, \qquad i = 1, 2, \ldots, n-1\,.$$

If we express that condition using

$$S_i(x) = y_{i-1} + s\,\frac{6\delta_i - 2h_i M_{i-1} - h_i M_i}{6} + s^2\,\frac{M_{i-1}}{2} + s^3\,\frac{M_i - M_{i-1}}{6h_i}$$

then we obtain

$$h_i M_{i-1} + 2(h_i + h_{i+1}) M_i + h_{i+1} M_{i+1} = 6(\delta_{i+1} - \delta_i)\,, \qquad i = 1, 2, \ldots, n-1\,.$$

If we chose boundary conditions as

$$S''(a) = M_a\,, \qquad S''(b) = M_b\,,$$

then solving the system we obtain

$$M_i,\ i = 1, 2, \ldots, n-1.$$

OUTLINE

1. Approximation and interpolation
2. Polynomial interpolation
   a. Lagrange polynomial
   b. Newton polynomial
   c. The error of approximation for interpolating polynomial
   d. Optimal distribution of interpolation nodes
   e. Hermite interpolation
3. Spline interpolation
   a. Linear spline interpolation
   b. Hermite cubic spline
   c. Cubic spline
   d. Cubic natural spline

Cubic natural spline

Cubic spline with a property

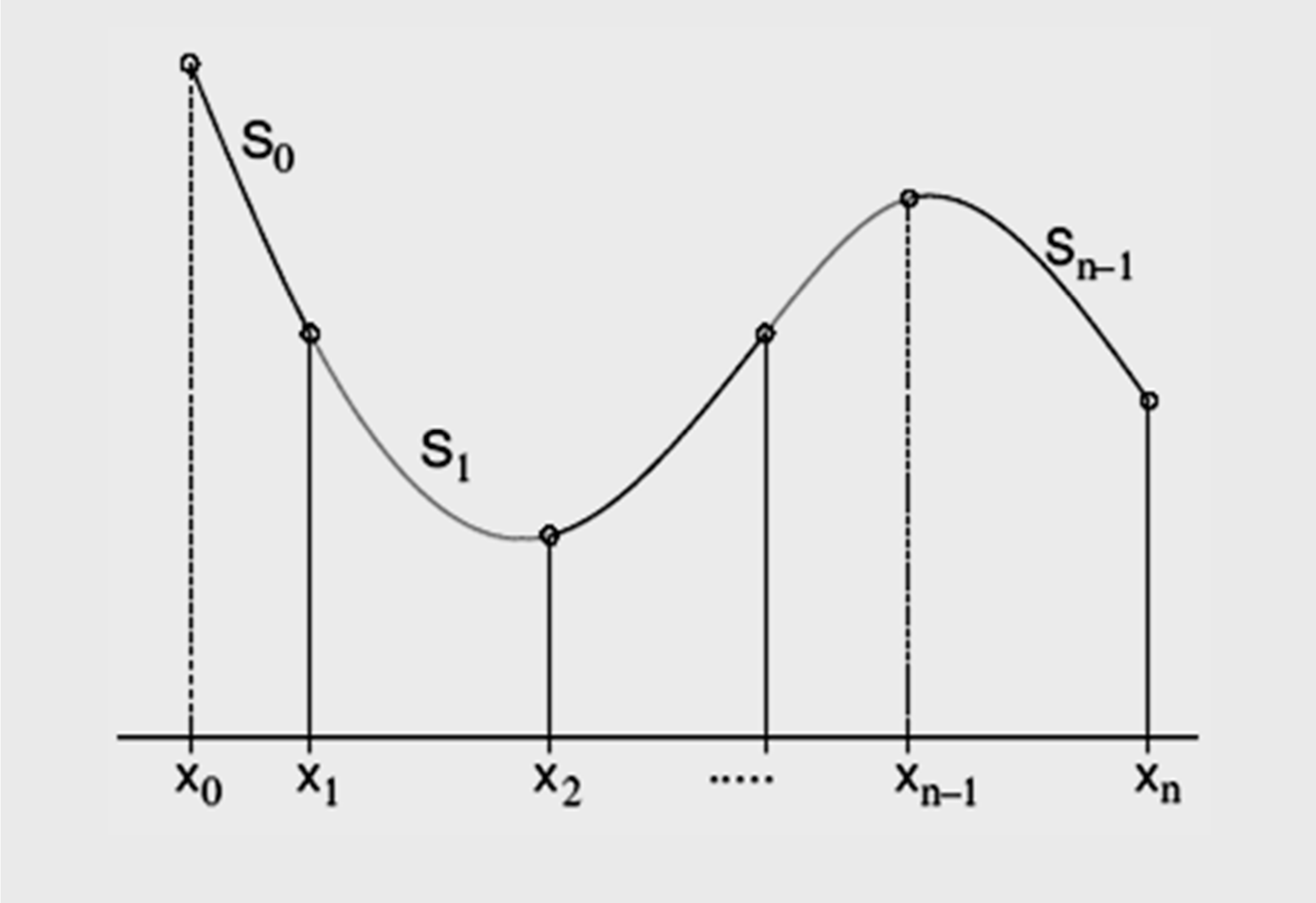
$$S''(a) = S''(b) = 0$$

is called **cubic natural spline**.

It is known that natural spline approximates bending of simply supported (homogeneous isotropic) beam so the beam passes through points $[x_i, y_i]$

M_i have the meaning of bending moments in $[x_i, y_i]$.

Cubic natural spline



Cubic spline

If we do not know the tangents d_a and d_b
in endpoints of interval $\langle a, b \rangle$,
we can use other boundary conditions.

One of them is called **not a knot**.
The idea is simple:
we require the spline to be
simple polynomial od the third degree
on the first two intervals,
i.e. for $x_0 \leq x \leq x_2$,
and on the last two intervals,
i.e. for $x_{n-2} \leq x \leq x_n$.

In nodes x_1 and x_{n-1} there is not
connection of two polynomials,
that is the points x_1 and x_{n-1} are not „knots“.

Cubic spline

Polynomials $S_1(x)$ and $S_2(x)$ have
common value y_1, common first and second derivative
in point x_1.

Therefore to be both polynomial the same it is enough to require
to have the continuous third derivative
in point x_1.

Similar though holds also in point x_{n-1}.

This way we obtain boundary conditions

$$S_1'''(x_1) = S_2'''(x_1)\,, \qquad S_{n-1}'''(x_{n-1}) = S_n'''(x_{n-1})\,.$$

Summary

Cubic spline $S(x)$ is function that

1. it is continuous together with its first and second derivatives on interval $\langle a, b\rangle$,
   i.e. $S \in C^2\langle a, b\rangle$,
2. it hold interpolating conditions $S(x_i) = y_i,\ i = 0, 1, \ldots, n,$
   where y_i are given functional values,
3. it is polynomial at most degree of three
   on each interval $\langle x_{i-1}, x_i\rangle$,
4. it holds boundary conditions

   a) $S'(a) = d_a\,, \quad S'(b) = d_b\,,$

   b) $S''(a) = S''(b) = 0\,,$

   c) $S_1'''(x_1) = S_2'''(x_1)\,, \qquad S_{n-1}'''(x_{n-1}) = S_n'''(x_{n-1})\,.$