Approximation of Functions and Data

Laura Grigori

EPFL and PSI

slides based on lecture notes/slides from L. Dede/S. Deparis

October 1/8 2025

EPFL

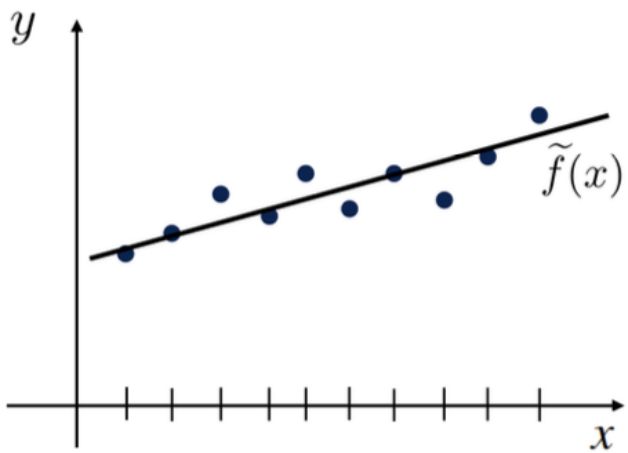
PAUL SCHERRER INSTITUT PSI

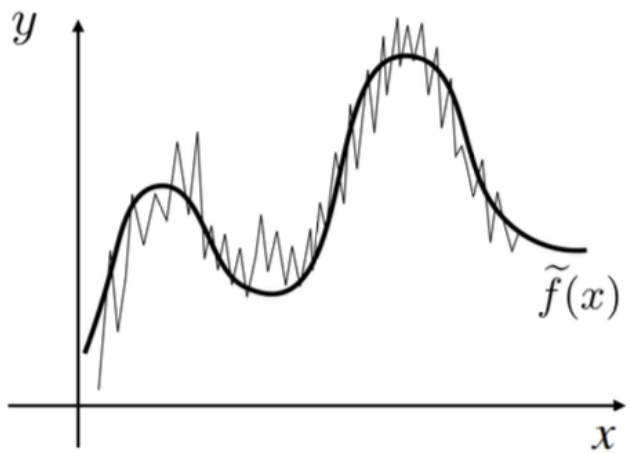
Plan

Plan

Motivation

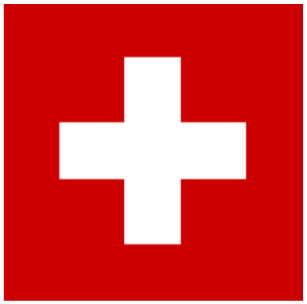
Goal: Approximate a set of data couples or a function $f(x)$ by another function $\tilde{f}(x)$ easier to handle.





Examples

Example 1: The result of census of the population of Switzerland between 1900 and 2010 (in thousands):



year	1900	1910	1920	1930	1941	1950
population	3315	3753	3880	4066	4266	4715

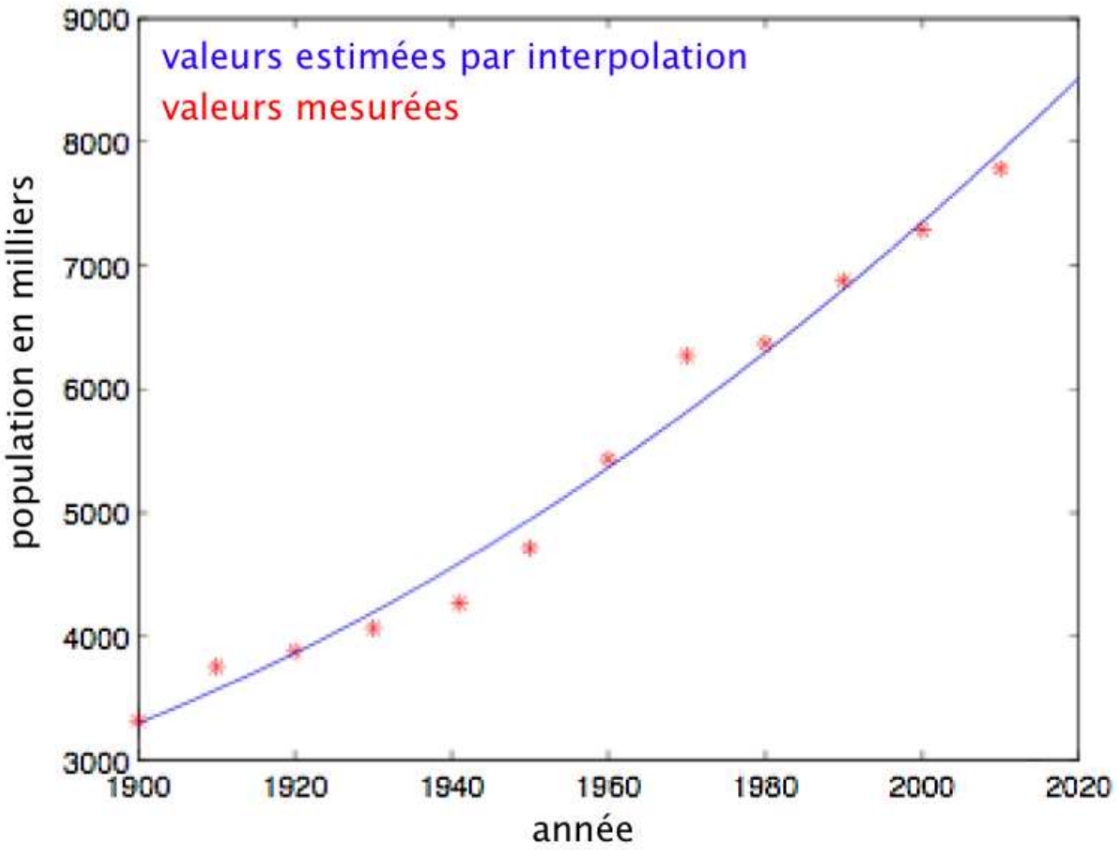
zear	1960	1970	1980	1990	2000	2010
population	5429	6270	6366	6874	7288	7783

- Is it possible to estimate the number of inhabitants of Switzerland during the year when there has not been census, for example in 1945 and 1975?
- Is it possible to predict the number of inhabitants of Switzerland in 2020?

Example 1

The polynomial of degree two (parabola) which approximates the data by the method of least squares is:

$$p(x) = 0.15x^2 - 549.9x + 501600$$



Approximation of functions by Taylor's polynomials

- Function $f \in C^n(I_{x_0})$ can be approximated in a neighborhood I_{x_0} of a point $x_0 \in \mathbb{R}$ by the Taylor polynomial (expansion) of order n
- The Taylor expansion of $f(x)$ around x_0 is given by:

$$\tilde{f}(x) = f(x_0) + \sum_{i=1}^{n} \frac{1}{i!} f^{(i)}(x_0)(x - x_0)^i.$$

Inconvenients:

- the evaluation of n derivatives of $f(x)$ is required
- Taylor's expansion is accurate only in a neighborhood of x_0

Approximation of functions by Taylor's polynomials

- Function $f \in C^n(I_{x_0})$ can be approximated in a neighborhood I_{x_0} of a point $x_0 \in \mathbb{R}$ by the Taylor polynomial (expansion) of order n
- The Taylor expansion of $f(x)$ around x_0 is given by:

$$\tilde{f}(x) = f(x_0) + \sum_{i=1}^{n} \frac{1}{i!} f^{(i)}(x_0)(x - x_0)^i.$$

Inconvenients:

- the evaluation of n derivatives of $f(x)$ is required
- Taylor's expansion is accurate only in a neighborhood of x_0

Approximation of functions by Taylor's polynomials

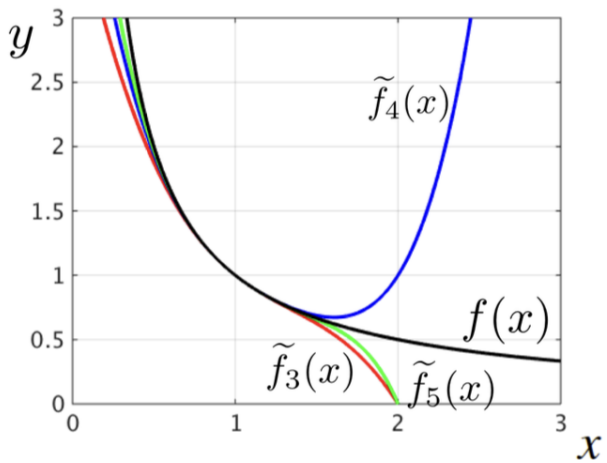
- Consider the Taylor expansion of $f(x) = \frac{1}{x}$ of order n, denoted as $\tilde{f}_n(x)$, around $x_0 = 1$. The expansion is given by:

$$\tilde{f}_n(x) = f(1) + \sum_{i=1}^{n} \frac{1}{i!} f^{(i)}(1)(x-1)^i.$$

- Since $f^{(i)}(x) = (-1)^i\, i!\, x^{-(i+1)}$ for $i = 0, 1, \ldots, n$, we have

$$\tilde{f}(x) = \tilde{f}_n(x) = 1 + \sum_{i=1}^{n} (-1)^i (x-1)^i.$$

- Approximation innacurate "far" from $x_0 = 1$



Plan

Interpolation

Definition

Consider $n+1$ data pairs $\{(x_i, y_i)\}_{i=0}^{n}$ with $\{x_i\}_{i=0}^{n}$ being $n+1$ distinct nodes, i.e., such that $x_i \neq x_j$ for all $i \neq j$, where $i, j = 0, \ldots, n$.

Interpolating data pairs $\{(x_i, y_i)\}_{i=0}^{n}$ means determining the approximate function $\tilde{f}(x)$ such that $\tilde{f}(x_i) = y_i$ for all $i = 0, \ldots, n$

If $f(x)$ is known, we set $y_i = f(x_i)$ for all $i = 0, \ldots, n$, and $\tilde{f}(x_i) = f(x_i)$ for all $i = 0, \ldots, n$.

The function $\tilde{f}(x)$ is called the interpolant of the data at the nodes.

Interpolation

Definition

Consider $n+1$ data pairs $\{(x_i, y_i)\}_{i=0}^{n}$ with $\{x_i\}_{i=0}^{n}$ being $n+1$ distinct nodes, i.e., such that $x_i \neq x_j$ for all $i \neq j$, where $i, j = 0, \ldots, n$.

Interpolating data pairs $\{(x_i, y_i)\}_{i=0}^{n}$ means determining the approximate function $\tilde{f}(x)$ such that $\tilde{f}(x_i) = y_i$ for all $i = 0, \ldots, n$

If $f(x)$ is known, we set $y_i = f(x_i)$ for all $i = 0, \ldots, n$, and $\tilde{f}(x_i) = f(x_i)$ for all $i = 0, \ldots, n$.

The function $\tilde{f}(x)$ is called the interpolant of the data at the nodes.

Interpolation types

- **Polynomial interpolation**, for which

$$\tilde{f}(x) = a_0 + a_1 x + \cdots + a_n x^n$$

  for some $n+1$ coefficients $a_0, a_1, \ldots, a_n$;
- **Rational interpolation**, for which

$$\tilde{f}(x) = \frac{a_0 + a_1 x + \cdots + a_k x^k}{a_{k+1} + a_{k+2} x + \cdots + a_{k+n+1} x^n}$$

  for some coefficients $a_0, a_1, \ldots, a_{k+n+1}$ with $k, n \geq 0$;
- **Trigonometric interpolation**, for which

$$\tilde{f}(x) = \sum_{j=-M}^{M} a_j e^{ijx}, \quad e^{ijx} = \cos(jx) + i \sin(jx),$$

  where i is the imaginary unit ($i^2 = -1$), for some M and complex coefficients a_j;
- **Piecewise polynomial interpolation**;
- **Splines**
- ...

Plan

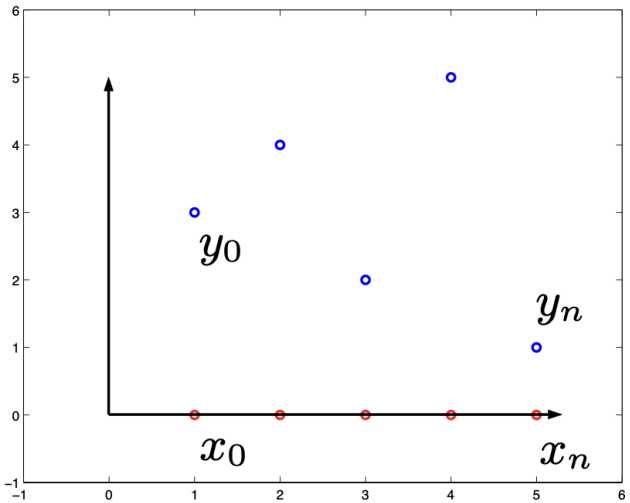
Polynomial interpolation

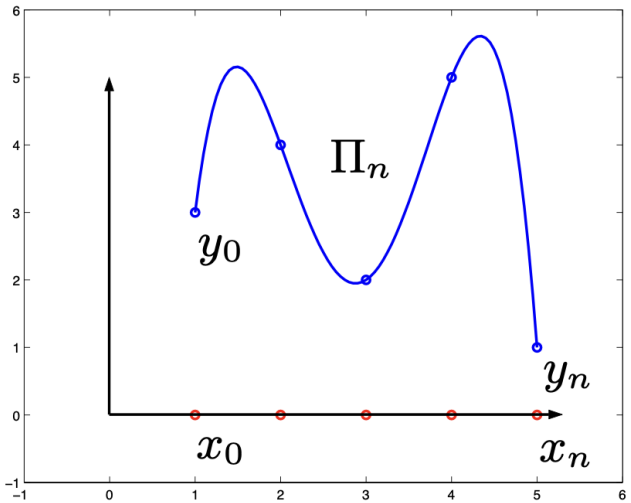
Proposition 3.1 Let $n \geq 0$ be an integer. Given distinct $n+1$ points $x_0, x_1, \ldots, x_n$ and associated $n+1$ values $y_0, y_1, \ldots, y_n$, there exists a unique polynomial $\Pi_n(x)$ of degree less than or equal to n, such that

$$\Pi_n(x_j) = y_j \quad \text{for} \quad 0 \leq j \leq n$$

- We call it the interpolating polynomial of the values y_j at nodes x_j, for $j = 0, \ldots, n$.

$(n = 4)$



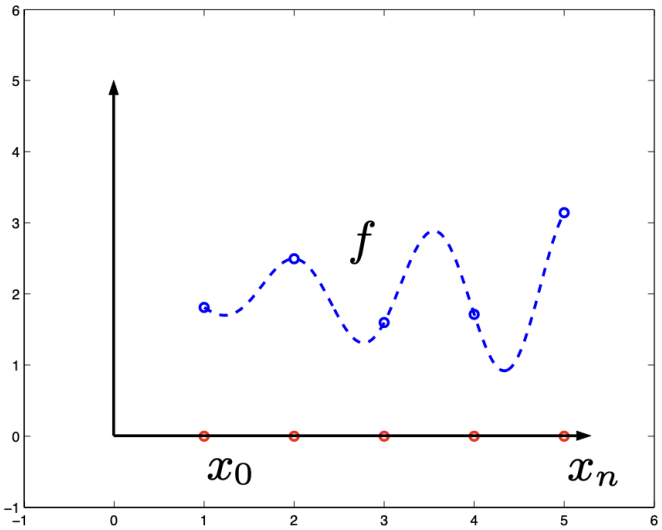


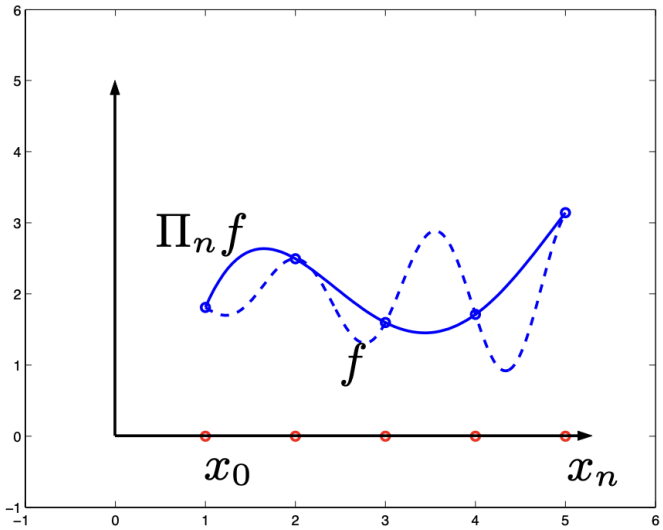
Interpolation of a function

Let $f \in C^0(I)$ and $x_0, \ldots, x_n \in I$. Taking $y_j = f(x_j)$ for $0 \leq j \leq n$, the interpolating polynomial $\Pi_n(x)$ is denoted by $\Pi_n f(x)$ and is called the interpolating polynomial of f at the points $x_0, \ldots, x_n$. We have

$$\Pi_n f(x_j) = y_j \quad \text{for} \quad 0 \leq j \leq n.$$

$(n = 4)$





Lagrange interpolating polynomials

Definition 3.2 For a set of distinct nodes $\{x_i\}_{i=0}^n$, the Lagrange characteristic function associated to the node x_k is a polynomial ϕ_k, for $k = 0, \ldots, n$, of degree n such that

$$\phi_k(x_i) = \delta_{ki}, \quad k, i = 0, \ldots, n,$$

where $\delta_{ki} = 1$ if $i = k$ and $\delta_{ki} = 0$ if $i \neq k$. Explicitly, we have

$$\phi_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i}.$$

- The set $\{\phi_k(x)\}_{k=0}^n$ is the basis of Lagrange characteristic polynomials, a basis of $\mathbb{P}_n$.

Lagrange interpolating polynomials

Definition 3.2 For a set of distinct nodes $\{x_i\}_{i=0}^n$, the Lagrange characteristic function associated to the node x_k is a polynomial ϕ_k, for $k = 0, \ldots, n$, of degree n such that

$$\phi_k(x_i) = \delta_{ki}, \quad k, i = 0, \ldots, n,$$

where $\delta_{ki} = 1$ if $i = k$ and $\delta_{ki} = 0$ if $i \neq k$. Explicitly, we have

$$\phi_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i}.$$

- The set $\{\phi_k(x)\}_{k=0}^n$ is the basis of Lagrange characteristic polynomials, a basis of $\mathbb{P}_n$.

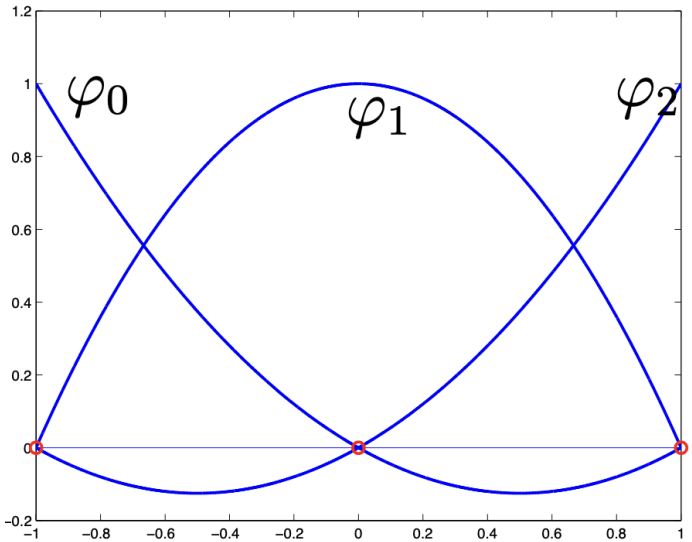
Lagrange basis: example

Example 2. For $n = 2$, $x_0 = -1$, $x_1 = 0$, $x_2 = 1$, the polynomials of the Lagrange basis are:

$$\phi_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} = \frac{1}{2}x(x - 1),$$

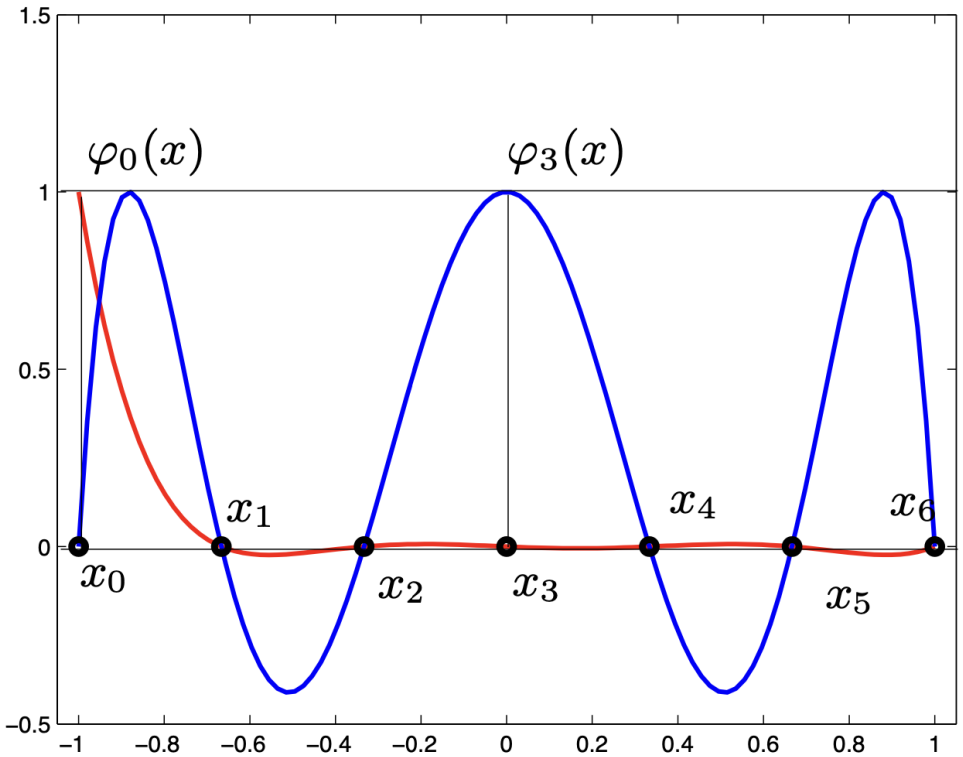
$$\phi_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} = -(x + 1)(x - 1),$$

$$\phi_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} = \frac{1}{2}x(x + 1).$$



Lagrange basis: example

The following figure shows two polynomials of the Lagrange basis of degree $n = 6$ with respect to the interpolation points $x_0 = -1$, $x_1 = -\frac{2}{3}, \ldots, x_5 = \frac{2}{3}$, and $x_6 = 1$.



Lagrange interpolating polynomials

Definition 3.3 The interpolating polynomial Π_n of values y_i at nodes x_i, $i = 0, \ldots, n$, is written as

$$\Pi_n(x) = \sum_{k=0}^{n} y_k \phi_k(x), \tag{1}$$

because it satisfies $\Pi_n(x_j) = \sum_{k=0}^{n} y_k \phi_k(x_j) = y_j$.

If the function $f(x)$ is given and is continuous, its Lagrange interpolating polynomial at the nodes $\{x_i\}_{i=0}^{n}$ is:

$$\Pi_n f(x) = \sum_{k=0}^{n} f(x_k) \phi_k(x).$$

Lagrange interpolating polynomials

Definition 3.3 The interpolating polynomial Π_n of values y_i at nodes x_i, $i = 0, \ldots, n$, is written as

$$\Pi_n(x) = \sum_{k=0}^{n} y_k \phi_k(x), \tag{1}$$

because it satisfies $\Pi_n(x_j) = \sum_{k=0}^{n} y_k \phi_k(x_j) = y_j$.
If the function $f(x)$ is given and is continuous, its Lagrange interpolating polynomial at the nodes $\{x_i\}_{i=0}^{n}$ is:

$$\Pi_n f(x) = \sum_{k=0}^{n} f(x_k) \phi_k(x).$$

Numerical interpolation using Matlab

- In Matlab, we can compute the interpolation polynomials using the command `polyfit` and `polyval`. See more how to use these commands.
- `p = polyfit(x,y,n)` calculates the coefficients of the polynomial of degree n that interpolates the values y in nodes x.

Examples in Matlab

Example A. We want to interpolate the values $y = [3.38, 3.86, 3.85, 3.59, 3.49]$ at the nodes $x = [0, 0.25, 0.5, 0.75, 1]$ by a polynomial of degree 4. Use the following commands in MATLAB:

- `x = [0:0.25:1]; % vector of the interpolation points`
- `y = [3.38 3.86 3.85 3.59 3.49]; % vector of the values`
- `p1 = polyfit(x, y, 4)`

The output will be:

$$p_1 = [1.8133, -0.1600, -4.5933, 3.0500, 3.3800]$$

p_1 is a vector of coefficients of the interpolating polynomial:

$$\Pi_4(x) = 1.8133x^4 - 0.16x^3 - 4.5933x^2 + 3.05x + 3.38.$$

To calculate the polynomial of degree n that interpolates a function f at given $n+1$ nodes, first construct the vector y of values f at the nodes x.

Examples in Matlab

Example B. Consider the following case:

- `f = @(x) cos(x);`
- `x = [0:0.25:1];`
- `y = f(x);`
- `p = polyfit(x, y, 4)`

The output will be:

$$p = [0.0362, 0.0063, -0.5025, 0.0003, 1.0000]$$

Remark 1. If the dimensions of x and y are $m+1 > n+1$ (where n is the degree of the interpolation polynomial), the command `polyfit(x, y, n)` returns the interpolation polynomial of degree n using the least squares method (see sec. 3.3 in the book). In the case when $m+1 = n+1$, we find the standard interpolation polynomial.

Examples in Matlab

The command $y = \text{polyval}(p, x)$ calculates the values y of the polynomial of degree n, where the $n+1$ coefficients are stored in the vector p, at the point x:

$$y = p(1) \cdot x^n + p(2) \cdot x^{n-1} + \ldots + p(n) \cdot x + p(n+1).$$

Example C. We want to evaluate the polynomial from Example A at the point $x = 0.4$ and then draw a graph. You can use the following commands:

- `x = 0.4;`
- `y = polyval(p1, x)`

The output will be:

$$y = 3.9012$$

Now, to plot the graph of the polynomial, use:

- `x = linspace(0, 1, 100);`
- `y = polyval(p1, x);`
- `plot(x, y)`

Error

Definition 3.4 Given continuous $f(x)$, $I = [a, b]$, $n+1$ nodes s.t. $a = x_0 < x_1 < \cdots < x_n = b$, we define the error function

$$E_n f(x) := f(x) - \Pi_n f(x)$$

associated to the interpolating polynomial $\Pi_n f(x)$. The error is

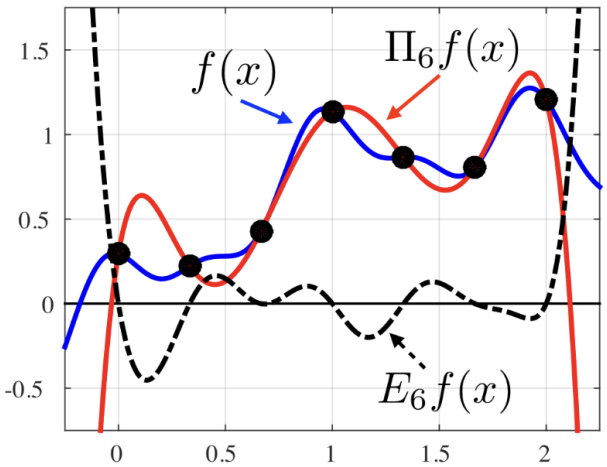
$$e_n(f) := \max_{x \in I} |E_n f(x)|.$$

Example: $f(x) = \sin(x) + \frac{1}{4}\sin\left(2\pi x + \frac{\pi}{3}\right) + \frac{1}{10}\sin\left(4\pi x + \frac{7\pi}{10}\right)$

$\Pi_6 f(x)$ of $f(x)$ at the nodes

$$x_0 = 0, \quad x_1 = \frac{1}{3}, \quad x_2 = \frac{2}{3}, \quad x_3 = 1,$$

$$x_4 = \frac{4}{3}, \quad x_5 = \frac{5}{3}, \quad x_6 = 2$$



Error

Definition 3.4 Given continuous $f(x)$, $I = [a, b]$, $n+1$ nodes s.t. $a = x_0 < x_1 < \cdots < x_n = b$, we define the error function

$$E_n f(x) := f(x) - \Pi_n f(x)$$

associated to the interpolating polynomial $\Pi_n f(x)$. The error is

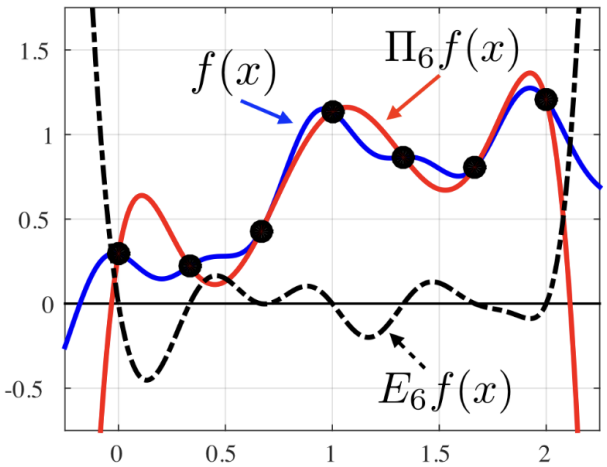
$$e_n(f) := \max_{x \in I} |E_n f(x)|.$$

Example: $f(x) = \sin(x) + \frac{1}{4}\sin\left(2\pi x + \frac{\pi}{3}\right) + \frac{1}{10}\sin\left(4\pi x + \frac{7\pi}{10}\right)$

$\Pi_6 f(x)$ of $f(x)$ at the nodes

$$x_0 = 0, \quad x_1 = \frac{1}{3}, \quad x_2 = \frac{2}{3}, \quad x_3 = 1,$$

$$x_4 = \frac{4}{3}, \quad x_5 = \frac{5}{3}, \quad x_6 = 2$$



Interpolation of a smooth function

Proposition 3.2 Consider $n+1$ distinct nodes in $I=[a,b]$ s.t. $a=x_0<x_1<\cdots<x_n=b$ and the polynomial interpolant $\Pi_n f(x)$ of $f(x)$ in such nodes.
If $f\in C^{n+1}(I)$, for all $x\in I$, there exists $\xi=\xi(x)\in I$ s.t.:

$$E_n f(x)=\frac{1}{(n+1)!}f^{(n+1)}(\xi(x))w_n(x),\ \text{where } w_n(x):=\prod_{i=0}^{n}(x-x_i) \qquad (2)$$

The error $e_n(f)$ is bounded by the error estimator $\tilde{e}_n(f)$ as:

$$e_n(f)\le\tilde{e}_n(f):=\frac{1}{(n+1)!}\max_{x\in I}\left|f^{(n+1)}(x)\right|\max_{x\in I}|w_n(x)|.$$

Interpolation of a smooth function

Proposition 3.2 Consider $n+1$ distinct nodes in $I = [a, b]$ s.t. $a = x_0 < x_1 < \cdots < x_n = b$ and the polynomial interpolant $\Pi_n f(x)$ of $f(x)$ in such nodes.
If $f \in C^{n+1}(I)$, for all $x \in I$, there exists $\xi = \xi(x) \in I$ s.t.:

$$E_n f(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi(x)) w_n(x), \text{ where } w_n(x) := \prod_{i=0}^{n} (x - x_i) \qquad (2)$$

The error $e_n(f)$ is bounded by the error estimator $\tilde{e}_n(f)$ as:

$$e_n(f) \leq \tilde{e}_n(f) := \frac{1}{(n+1)!} \max_{x \in I} \left| f^{(n+1)}(x) \right| \max_{x \in I} |w_n(x)|.$$

Interpolation error of a smooth function

Proposition 3.3

Let $x_0, x_1, \ldots, x_n$ be $n+1$ uniformly distributed interpolation nodes in $I = [a, b]$. Let $f \in C^{n+1}(I)$. Then

$$e_n(f) = \max_{x \in I} |f(x) - \Pi_n f(x)| \leq \frac{1}{4(n+1)} \left(\frac{b-a}{n} \right)^{n+1} \max_{x \in I} |f^{(n+1)}(x)|. \quad (4)$$

Note that the interpolation error depends on the $n+1$-th derivative of f.

Result obtained since

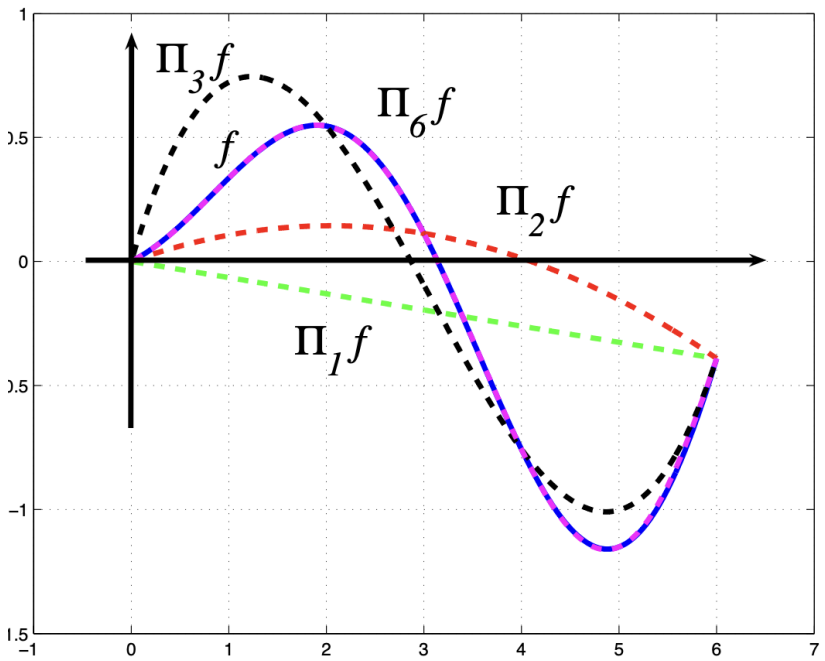
$$\max_{x \in I} |w_n(x)| \leq \frac{n!}{4} \left(\frac{b-a}{n} \right)^{n+1}.$$

Example 3

Interpolation polynomials $\Pi_i f$ for $i = 1, 2, 3, 6$ and

$$f(x) = \frac{x+1}{5}\sin(x),$$

with uniformly distributed nodes on $[0, 6]$.



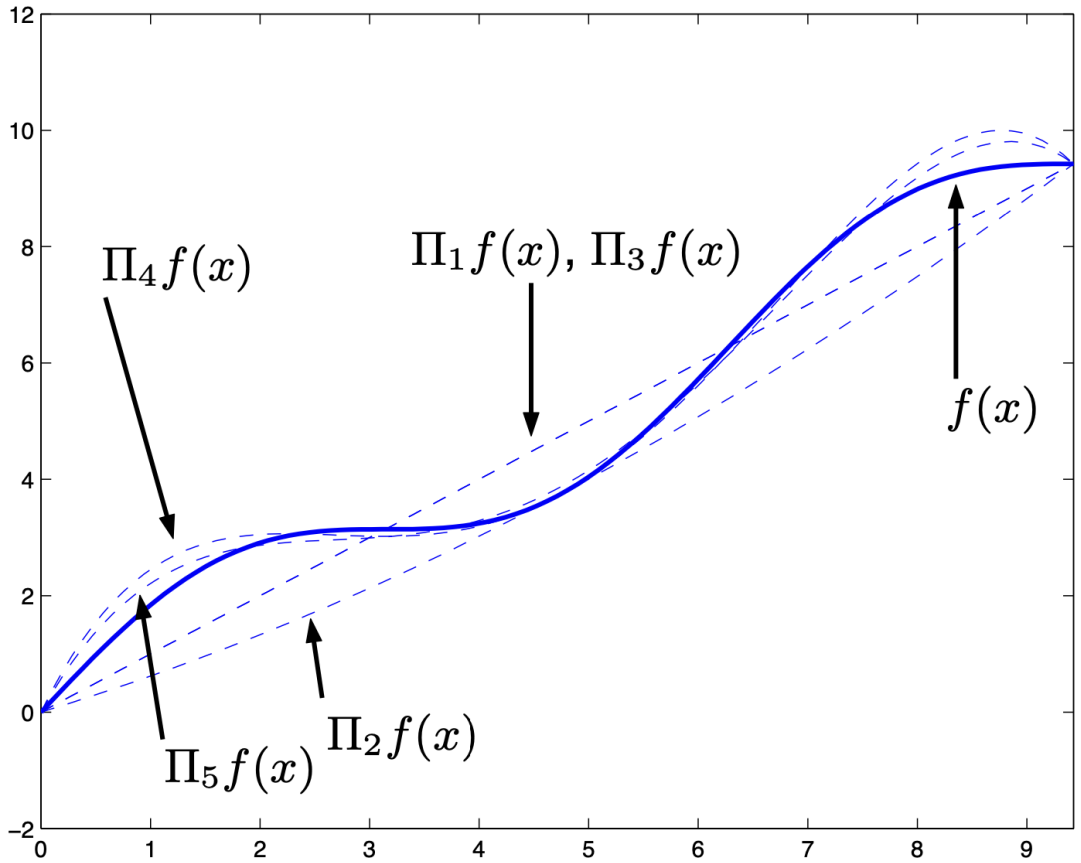
Examples in Matlab

Example 4. We want to interpolate the function $\sin(x) + x$ using $n = 2, 3, 4, 5, 6$ nodes. In MATLAB, you can use the command `polyfit` to calculate the coefficients of the interpolating polynomial and `polyval` to evaluate a polynomial with known coefficients at the set of nodes. Here are the commands in MATLAB:

```
f = @(x) sin(x) + x;
x_sample = [0:3*pi/100:3*pi];
plot(x_sample, f(x_sample), 'b'); hold on
for i = 2:6
x = linspace(0, 3*pi, i);
y = f(x);
c = polyfit(x, y, i - 1);
plot(x_sample, polyval(c, x_sample), 'b-')
end
```

Examples in Matlab

The following figure shows the 5 polynomials that we obtained.



Error

From Proposition 3.3 we have:

$$e_n(f) = \max_{x \in I} |f(x) - \Pi_n f(x)| \leq \frac{1}{4(n+1)} \left(\frac{b-a}{n} \right)^{n+1} \max_{x \in I} |f^{(n+1)}(x)|. \quad (4)$$

Remark The fact that

$$\lim_{n \to \infty} \frac{1}{4(n+1)} \left(\frac{b-a}{n} \right)^{n+1} = 0$$

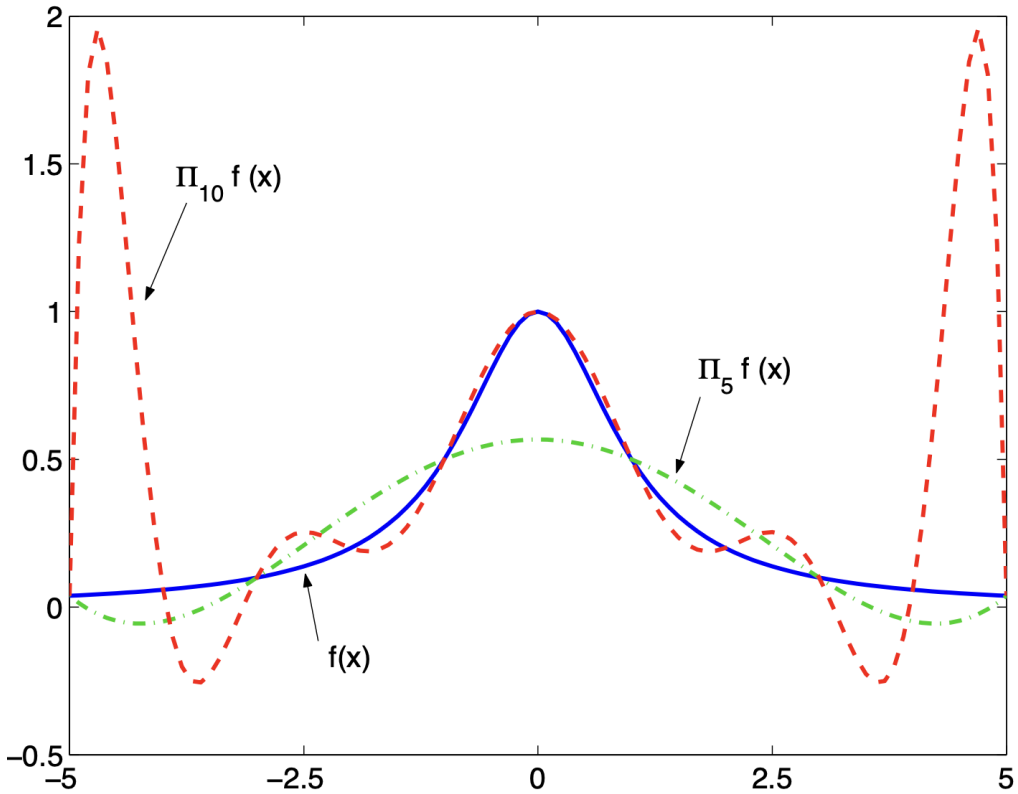
does not imply that $e_n(f)$ goes to zero as $n \to \infty$.
If the growth of $\max_{x \in I} |f^{(n+1)}(x)|$ is not compensated by the decrease in $\frac{1}{4(n+1)} \left(\frac{b-a}{n} \right)^{n+1}$, the error $e_n(f)$ "blows up" (e.g. Runge's phenomenon)

Runge's phenomenon

Runge function Let

$$f(x) = \frac{1}{1+x^2}, \quad x \in [-5, 5].$$

Problem: The interpolants with uniformly distributed nodes exhibit increasing oscillations with increasing degree of interpolation.



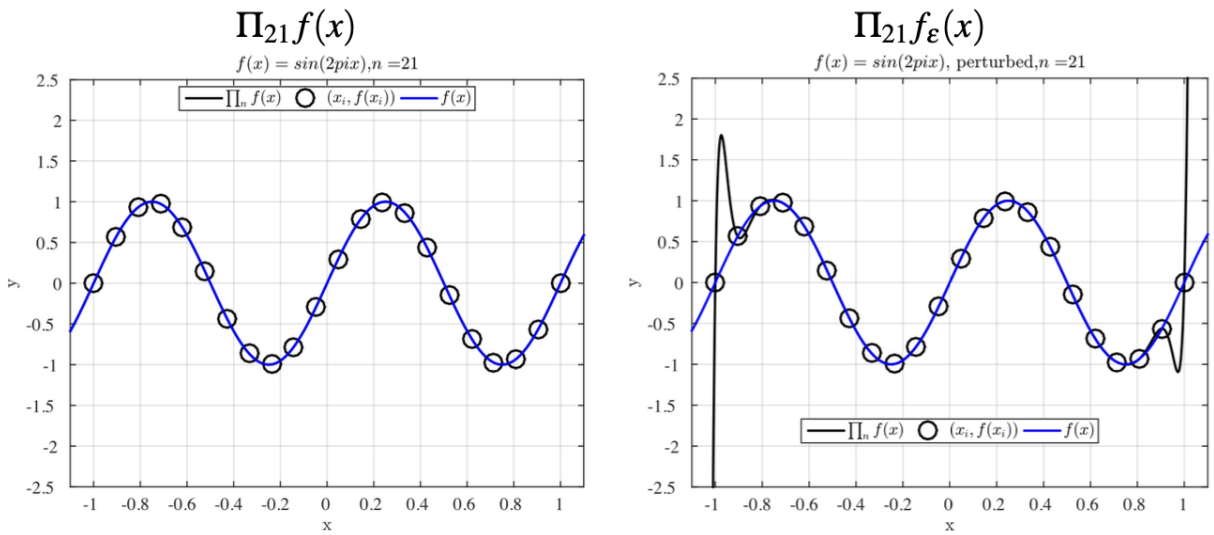
Stability of polynomial interpolation

Problem: Lack of stability for equally spaced nodes
Let $f(x)$ and $f_\epsilon(x) = f(x) + \epsilon(x)$ be a small perturbation.

$$\max_{x\in I} |\Pi_n f(x) - \Pi_n f_\epsilon(x)| = \max_{x\in I} |\sum_{i=0}^{n} (f(x_i) - f_\epsilon(x_i))\phi_i(x)| \leq \Lambda_n(x_0, \ldots, x_n) \max_{0\leq i\leq n} |f(x_i) - f_\epsilon(x_i)|,$$

$$\Lambda_n(x_0, \ldots, x_n) = \max_{x\in I} |\sum_{i=0}^{n} \phi_i(x)| \approx \frac{2^n}{nlog(n)}$$

For large n, a small perturbation of data may lead to large changes in the interpolating polynomial.



$|\epsilon(x)| < 10^{-3}$ for all $x \in I$

Chebyshev–Gauss–Lobatto nodes

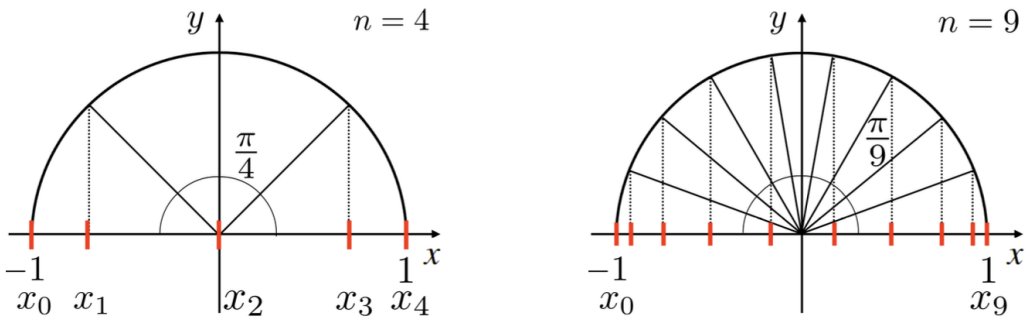
Definition 3.5

For a given $n \geq 1$, the $n+1$ Chebyshev–Gauss–Lobatto nodes in the reference interval $\hat{I} = [-1, 1]$ are:

$$\hat{x}_i = -\cos\left(\frac{\pi i}{n}\right), \quad i = 0, \ldots, n;$$

In general interval $I = [a, b]$, $n+1$ Chebyshev–Gauss–Lobatto nodes are:

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\hat{x}_i, \quad i = 0, \ldots, n.$$



Chebyshev–Gauss–Lobatto nodes

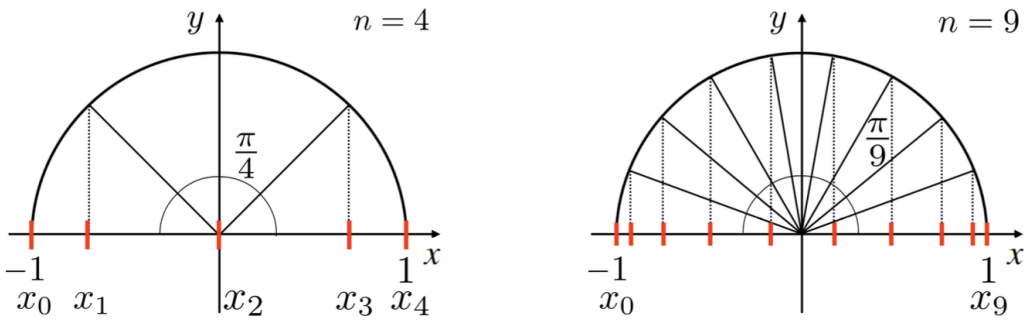
Definition 3.5

For a given $n \geq 1$, the $n+1$ Chebyshev–Gauss–Lobatto nodes in the reference interval $\hat{I} = [-1, 1]$ are:

$$\hat{x}_i = -\cos\left(\frac{\pi i}{n}\right), \quad i = 0, \ldots, n;$$

In general interval $I = [a, b]$, $n+1$ Chebyshev–Gauss–Lobatto nodes are:

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\hat{x}_i, \quad i = 0, \ldots, n.$$



Chebyshev–Gauss–Lobatto nodes

Proposition 3.5

If $f \in C^1(I)$, with the interval $I = [a, b]$ and the $n + 1$ Chebyshev-Gauss-Lobatto nodes are used in I, then

$$\lim_{n \to +\infty} \Pi_n f(x) = f(x) \quad \text{for all } x \in I.$$

Moreover, if $f \in C^\infty(I)$, then

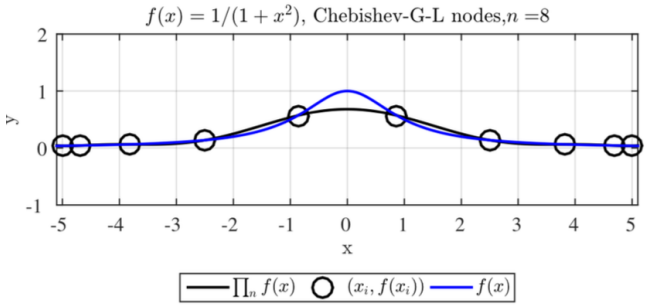
$$\lim_{n \to +\infty} e_n(f) = 0.$$

Furthermore, the stability issues are mitigated.

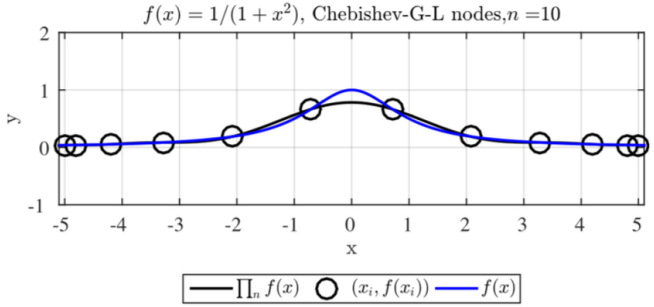
Chebyshev–Gauss–Lobatto nodes: example

Consider again the Runge function $f(x) = \frac{1}{1+x^2}$:

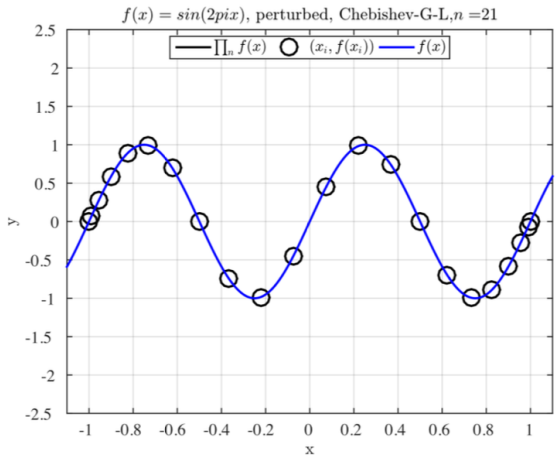
$n = 8$ with CGL nodes



$n = 10$ with CGL nodes



$\Pi_{21} f_\varepsilon(x)$ with CGL nodes



Compute coefficients in polynomial interpolation

- Alternatively to using Lagrange basis, the coefficients $\{a_i\}_{i=0}^n$ of

$$\Pi_n(x) = \sum_{k=0}^{n} y_k \varphi_k(x) = a_0 + a_1 x + \cdots + a_n x^n.$$

  can be computed directly as $a = (a_0, a_1, \ldots, a_n)^T \in \mathbb{R}^{n+1}$ and enforcing the $n+1$ interpolation constraints $P_n(x_i) = y_i$ for all $i = 0, \ldots, n$; i.e., $P_n(x_i) = a_0 + a_1 x_i + \cdots + a_n x_i^n = y_i$ for all $i = 0, \ldots, n$.
- This leads to solving the linear system:

$$Ba = y$$

  where $B \in \mathbb{R}^{(n+1)\times(n+1)}$ is the Vandermonde matrix, with $B_{ij} = (x_{i-1})^{j-1}$ for $i, j = 1, \ldots, n+1$, and $y = (y_0, y_1, \ldots, y_n)^T \in \mathbb{R}^{n+1}$
- Unique solution iff $\det(B) \neq 0$, i.e. iff the $n+1$ nodes $\{x_i\}_{i=0}^n$ are distinct
- Stability issues for relatively "small" values of n, due to large condition number of B.

Plan

Trigonometric interpolation

- The trigonometric interpolant uses trigonometric basis functions (also referred to as the discrete Fourier series)
- Used for periodic signals and functions
- We consider a periodic function $f : [0, 2\pi] \to \mathbb{C}$, i.e., s.t. $f(0) = f(2\pi)$

Trigonometric interpolation

Given $n+1$ nodes $\{x_j\}_{j=0}^n$ s.t. $x_j = jh$ for $j = 0, \ldots, n$ with $h = \frac{2\pi}{n}$, suppose n is even, $n = 2M$, $i^2 = -1$.
The trigonometric interpolant $I_t f(x)$ of the periodic function $f : [0, 2\pi] \to \mathbb{C}$ is

$$I_t f(x) = \frac{a_0}{2} + \sum_{k=1}^{M} (a_k cos(kx) + b_k sin(kx))$$

Use Euler's formula $e^{ikx} = cos(kx) + isin(kx)$ to write

$$I_t f(x) = \sum_{k=-M}^{M} c_k e^{ikx}, \quad a_k = c_k + c_{-k}, \quad b_k = i(c_k - c_{-k})$$

By imposing interpolation conditions $I_t f(x_j) = f(x_j)$ for all $j = 0, \ldots, n$, we find

$$c_k = \frac{1}{n+1} \sum_{j=0}^{n} f(x_j) e^{-ikjh}, \quad h = 2\pi/n$$

Trigonometric interpolation

Given $n+1$ nodes $\{x_j\}_{j=0}^n$ s.t. $x_j = jh$ for $j = 0, \ldots, n$ with $h = \frac{2\pi}{n}$, suppose n is even, $n = 2M$, $i^2 = -1$.
The trigonometric interpolant $I_t f(x)$ of the periodic function $f : [0, 2\pi] \to \mathbb{C}$ is

$$I_t f(x) = \frac{a_0}{2} + \sum_{k=1}^{M} (a_k cos(kx) + b_k sin(kx))$$

Use Euler's formula $e^{ikx} = cos(kx) + isin(kx)$ to write

$$I_t f(x) = \sum_{k=-M}^{M} c_k e^{ikx}, \quad a_k = c_k + c_{-k}, \quad b_k = i(c_k - c_{-k})$$

By imposing interpolation conditions $I_t f(x_j) = f(x_j)$ for all $j = 0, \ldots, n$, we find

$$c_k = \frac{1}{n+1} \sum_{j=0}^{n} f(x_j) e^{-ikjh}, \quad h = 2\pi/n$$

Trigonometric interpolation: remarks

- The trigonometric interpolant $I_t f(x)$ interpolates $f(x)$ at nodes $\{x_j\}_{j=0}^n$,

$$I_t f(x_j) = f(x_j) \text{ for all } j = 0, \ldots, n.$$

- Similar construction if n is odd (see lecture notes)
- Computation of coefficients $\{c_k\}_{k=0}^n$ requires $O(n^2)$ flops
- Using the Fast Fourier Transform (FFT) reduces the cost to $O(n \log n)$ flops.

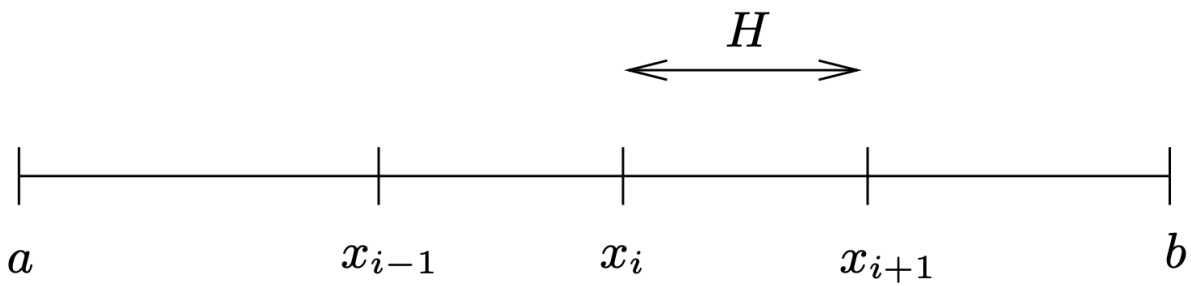
Plan

Piecewise linear interpolation

(Sect. 3.2.3)

Let $x_0 = a < x_1 < \cdots < x_N = b$ be a distribution of nodes that divide the interval $I = [a, b]$ into subintervals $I_i = [x_i, x_{i+1}]$, $i = 0, n$. Let H be the characteristic size of the intervals:

$$H = \max_{i=0,\ldots,n} (x_{i+1} - x_i).$$



The piecewise linear interpolating polynomial $\Pi_1^H(x)$ is a piecewise polynomial of degree 1 such that $\Pi_1^H(x)|_{I_i} \in \mathbb{P}_1$ for all $i = 0, \ldots, n-1$), with:

$$\Pi_1^H(x) = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i) \quad \text{for all} \quad i = 0, \ldots, n-1.$$

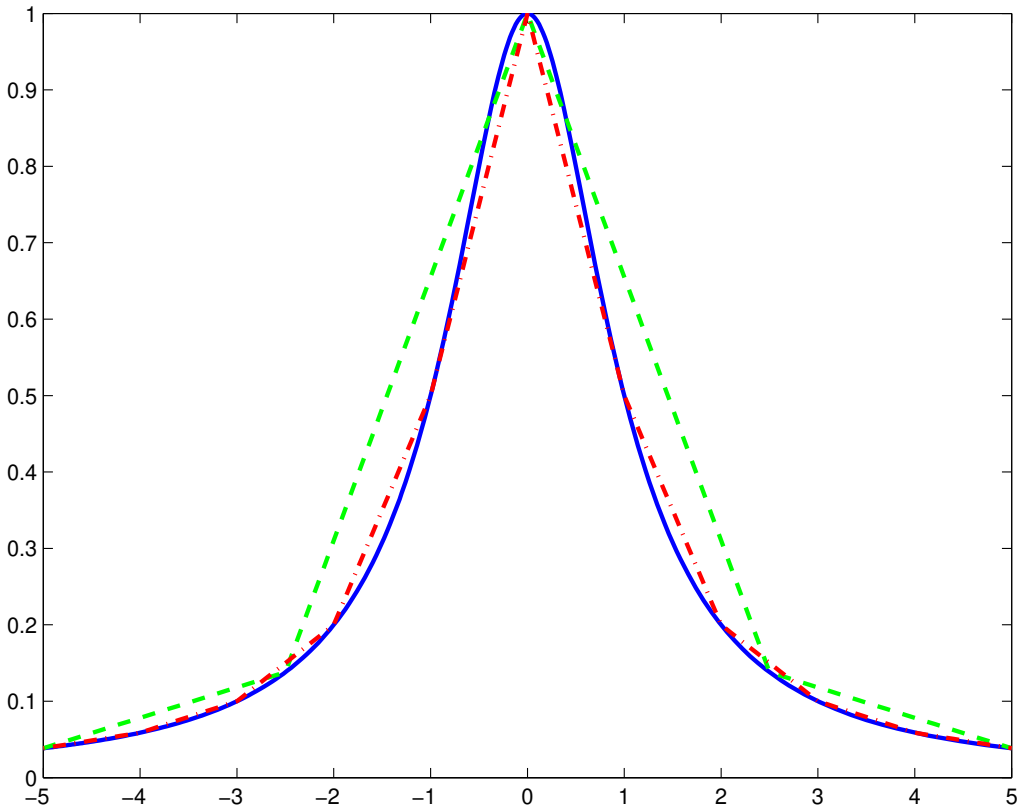
Piecewise linear interpolation

Let $x_0 = a < x_1 < \cdots < x_N = b$ be a distribution of nodes that divide the interval $I = [a, b]$ into subintervals $I_i = [x_i, x_{i+1}]$, $i = 0, n$. Let $f \in C^0(I)$ be known.
On each sub-interval I_i, we interpolate $f_{|I_i}$ by a polynomial of degree 1. The obtained interpolating function is called *piecewise linear interpolation polynomial* of f and is noted $\Pi_1^H f(x)$.

$$\Pi_1^H f(x) = f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i) \quad \text{for } x \in I_i.$$

Runge with piecewise linear interpolation

Piecewise linear interpolation for $f(x) = \dfrac{1}{1+x^2}$, $x \in [-5, 5]$



The figure shows the polynomial $\Pi_1^{H_1} f$ and $\Pi_1^{H_2} f$ for $H_1 = 2.5$ and $H_2 = 1.0$.

Runge with piecewise linear interpolation

The commands in Matlab for this interpolation:

```
>> f = @(x) 1./(1+x.^2);
>> H1 = 2.5; H2=1.0;
>> x1 = [-5:H1:5]; x2 = [-5:H2:5];
>> y1 = f (x1); y2 = f(x2);
>> x_plot = [-5:.1:5];
>> y1_plot = interp1(x1,y1,x_plot);
>> y2_plot = interp1(x2,y2,x_plot);
>> plot(x_plot, f(x_plot)); hold on;
>> plot(x_plot, y1_plot,'r'); plot(x_plot, y2_plot,'g');
```

Error

Theorem (Prop. 3.6)

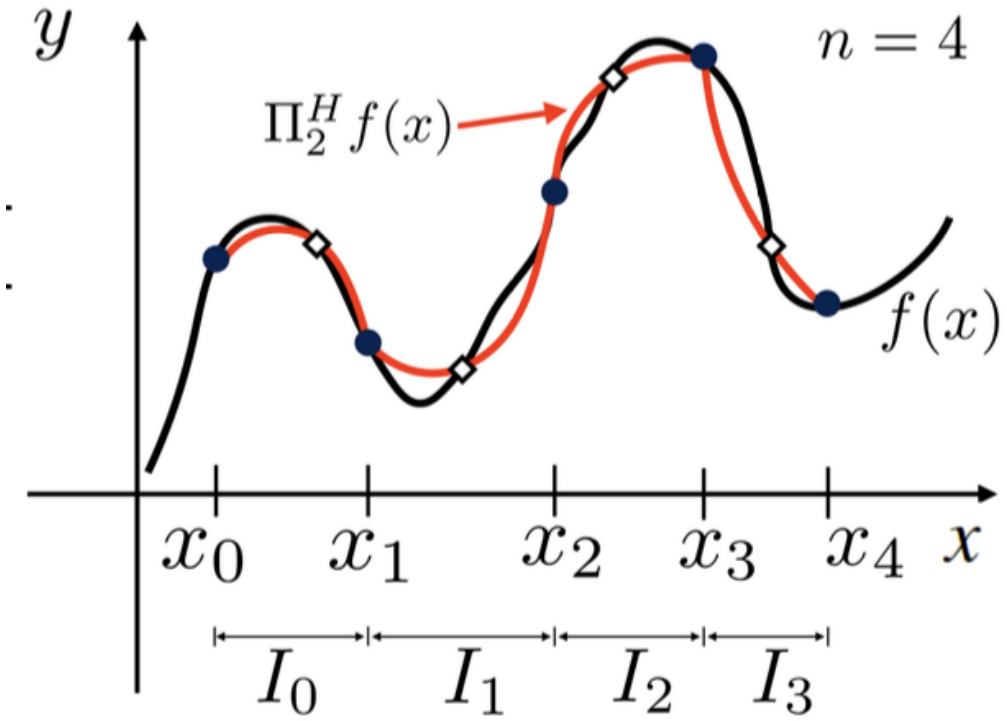
If $f \in C^2(I)$, $(I = [x_0, x_N])$ *then*

$$e_1^H(f) := \max_{x \in I} \mid f(x) - \Pi_1^H f(x) \mid \leq \frac{H^2}{8} \max_{x \in I} |f''(x)|,$$

for which the error converges to zero with order 2 in H *(quadratically)*

Piecewise quadratic polynomials with higher degree

Define the piecewise quadratic polynomial $\Pi_2^H(x)$ as $\Pi_2^H(x)|_{I_i} \in \mathbb{P}_2$ for all the subintervals I_i of I from $i = 0, \ldots, n-1$; it interpolates at the nodes and at intermediate points (e.g. mid-points of the sub-interval).
If $f \in C^0(I)$ is known, then we use the notation $\Pi_2^H f(x)$.



Piecewise interpolating polynomials

Define the piecewise interpolating polynomial of degree $r \geq 1$, $\Pi_r^H(x)$, as $\Pi_r^H(x)|_{I_i} \in \mathbb{P}_r$ for all $i = 0, \ldots, n-1$ (or $\Pi_r^H f(x)$ if $f \in C^0(I)$ is known).

Theorem ((Prop. 3.7))

If $f \in C^{r+1}(I)$, then the error $e_r^H(f) := \max_{x \in I} \left| f(x) - \Pi_r^H f(x) \right|$ associated with the piecewise interpolating polynomial of degree $r \geq 1$, $\Pi_r^H f(x)$, can be bounded by the error estimator $\tilde{e}_r^H(f)$ as:

$$e_r^H(f) \leq \tilde{e}_r^H(f) := C_r H^{r+1} \max_{x \in I} \left| f^{(r+1)}(x) \right|,$$

with C_r a positive constant, for which the error converges to zero with order $r+1$ in H.

Remark 3.7 Piecewise interpolating polynomials $\Pi_r^H f(x)$ of any degree $r \geq 1$ are only C^0–continuous across the subintervals (internal nodes).

Piecewise interpolating polynomials

Define the piecewise interpolating polynomial of degree $r \geq 1$, $\Pi_r^H(x)$, as $\Pi_r^H(x)|_{I_i} \in \mathbb{P}_r$ for all $i = 0, \ldots, n-1$ (or $\Pi_r^H f(x)$ if $f \in C^0(I)$ is known).

Theorem ((Prop. 3.7))

If $f \in C^{r+1}(I)$, then the error $e_r^H(f) := \max_{x \in I} \left| f(x) - \Pi_r^H f(x) \right|$ associated with the piecewise interpolating polynomial of degree $r \geq 1$, $\Pi_r^H f(x)$, can be bounded by the error estimator $\tilde{e}_r^H(f)$ as:

$$e_r^H(f) \leq \tilde{e}_r^H(f) := C_r H^{r+1} \max_{x \in I} \left| f^{(r+1)}(x) \right|,$$

with C_r a positive constant, for which the error converges to zero with order $r+1$ in H.

Remark 3.7 Piecewise interpolating polynomials $\Pi_r^H f(x)$ of any degree $r \geq 1$ are only C^0–continuous across the subintervals (internal nodes).

Runge with piecewise linear interpolation (contd)

We estimate the interpolation error when f is the Runge function

$$\frac{1}{1+x^2}$$

on the interval $[-5,5]$. We take K sub-intervals with $K = 20, 40, 80, 160$ and we estimate the interpolation error $|f(x) - \Pi_1^H f(x)|$ on a fine grid

```
>> f=@(x) 1./(1+x.^2);
>> K=[20 40 80 160]; H=10./K;
>> x_fine = [-5:0.001:5];
>> f_fine = f(x_fine);
>> for i=1:4
    x = [-5:H(i):5]; y = f(x);
    y_fine = interp1(x,y,x_fine);
    err1(i) = max(abs(f_fine - y_fine));
   end
>> loglog(H,err1);
```

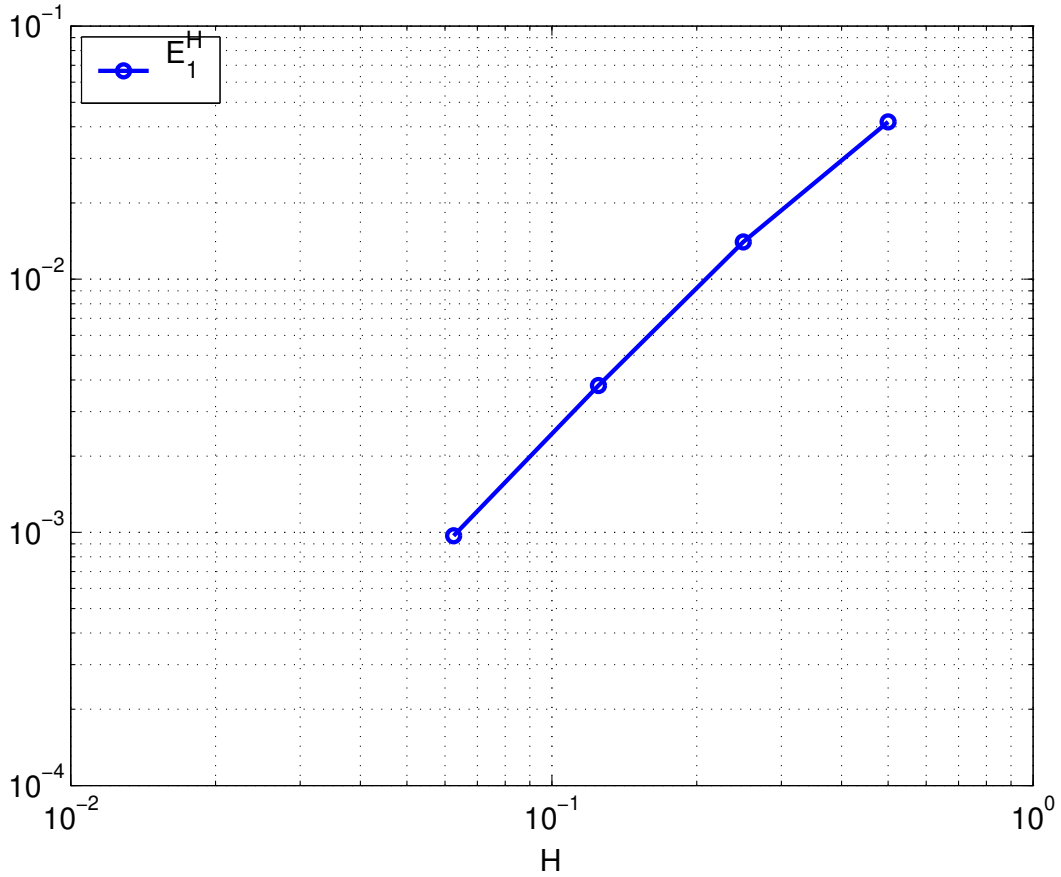
Runge with piecewise linear interpolation (contd)

The following figure shows (in logarithmic scale) the error according to the size $H = 10/K$ of sub-intervalles. We see that the error $\max_{x \in I} \mid e_1^H f(x) \mid$ for piecewise linear interpolation behaves like CH^2: the result is in agreement with Prop. 3.6. In addition, if we calculate the relations $\max_{x \in I} \mid e_1^H f(x) \mid / H^2$, we can estimate the constant C:

```
>> err1./H.^2
ans =
  0.16734   0.22465   0.24330   0.24829
```

Runge with piecewise linear interpolation (contd)

The interpolation error is quadratic in $H = 10/K$ (in the graph, E_1^H displays the error e_1^H from our formulas)



Interpolation by spline functions

(Chapt. 3.2.4 of the book)

Let $a = x_0 < x_1 < \cdots < x_n = b$ be the points that divide the interval $I = [a, b]$ into disjoint intervals $I_i = [x_i, x_{i+1}]$.

Definition

A function s_3 is called a interpolating cubic spline of f if it satisfies:

1. $s_3(x)_{|I_i} \in \mathbb{P}_3$ for all $i = 0, \ldots n-1$, $\mathbb{P}_3$ is the set of polynomials of degree 3,
2. $s_3(x_i) = f(x_i)$ for all $i = 0, \ldots n$,
3. $s_3 \in C^2([a, b])$.

Interpolation by spline functions

- Let $s_3(x)|_{I_i} = a_{0,i} + a_{1,i}x + a_{2,i}x^2 + a_{3,i}x^3$ for $i = 0$ to $n-1$
- Determine $4n$ coefficients $\{a_{j,i}\}$, $j = 0, \ldots 3, i = 0, \ldots, n-1$
- We have to verify the following conditions ($s_3(x_i^-)$ means the one-sided limit (from the right) of s_3 in the point x_i and $s_3(x_i^+)$ means the one-sided limit (from the left)) :

$$
\begin{aligned}
s_3(x_i^-) &= f(x_i) && \text{for all } 1 \le i \le n-1,\\
s_3(x_i^+) &= f(x_i) && \text{for all } 1 \le i \le n-1,\\
s_3(x_0) &= f(x_0), &&\\
s_3(x_n) &= f(x_n), &&\\
s_3'(x_i^-) &= s_3'(x_i^+) && \text{for all } 1 \le i \le n-1,\\
s_3''(x_i^-) &= s_3''(x_i^+) && \text{for all } 1 \le i \le n-1,
\end{aligned}
$$

- That means $2(n-1) + 2 + 2(n-1) = 4n - 2$ conditions.

Interpolation by spline functions

We need to find $4n$ unknowns (which are the 4 coefficients of each of the n restrictions from $s_{3|_{I_i}}$, $i = 0, \ldots n-1$) which satisfy $4n-2$ relations.
We add 2 additional conditions to check:

- If we assume

$$s_3''(x_0^+) = 0 \quad \text{and} \quad s_3''(x_n^-) = 0, \tag{3}$$

  then the spline s_3 is completely determined and called *a natural interpolating cubic spline*.

- Another possibility is to assume the continuity of the third derivative in the nodes x_2 and x_{n-1}. That means:

$$s_3'''(x_2^-) = s_3'''(x_2^+) \quad \text{and} \quad s_3'''(x_{n-1}^-) = s_3'''(x_{n-1}^+). \tag{4}$$

  The conditions (4) are called *not-a-knot*.

Interpolation by spline functions

We need to find $4n$ unknowns (which are the 4 coefficients of each of the n restrictions from $s_{3|_{I_i}}$, $i = 0, \ldots n-1$) which satisfy $4n-2$ relations.
We add 2 additional conditions to check:

- If we assume

$$s_3''(x_0^+) = 0 \quad \text{and} \quad s_3''(x_n^-) = 0, \tag{3}$$

  then the spline s_3 is completely determined and called *a natural interpolating cubic spline*.

- Another possibility is to assume the continuity of the third derivative in the nodes x_2 and x_{n-1}. That means:

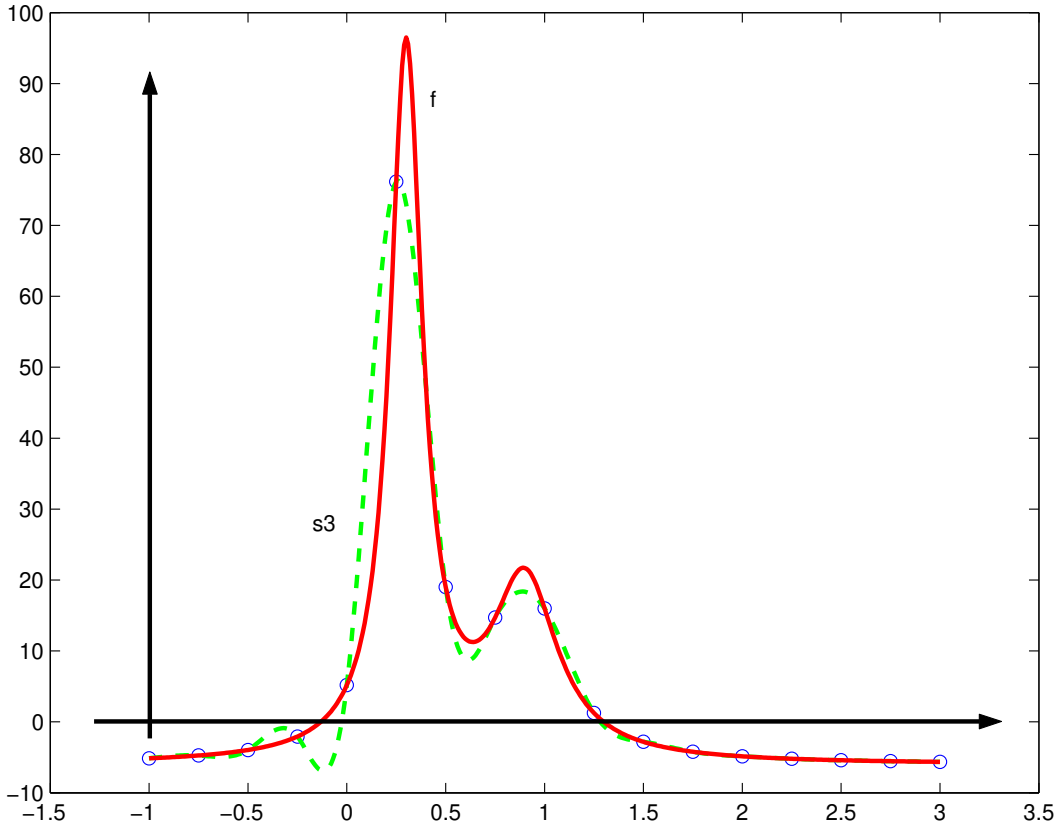
$$s_3'''(x_2^-) = s_3'''(x_2^+) \quad \text{and} \quad s_3'''(x_{n-1}^-) = s_3'''(x_{n-1}^+). \tag{4}$$

  The conditions (4) are called *not-a-knot*.

Example natural interpolating cubic spline

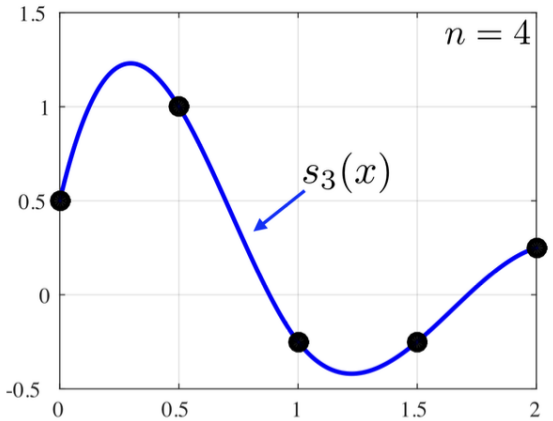
A natural interpolating cubic spline of a function

$$f(x) = \frac{1}{(x-0.3)^2+0.01} + \frac{1}{(x-0.9)^2+0.04} - 6, \text{ (nodes equidistributed } x_j = -1 + j/4,\ j = 0, \ldots, 16)$$



Example not-a-knot interpolating cubic spline

- MATLAB command spline considers not–a–knot interpolating cubic splines
- The cubic spline $s_3(x)$ interpolates the data at the $n+1=5$ nodes $\{x_i\}_{i=0}^{4}$ and is C^2–continuous across each internal node $\{x_i\}_{i=1}^{3}$.



Error estimates

Theorem (Prop. 3.8)

Consider $n+1$ distinct nodes $\{x_i\}_{i=0}^n$ in the interval $I = [x_0, x_n]$, n subintervals $I_i = [x_i, x_{i+1}]$, and $H := \max_{i=0,\dots,n-1} |I_i|$.

If $f \in C^4(I)$ and $s_3(x)$ is its natural interpolating cubic spline at the nodes, then:

$$\max_{x\in I} \left| f^{(k)}(x) - s_3^{(k)}(x) \right| \leq C_k H^{4-k} \max_{x\in I} \left| f^{(4)}(x) \right| \quad \text{for} \quad k = 0, 1, 2,$$

and

$$\max_{x\in I\setminus\{x_1,\dots,x_{n-1}\}} \left| f^{(3)}(x) - s_3^{(3)}(x) \right| \leq C_3 H \max_{x\in I} \left| f^{(4)}(x) \right|,$$

with $C_k > 0$ positive constants, for which the convergence order of the error is $4-k$ in H, depending on the order of derivation $k = 0, 1, 2, 3$.

Error estimates

Theorem (Prop. 3.8)

Consider $n+1$ distinct nodes $\{x_i\}_{i=0}^n$ in the interval $I = [x_0, x_n]$, n subintervals $I_i = [x_i, x_{i+1}]$, and $H := \max_{i=0,\dots,n-1} |I_i|$.

If $f \in C^4(I)$ and $s_3(x)$ is its natural interpolating cubic spline at the nodes, then:

$$\max_{x\in I} \left| f^{(k)}(x) - s_3^{(k)}(x) \right| \leq C_k H^{4-k} \max_{x\in I} \left| f^{(4)}(x) \right| \quad \text{for} \quad k = 0, 1, 2,$$

and

$$\max_{x\in I\setminus\{x_1,\dots,x_{n-1}\}} \left| f^{(3)}(x) - s_3^{(3)}(x) \right| \leq C_3 H \max_{x\in I} \left| f^{(4)}(x) \right|,$$

with $C_k > 0$ positive constants, for which the convergence order of the error is $4-k$ in H, depending on the order of derivation $k = 0, 1, 2, 3$.

Plan

Approximation by the least squares method

(Chapt. 3.3 of the book)

Suppose we have $n+1$ points $x_0, x_1, \ldots, x_n$ and $n+1$ values $y_0, y_1, \ldots, y_n$. We have seen that if n is large, the interpolating polynomial may show large oscillations

Instead of interpolating the values, it is possible to define a polynomial of degree $m < n$ that approximates the data "at best"

Least squares method

Definition

We call *least squares polynomial approximation of degree* $m \geq 0$ the polynomial $\tilde{f}_m(x)$ of degree m such that

$$\sum_{i=0}^{n}(y_i - \tilde{f}_m(x_i))^2 \leq \sum_{i=0}^{n}(y_i - p_m(x_i))^2 \qquad \forall p_m(x) \in \mathbb{P}_m$$

Remark

When $y_i = f(x_i)$ *(*f *is a continuous function) then* $\tilde{f}_m$ *is called the approximation of* f *in the least squares sense.*

Least squares method

Definition

We call *least squares polynomial approximation of degree* $m \geq 0$ the polynomial $\tilde{f}_m(x)$ of degree m such that

$$\sum_{i=0}^{n}(y_i - \tilde{f}_m(x_i))^2 \leq \sum_{i=0}^{n}(y_i - p_m(x_i))^2 \qquad \forall p_m(x) \in \mathbb{P}_m$$

Remark

When $y_i = f(x_i)$ *(*f *is a continuous function) then* $\tilde{f}_m$ *is called the approximation of* f *in the least squares sense.*

Least squares method

Remark

If $m = n$ and the nodes are distinct, then $\tilde{f}_m(x) = \Pi_m f(x)$ is the interpolating polynomial of degree m.

- Remember we use the same command in Matlab polyfit as for Lagrange interpolation.
- In general $m \ll n$ and $\tilde{f}_m(x)$ does not interpolate the data.

Least squares method

In other words, the least squares polynomial approximation is the polynomial of degree m that minimizes the distance to the data.
Let note $\tilde{f}_m(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m$ and define the function

$$\Phi(b_0, b_1, \ldots, b_m) = \sum_{i=0}^{n} \left| y_i - \left(b_0 + b_1 x_i + b_2 x_i^2 + \ldots + b_m x_i^m \right) \right|^2$$

Least squares method

The least-squares method consists in determining the coefficients vector $\mathbf{a} = (a_0, \ldots, a_m)^T$ of the polynomial $\tilde{f}_m(x)$ of degree m such that:

$$\phi(\mathbf{a}) = \min_{\mathbf{b} \in \mathbb{R}^{m+1}} \phi(\mathbf{b}).$$

Since ϕ is differentiable, the previous minimization problem is equivalent to solving the following differential problem:

$$\text{find } \mathbf{a} \in \mathbb{R}^{m+1} \text{ such that}$$

$$\frac{\partial \phi}{\partial b_j}(\mathbf{a}) = 0 \quad \text{for all } j = 0, \ldots, m. \tag{5}$$

Finding the coefficients

We can write for each $k = 0, \ldots, m$,

$$\begin{aligned}
\frac{\partial \Phi}{\partial a_k} &= -2 \sum_{i=0}^{n} x_i^k [y_i - (a_0 + \ldots + a_m x_i^m)] \\
&= -2 \left[\sum_{i=0}^{n} x_i^k y_i - \left(a_0 \sum_{i=0}^{n} x_i^k + a_1 \sum_{i=0}^{n} x_i^{k+1} + \ldots + a_m \sum_{i=0}^{n} x_i^{k+m} \right) \right]
\end{aligned}$$

We obtain the linear system with the unknowns $a_0, \ldots, a_m$:

$$\begin{cases}
a_0(n+1) + a_1 \sum_{i=0}^{n} x_i + \ldots + a_m \sum_{i=0}^{n} x_i^m = \sum_{i=0}^{n} y_i \\
a_0 \sum_{i=0}^{n} x_i + a_1 \sum_{i=0}^{n} x_i^2 + \ldots + a_m \sum_{i=0}^{n} x_i^{m+1} = \sum_{i=0}^{n} y_i x_i \\
\vdots \qquad\qquad \vdots \\
a_0 \sum_{i=0}^{n} x_i^m + a_1 \sum_{i=0}^{n} x_i^{m+1} + \ldots + a_m \sum_{i=0}^{n} x_i^{2m} = \sum_{i=0}^{n} y_i x_i^m
\end{cases}$$

Finding the coefficients

We can write for each $k = 0, \ldots, m$,

$$\begin{aligned}
\frac{\partial \Phi}{\partial a_k} &= -2 \sum_{i=0}^{n} x_i^k [y_i - (a_0 + \ldots + a_m x_i^m)] \\
&= -2 \left[\sum_{i=0}^{n} x_i^k y_i - \left(a_0 \sum_{i=0}^{n} x_i^k + a_1 \sum_{i=0}^{n} x_i^{k+1} + \ldots + a_m \sum_{i=0}^{n} x_i^{k+m} \right) \right]
\end{aligned}$$

We obtain the linear system with the unknowns $a_0, \ldots, a_m$:

$$\begin{cases}
a_0(n+1) + a_1 \sum_{i=0}^{n} x_i + \ldots + a_m \sum_{i=0}^{n} x_i^m = \sum_{i=0}^{n} y_i \\
a_0 \sum_{i=0}^{n} x_i + a_1 \sum_{i=0}^{n} x_i^2 + \ldots + a_m \sum_{i=0}^{n} x_i^{m+1} = \sum_{i=0}^{n} y_i x_i \\
\vdots \qquad\qquad \vdots \\
a_0 \sum_{i=0}^{n} x_i^m + a_1 \sum_{i=0}^{n} x_i^{m+1} + \ldots + a_m \sum_{i=0}^{n} x_i^{2m} = \sum_{i=0}^{n} y_i x_i^m
\end{cases}$$

Least squares method

We obtain the linear system $A\mathbf{a} = \mathbf{q}$, where $A \in \mathbb{R}^{(m+1)\times(m+1)}$ and $\mathbf{a} = (a_0, \ldots, a_m)^T$,

$$A = \begin{pmatrix} n+1 & \cdots & \sum_{i=0}^{n} x_i^m \\ \vdots & & \vdots \\ \sum_{i=0}^{n} x_i^m & \cdots & \sum_{i=0}^{n} x_i^{2m} \end{pmatrix} \quad \text{et} \quad \mathbf{q} = \begin{pmatrix} \sum_{i=0}^{n} y_i \\ \vdots \\ \sum_{i=0}^{n} y_i x_i^m \end{pmatrix} \tag{6}$$

For $m = 1$, we obtain the *regression line* $\tilde{f}_1(x) = a_0 + a_1 x$ where $\{a_0, a_1\}$ are solutions of the linear system:

$$\begin{pmatrix} n+1 & \sum_{i=0}^{n} x_i \\ \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} x_i^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{n} y_i \\ \sum_{i=0}^{n} y_i x_i \end{pmatrix} \tag{7}$$

Normal equations

Ideally, for $\tilde{f}_m(x) = a_0 + a_1 x + a_2 x^2 + \ldots$ we would like to impose $\tilde{f}_m(x_i) = y_i$ for $i = 0, \ldots, n$.
This can be written as a system with unknowns a_k, $k = 0, ..., m$: $B\mathbf{a} = \mathbf{y}$, where B is a matrix of dimension $(n+1) \times (m+1)$

$$B = \begin{pmatrix} 1 & x_0 & \ldots & x_0^m \\ 1 & x_1 & \ldots & x_1^m \\ \vdots & & & \vdots \\ 1 & x_n & \ldots & x_n^m \end{pmatrix}$$

Since $m < n$, the system is overdetermined. The solution to (5) is equivalent to the square system (*system of normal equations*)

$$B^T B\mathbf{a} = B^T\mathbf{y}, \quad A = B^T B, \quad \mathbf{q} = B^T\mathbf{y}$$

Example regression line

Example Let $n = 2$ and $m = 1$, nodes $x_0 = 1$, $x_1 = 3$, $x_2 = 4$ with values $y_0 = 0$, $y_1 = 2$, $y_2 = 7$. We want to compute the (*regression line*), i.e., the the least squares polynomial approximation of degree 1, $\tilde{f}_1(x) = a_0 + a_1 x$.

We set $\Phi(a_0, a_1) = \sum_{i=0}^{2} [y_i - (a_0 + a_1 x_i)]^2$ and impose $\dfrac{\partial \Phi}{\partial a_0} = 0$ and $\dfrac{\partial \Phi}{\partial a_1} = 0$:

$$\begin{aligned}
\frac{\partial \Phi}{\partial a_0} &= -2 \sum_{i=0}^{2} [y_i - (a_0 + a_1 x_i)] = -2 \left(\sum_{i=0}^{2} y_i - 3a_0 - a_1 \sum_{i=0}^{2} x_i \right) \\
&= -2(9 - 3a_0 - 8a_1) \\
\frac{\partial \Phi}{\partial a_1} &= -2 \sum_{i=0}^{2} x_i [y_i - (a_0 + a_1 x_i)] = -2 \left(\sum_{i=0}^{2} x_i y_i - a_0 \sum_{i=0}^{2} x_i - a_1 \sum_{i=0}^{2} x_i^2 \right) \\
&= -2(34 - 8a_0 - 26a_1)
\end{aligned}$$

Hence the coefficients a_0 and a_1 are the solution of the system

$$\begin{cases} 3a_0 + 8a_1 = 9 \\ 8a_0 + 26a_1 = 34 \end{cases}$$

Examples

Example 6 We have 8 measures (σ-ϵ):

```
>> sigma = [0.00 0.06 0.14 0.25 0.31 0.47 0.50 0.70];
>> epsilon = [0.00 0.08 0.14 0.20 0.22 0.26 0.27 0.29];
```

We want to extrapolate the value of ϵ for $\sigma = 0.4$. We consider two ways:

- compute the interpolating polynomial Π_7 of degree 7
- compute the least squares polynomial approximation of degree 1

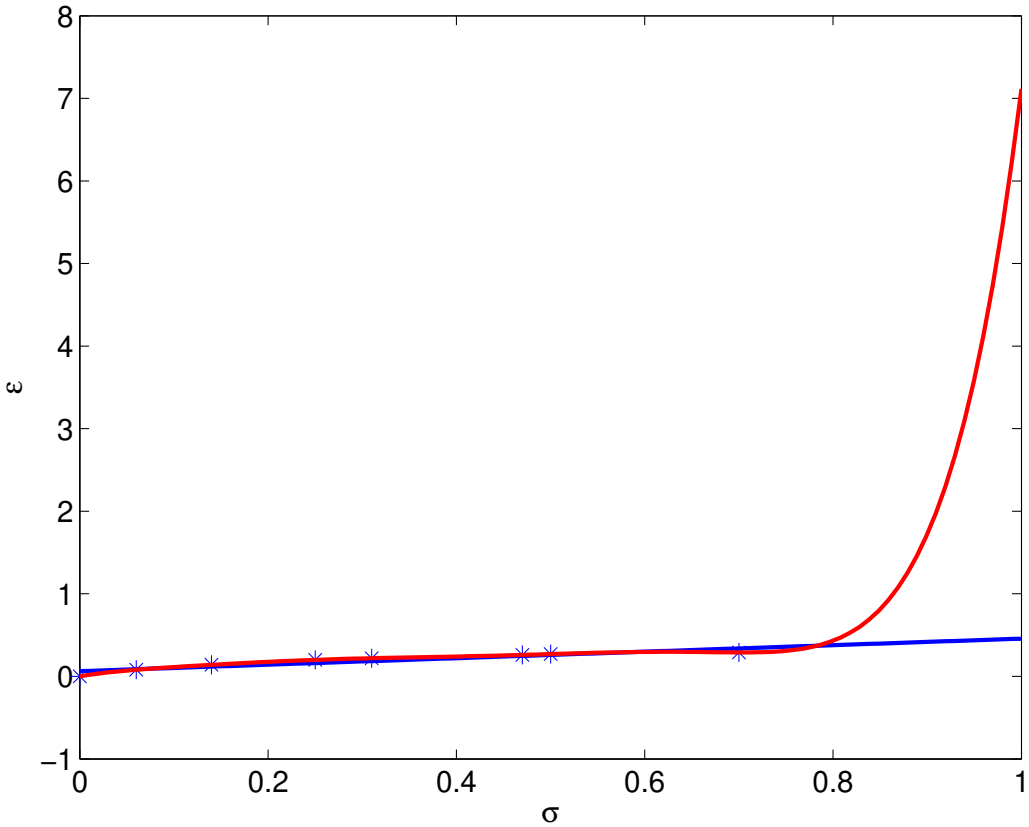
On peut utiliser les commandes suivantes:

```
>>  plot(sigma, epsilon, '*'); hold on;  % plotting the known values
>>  sigma_sample = linspace(0,1.0,100);
>>  p7 = polyfit(sigma, epsilon, 7);
>>  pol = polyval(p7, sigma_sample);     % interpolating polynomial
>>  plot(sigma_sample,pol, 'r');
>>  p1 = polyfit(sigma,epsilon,1);
>>  pol_mc = polyval(p1, sigma_sample);  % least square
>>  plot(sigma_sample, pol_mc, 'g'); hold off;
```

Examples

Example 6 (contd)

- the interpolating polynomial Π_7 of degree 7 is in red
- the least squares polynomial approximation of degree 1 is in blue



For $\sigma > 0.7$,the behavior of the two polynomials are very different.

Examples

Example 6 (contd): In particular, for $\sigma = 0.9$ the values of $\epsilon(\sigma)$ extrapolated with the two methods are

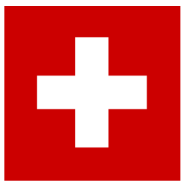
```
>> polyval(p7, 0.9)
ans =
   1.7221

>> polyval(p1, 0.9)
ans =
    0.4173
```

- The value obtained by Π_7 (172.21%) is unrealistic
- On the contrary, the value obtained with the regression line is more appropriate to compute the value at $\sigma = 0.9$.

Examples: swiss population

Example 1 (from last lecture): Starting from the population at the 20th century decades, we extrapolate the Swiss popululation this year. We use a least square polynomial approximation of degree 2

```
>> year = [1900, 1910, 1920, 1930, 1941, 1950,...
           1960, 1970, 1980, 1990, 2000];
>> population = [3315, 3753, 3880, 4066, 4266, 4715,...
           5429, 6270, 6366, 6874, 7288];
>> p2 = polyfit(year, population, 2);
>> year_sample = [1900:2:2010];
>> vp2 = polyval(p2, year_sample);
>> plot(year, population, '*r', ...
       year_sample, vp2, 'b');
```



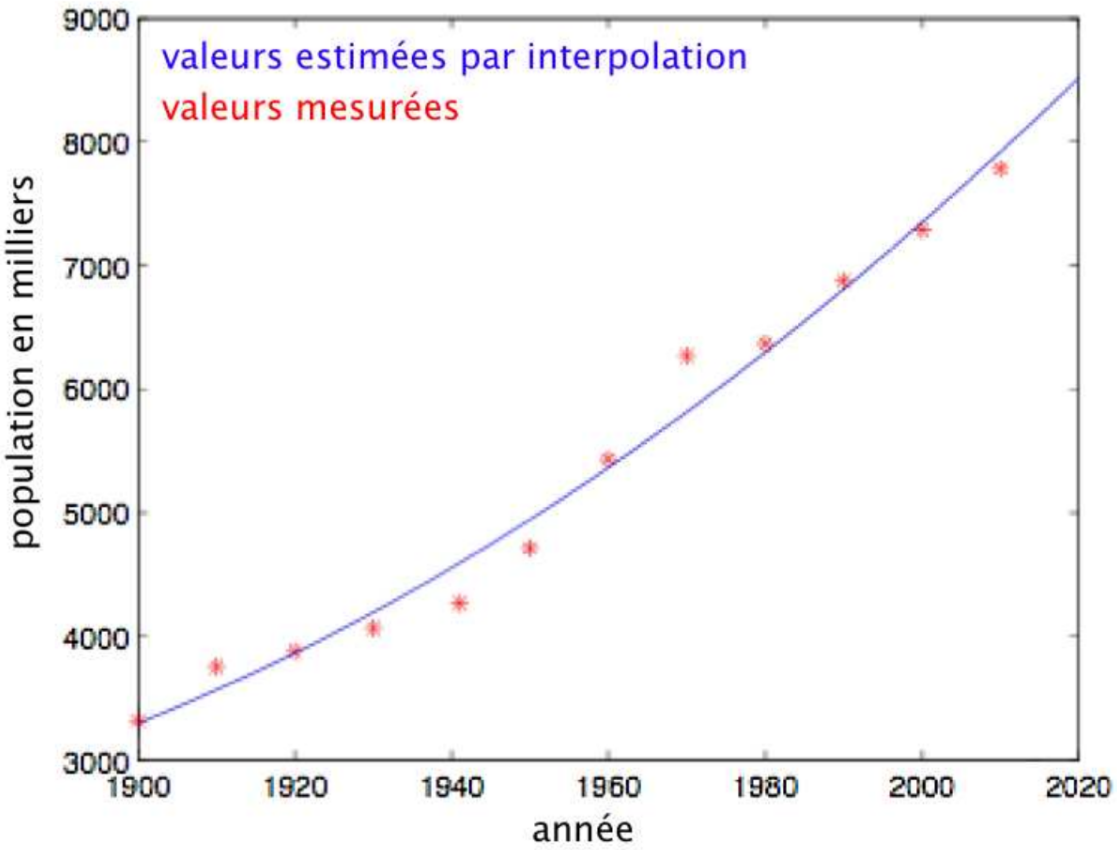
year	1900	1910	1920	1930	1941	1950
population	3315	3753	3880	4066	4266	4715

zear	1960	1970	1980	1990	2000	2010
population	5429	6270	6366	6874	7288	7783

Examples: swiss population

The polynomial of degree two (parabola) which approximates the data by the method of least squares is:

$$p(x) = 0.15x^2 - 549.9x + 501600$$



Processing of polynomials

In Matlab, there are specific commands for doing calculations with polynomials. Let `x` be a vector of abscissas, `y` be a vector of ordinates and `p` (respectively $\mathtt{p}_i$) be the vector of coefficients of a polynomial $P(x)$ (respectively P_i); then, we have following commands:

command	action
`y=polyval(p,x)`	`y` = values of $P(x)$
`p=polyfit(x,y,n)`	`p` = coefficients of the interpolating polynomial Π_n
`z=roots(p)`	`z` = zeros of P such that $P(\mathtt{z}) = 0$
`p=conv(p1,p2)`	`p` = coefficients of the polynomial $P_1 P_2$
`[q,r]=deconv(p1,p2)`	`q` = coefficients of Q, `r` = coefficients of R such that $P_1 = QP_2 + R$
`y=polyderiv(p)`	`y` = coefficients of $P'(x)$
`y=polyinteg(p)`	`y` = coefficients of $\int P(x)\ dx$

Remarks

- Least squares approximation is not limited to polynomials
- They can be done with trigonometric, exponentials, logarithmic
- Interpolation and least squares approximation can be generalized to 2D, 3D, higher dimensions.