①

ETH Lecture 401-0663-00L Numerical Methods for CSE

Numerical Methods for Computational Science and Engineering

Prof. R. Hiptmair, SAM, ETH Zurich

(with contributions from Prof. P. Arbenz and Dr. V. Gradinaru)

Autumn Term 2015

URL: http://www.sam.math.ethz.ch/~hiptmair/tmp/NumCSE/NumCSE15.pdf

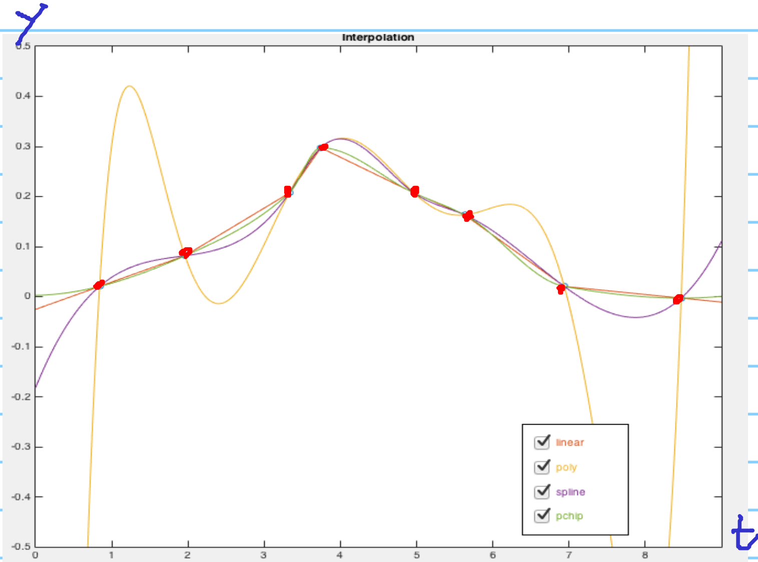
III. Data Interpolation in 1D

Given: data points $(t_i, y_i) \in \mathbb{R}$, $i = 0, \dots, n$,
(t_i: nodes, y_i: values)

Sought: interpolant $f: I \subset \mathbb{R} \to \mathbb{R}$

: $f(t_i) = y_i$, $i = 0, \dots, n$

[interpolation conditions]



Assumption: $t_0 < t_1 < \dots < t_n$ (not for coding!)
$I \subset [t_0, t_1]$

Application: empiric constitutive relations

For non-linear circuit element we measure (U_i, I_i)

Circuit simulation requires function $I = I(U)$, $U \to I(U)$ should be differentiable (Newton's method)

Our setting: interpolant f can be represented as a linear combination of basis function $b_j: I \to \mathbb{R}$

$$f(t) = \sum_{j=0}^{m} c_j \, b_j(t)$$

c_j: coefficients / parameter

b_j: may depend on t_i, must not depend on y_i

Example: $\{t_i\}$-piecewise linear interpolation

Piecewise linear interpolation

= connect data points (t_i, y_i), $i = 1, \dots, m$, $t_{i-1} < t_i$, by line segments

➤ interpolating polygon

Piecewise linear interpolant of data ▷

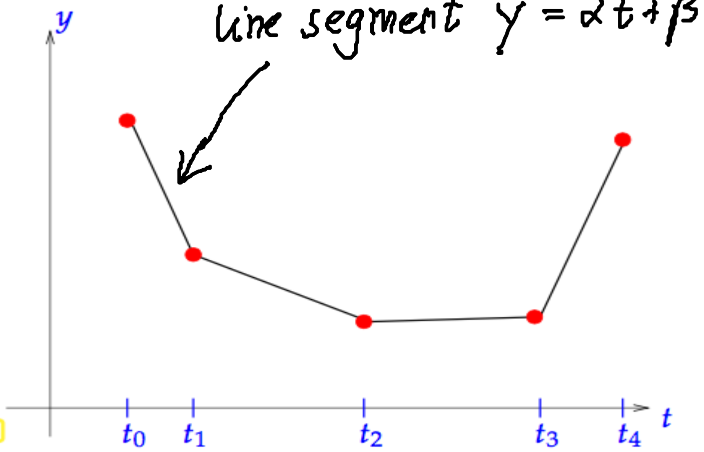
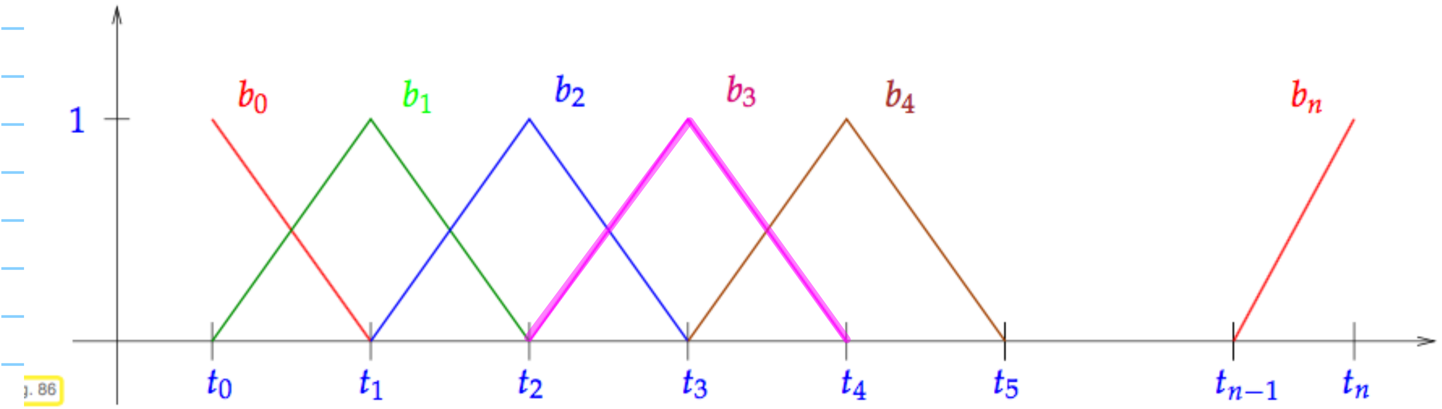


Fig. 85

② Basis functions for p.w. linear ip.: *"tent functions"*



(*) $\quad f(t) = \sum_{j=0}^{n} y_j b_j(t)$ p.w. linear ✓

& $\quad f(t_i) = y_i$, because $\underbrace{b_j(t_i) = \begin{cases} 1, & i=j \\ 0, & \text{else} \end{cases} = \delta_{ij}}$

→ cardinal basis for $\{t_i\}$

Linear interpolation: $\underbrace{\begin{cases} \mathbb{R}^{n+1} \longrightarrow C^0[t_0,t_n] \text{ [cont. fnct.]} \\ \mathbf{y} := [y_i]_{i=0}^{n} \longrightarrow f \text{ by } (*) \end{cases}}$

a linear mapping!

Definition 3.1.14. Linear interpolation operator

An interpolation operator $\mathsf{I} : \mathbb{R}^{n+1} \mapsto C^0([t_0, t_m])$ for the given nodes $t_0 < t_1 < \cdots < t_n$ is called linear, if

addition of functions

$$\mathsf{I}(c\mathbf{y} + \beta\mathbf{z}) = c\,\mathsf{I}(\mathbf{y}) + \beta\,\mathsf{I}(\mathbf{z}) \quad \forall \mathbf{y}, \mathbf{z} \in \mathbb{R}^{n+1},\ c, \beta \in \mathbb{R}\,. \tag{3.1.15}$$

Basis representation (□) + interpolation conditions

$$\sum_{j=0}^{m} c_j b_j(t_i) = y_i \quad , i = 0, \dots, n$$

$$\boxed{\mathbf{A}\mathbf{c} := \begin{bmatrix} b_0(t_0) & \dots & b_m(t_0) \\ \vdots & & \vdots \\ b_0(t_n) & \dots & b_m(t_n) \end{bmatrix} \begin{bmatrix} c_0 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ y_n \end{bmatrix} =: \mathbf{y}}\,. \qquad (\diamond)$$

$\underbrace{f \text{ exists \& unique}}$ ⟺ LSE ($\diamond$) has unique solution

↳ Necessary: $m = n$

3.2. (Global) polynomial interpolation

3.2.1. Polynomials

$$\mathcal{P}_n := \{ t \to p(t) := \sum_{j=0}^{n} \alpha_j t^j,\ \alpha_j \in \mathbb{R} \}$$

↳ degree $\leq n$ polynomials: a vector space

$\dim \mathcal{P}_n = n+1$

monomial basis

Advantages of polynomials:
- smooth C^∞, easy to differentiate
- easy to evaluate *
- ↔ Taylor polynomials

* Horner scheme : "distributive evaluation"

$$p(t) = t(t(\cdots(\alpha_n t+\alpha_{n-1})+a_{n-2})\cdots)+\alpha_0$$

MATLAB-code 3.2.7: Horner scheme (polynomial in MATLAB format, see Rem. 3.2.4)

```
function y = polyval(p,x)
y = p(1); for i=2:length(p), y = x.*y+p(i); end
```

Effort : $O(n)$

3.2.2. Theory of polynomial interpolation

Lagrange polynomial interpolation problem*

Given the simple nodes $t_0,\dots,t_n$, $n \in \mathbb{N}$, $-\infty < t_0 < t_1 < \cdots < t_n < \infty$ and the values $y_0,\dots,y_n \in \mathbb{R}$ compute $p \in \mathcal{P}_n$ such that

$$p(t_j) = y_j \quad \text{for} \quad j = 0,\dots,n\,. \tag{3.2.9}$$

Lagrange polynomials:

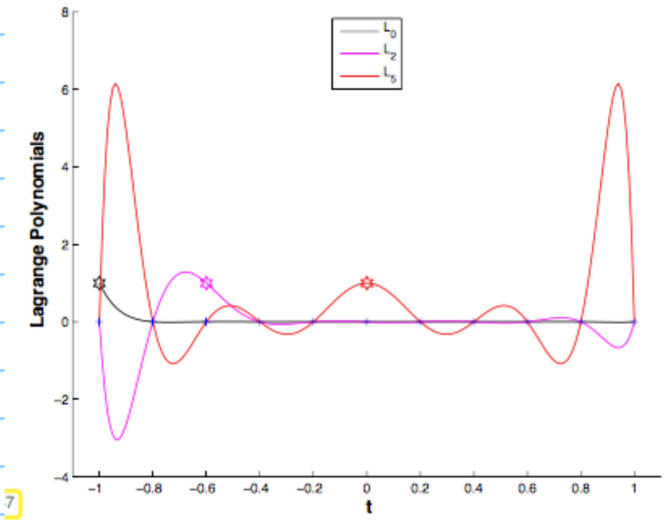
$$L_i(t) = \prod_{\substack{j=0\\ j\neq i}}^{n} \left(\frac{t-t_j}{t_i-t_j}\right) \in \mathcal{P}_n\,, \quad L_i(t_j) = \delta_{ij}$$

$\Downarrow$ cardinal basis of $\mathcal{P}_n$

$$p(t) = \sum_{j=0}^{n} y_j L_j(t) \;\to\; \text{existence} \quad (\heartsuit)$$

$\Downarrow$ uniqueness

* a linear mapping $\mathbb{R}^{n+1} \to \mathcal{P}_n$



◁ Lagrange polynomials for $n = 10$ on $[-1,1]$, equidistant nodes

Generalization : Hermite interpolation problem

Given $t_0,\dots,t_n$ & $y_0,\dots,y_n$: values
$c_0,\dots,c_n$: slopes

Find $p \in \mathcal{P}_{2n+1}$: $p(t_i) = y_i$, $p'(t_i) = c_i$

$\Downarrow$ existence & uniqueness

3.2.3. Algorithms

↳ for Lagr. IP through $(t_i,y_i)_{i=0}^{n}$

④ 3.2.3.1 Multiple evaluations


```
class PolyInterp {
  private:
  // various internal data describing p
  Eigen::VectorXd t;
  public:
    // Constructors taking node vector (t_0,...,t_n) as argument
  PolyInterp(const Eigen::VectorXd &_t);
  template <typename SeqContainer>
  PolyInterp(const SeqContainer &v);
  // Evaluation operator for data (y_0,...,y_n); computes
  // p(x_k) for x_k's passed in x
  Eigen::VectorXd eval(const Eigen::VectorXd &y,const Eigen::VectorXd
     &x) const;
};
```

PolyInterp(...) → called once

eval: many calls; x: long vector, size $N \gg n$

Attempt: ~~• Constructor computes monomial coeffs.~~
~~• eval → Horner scheme~~ } $O(n^3 + Nn)$

Attempt: • Constructor idle
• eval → use (♡) } $O(n + Nn^2)$

Solution : Barycentric formula

$$p(x) = \sum_{i=0}^{n} y_i L_i(x) = \sum_{i=0}^{n} y_i \prod_{j \neq i} \frac{(x-t_j)}{(t_i - t_j)}$$

$$= \prod_{j=0}^{n} (x-t_j) \sum_{i=0}^{n} y_i \frac{1}{x-t_i} \underbrace{\prod_{j \neq i} \frac{1}{(t_i - t_j)}}_{=:\ \lambda_i}$$

$$= \prod_{j=0}^{n} (x-t_j) \sum_{i=0}^{n} \frac{y_i \lambda_i}{x-t_i} = \frac{\sum_{i=0}^{n} \frac{\lambda_i}{x-t_i} y_i}{\sum_{i=0}^{n} \frac{\lambda_i}{x-t_i}}$$

$y_i = 1 \ \forall i \Rightarrow p \equiv 1$

$$1 = \prod_{j=0}^{n} (x-t_j) \sum_{i=0}^{n} \frac{\lambda_i}{x-t_i}$$

$$= \sum_{i=0}^{n} \frac{\lambda_i}{x-t_i} \prod_{j=0}^{n} (x-t_j)$$

$$\Rightarrow \prod_{j=0}^{n} (x-t_j) = \frac{1}{\sum_{i=0}^{n} \frac{\lambda_i}{x-t_i}}$$

λ_i depend only on t_i ⟹ Compute in constructor

In eval() : evaluate $\underline{\underline{}}$ with cost $O(n)$

▷ eval costs $O(Nn)$

5

MATLAB-code 3.2.28: Evaluation of the interpolation polynomials with barycentric formula

```
function p = intpolyval(t,y,x)
% t:  row vector of nodes t_0,...,t_n
% y:  row vector of data y_0,...,y_n
% x:  row vector of evaluation points x_1,...,x_N
n = length(t); % number of interpolation nodes = degree of polynomial
  -1
N = length(x); % Number of evaluation points stored in x
% Precompute the weights λ_i with effort O(n^2)
for k = 1:n
  lambda(k) = 1 / prod(t(k) - t([1:k-1,k+1:n])); end;
for i = 1:N
% Compute quotient of weighted sums of λ_i/(t-t_i), effort O(n)
  z = (x(i)-t); j = find(z == 0);
  if (~isempty(j)), p(i) = y(j); % avoid division by zero
  else
    mu = lambda./z; p(i) = sum(mu.*y)/sum(mu);
  end
end
```

(lines 8–9: $\}\to$ Constructor; lines 10–16: $\}$ eval())

3.2.3.2 "Update-friendly" single evaluation

Given: data points $(t_i, y_i) \in \mathbb{R}^2$, $i = 0, \dots, n$

Lagrangian interp.: find $p \in \mathcal{P}_n$: $p(t_i) = y_i$, $i = 0, \dots, n$

$$p = \sum_{j=0}^{n} y_j L_j \quad [\text{in Lagrangian basis}]$$

```
double eval(const Eigen::VectorXd &t,const Eigen::VectorXd &y,
            double x);
```

↑ evaluation at a single point ("known in advance")

Algorithm: Aitken-Neville

- Partial Lagrange interpolants

$$p_{k,\ell} \in \mathcal{P}_{\ell-k} : \quad p(t_j) = y_j, \quad j = k, \dots, \ell$$

$\longrightarrow$ Recursion formula

$$p_{k,k} \equiv y_k, \quad k = 0, \dots, n$$

$$p_{k,\ell}(x) = \frac{1}{t_\ell - t_k}\underbrace{\{(x-t_k)\,p_{k+1,\ell}(x) - (x-t_\ell)\,p_{k,\ell-1}(x)\}}_{\text{satisfies i.c. for } (t_j, y_j),\ j = k, \dots, \ell}$$

↑ by uniqueness

$n =$	0	1	2	3
t_0	$y_0 =: p_{0,0}(x)$	$\to p_{0,1}(x)$	$\to p_{0,2}(x)$	$\to p_{0,3}(x)$
t_1	$y_1 =: p_{1,1}(x)$	$\to p_{1,2}(x)$	$\to p_{1,3}(x)$	
t_2	$y_2 =: p_{2,2}(x)$	$\to p_{2,3}(x)$		
t_3	$y_3 =: p_{3,3}(x)$			

$p_{0,n} \stackrel{!}{=} p$

⑥

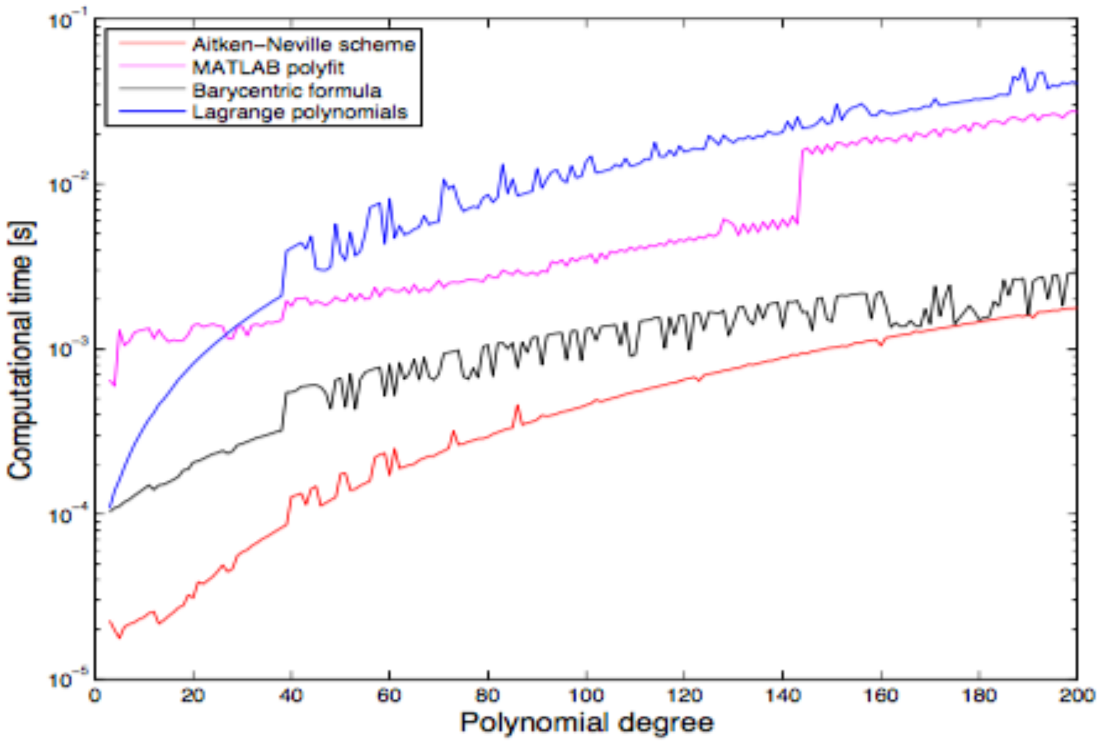
MATLAB-code 3.2.31: Aitken-Neville algorithm Effort : $O(n^2)$

```
function v = ANipoleval(t,y,x)
for i=1:length(y)
  for k=i-1:-1:1
    y(k) = y(k+1)+(y(k+1)-y(k))*(x-t(i))/(t(i)-t(k));
  end
end
v = y(1);
```

"update-friendly": A.N. scheme suitable for simultaneous evaluation & adding of data points

$$
\begin{array}{c|cccc}
n= & 0 & 1 & 2 & 3 \\
\hline
t_0 & y_0 =: p_{0,0}(x) & \to p_{0,1}(x) & \to p_{0,2}(x) & \to p_{0,3}(x) \\
 & & \nearrow & \nearrow & \nearrow \\
t_1 & y_1 =: p_{1,1}(x) & \to p_{1,2}(x) & \to p_{1,3}(x) & \\
 & & \nearrow & \nearrow & \\
t_2 & y_2 =: p_{2,2}(x) & \to p_{2,3}(x) & & \\
 & & \nearrow & & \\
t_3 & y_3 =: p_{3,3}(x) & & &
\end{array}
$$

```
class PolyEval {
  private:
    // evaluation point and various internal data describing the
       polynomials
  public:
    // Constructor taking the evaluation point as argument
    PolyEval(double x);
    // Add another data point and update internal information
    void addPoint(t,y);
    // Value of current interpolating polynomial at x
    double eval(void) const;
};
```



single point evaluation

NCSE15

3.2.3.3. Extrapolation to zero

Objective : Compute $\lim_{h\to 0} \Psi(h)$

↑ Smooth, but hard to evaluate* for $|h| \ll 1$.

Example : Difference quotient $\Psi(h) := \dfrac{f(x+h)-f(x)}{h}$

↳ * cancellation for $h < \sqrt{\text{EPS}}$

Idea: computing inaccessible limit by extrapolation to zero

0: Pick "rather large" $h_0, \dots, h_n$

① evaluation of $\psi(h_i)$ for different h_i, $i = 0, \dots, n$, $|t_i| > 0$.

② $\psi(0) \approx p(0)$ with interpolating polynomial $p \in \mathcal{P}_n$, $p(h_i) = \psi(h_i)$.

Apply to $\psi(h) := \dfrac{f(x+h) - f(x-h)}{2h}$

→ symmetric difference quotient: $\psi(-h) = \psi(h)$

Good setting for extrapolation

$f(x) = \arctan(x)$		$f(x) = \sqrt{x}$		$f(x) = \exp(x)$	
h	Relative error	h	Relative error	h	Relative error
2^{-1}	0.20786640808609	2^{-1}	0.09340033543136	2^{-1}	0.29744254140026
2^{-6}	0.00773341103991	2^{-6}	0.00352613693103	2^{-6}	0.00785334954789
2^{-11}	0.00024299312415	2^{-11}	0.00011094838842	2^{-11}	0.00024418036620
2^{-16}	0.00000759482296	2^{-16}	0.00000346787667	2^{-16}	0.00000762943394
2^{-21}	0.00000023712637	2^{-21}	0.00000010812198	2^{-21}	0.00000023835113
~~2^{-26}~~	~~0.00000001020730~~	~~2^{-26}~~	~~0.00000001923506~~	~~2^{-26}~~	~~0.00000000429331~~
2^{-31}	0.00000005960464	2^{-31}	0.00000001202188	2^{-31}	0.00000012467100
2^{-36}	0.00000679016113	2^{-36}	0.00000198842224	2^{-36}	0.00000495453865

Extrapolation: $h_0 = 1/2$, $h_1 = 1/4$, $h_2 = 1/8$, ...

Degree	Relative error	Degree	Relative error	Degree	Relative error
0	0.04262829970946	0	0.02849215135713	0	0.04219061098749
1	0.02044767428982	1	0.01527790811946	1	0.02129207652215
2	0.00051308519253	2	0.00061205284652	2	0.00011487434095
3	0.00004087236665	3	0.00004936258481	3	0.00000825582406
4	0.00000048930018	4	0.00000067201034	4	0.00000000589624
5	0.00000000746031	5	0.00000001253250	5	0.00000000009546
6	0.00000000001224	6	0.00000000004816	6	0.00000000000002
		7	0.00000000000021		

MATLAB-code 3.2.42: Numerical differentiation by extrapolation to zero

```
function d = diffex(f,x,h0,atol,rtol)
% f:  handle of to a function defined in a neighborhood of x ∈ ℝ,
% x:  point at which approximate derivative is desired,
% h0:  initial distance from x,
% tol:  relative target tolerance
h = h0;
% Aitken-Neville scheme, see Code 3.2.31 (x = 0!)

y(1) = (f(x+h0)-f(x-h0))/(2*h0);
for i=2:10
  h(i) = h(i-1)/2;
  y(i) = f(x+h(i))-f(x-h(i)))/h(i-1);
  for k=i-1:-1:1
    y(k) = y(k+1)-(y(k+1)-y(k))*h(i)/(h(i)-h(k));
  end
  % termination of extrapolation, when desired tolerance is achieved

  errest = abs(y(2)-y(1)); % error indicator
  if ((errest < rtol*abs(y(1))) || (errest < atol)), break; end %
end
d = y(1);
```

"Correction based termination" ⟹ error control !

(8) 3.2.3.4. Newton basis and divided differences

A.N. : x has to be known in advance

C++code 3.2.43: Polynomial evaluation

```
class PolyEval {
  private:
  // evaluation point and various internal data describing the
     polynomials
  public:
    // Idle Constructor
    PolyEval();
    // Add another data point and update internal information
    void addPoint(t,y);
    // Evaluation of current interpolating polynomial at x
    Eigen::VectorXd operator () (const Eigen::VectorXd &x) const;
};
```

"Update-friendly" Newton basis

$$N_0(t) \equiv 1\ , \quad N_k(t) = \prod_{\ell=0}^{k-1}(t-t_\ell) \in \mathcal{P}_k$$

↳ leading coefficient $= 1$

▷ $N_k(t_j) = 0\ , \quad j=0,\dots,k-1$

$$p = \sum_{j=0}^{n} a_j N_j\ , \quad p \triangleq \text{Lagrange interpolant}$$

I.C. ⇒

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

triangular LSE

Structured way to perform forward elimination:

$a_{k,\ell} \triangleq$ leading coefficient of $p_{k,\ell}$

$$a_j = a_{0,j} \qquad \left[\, p_{0,\ell} = \sum_{j=0}^{\ell} a_j N_j \ ! \,\right]$$

A.N. recursion:

$$\begin{aligned} p_{k,k}(x) &\equiv y_k \quad \text{("constant polynomial")}\ , \quad k=0,\dots,n\ , \\ p_{k,\ell}(x) &= \frac{(x-t_k)p_{k+1,\ell}(x) - (x-t_\ell)p_{k,\ell-1}(x)}{t_\ell - t_k} \\ &= p_{k+1,\ell}(x) + \frac{x-t_\ell}{t_\ell - t_k}\, p_{k,\ell-1}(x)\ , \quad 0 \le k \le \ell \le n\ , \end{aligned} \tag{3.2.30}$$

$$a_{k,\ell} = \frac{1}{t_\ell - t_k}\left(1 \cdot a_{k+1,\ell} - 1 \cdot a_{k,\ell-1}\right)$$

$$a_{k,k} = y_k$$

Divided difference recursion

MATLAB-code 3.2.51: Divided differences, recursive implementation,

```
function y = divdiff(t,y)
n = length(y)-1;
if (n > 0)
  y(1:n) = divdiff(t(1:n),y(1:n));
  for j=0:n-1
    y(n+1) = (y(n+1)-y(j+1))/(t(n+1)-t(j+1));
  end
end
```

← "diagonal traversal"

Algorithm behind addPoint()

⑨ Implementation of ( ) : Horner scheme

```
function p = evaldivdiff(t,y,x)
dd=divdiff(t,y); % Compute divided differences, see Code 3.2.51
n = length(y)-1;
p=dd(n+1);
for j=n:-1:1, p = (x-t(j)).*p+dd(j); end
```

: Effort $O(n)$

$$p(x) = ((a_n(x-t_n)+a_{n-1})(x-t_{n-1})+a_{n-2})+\dots + a_0$$

3.2.4 : Polynomial Interpolation : Sensitivity

Given : Fixed nodes $t_0,\dots,t_n$ → set $\mathcal{T}$

Input : $y_0,\dots,y_n$ [might be perturbed : measured]

Output : $p \hat{=}$ Lagrange interpolant

Needed : norms on data space $\mathbb{R}^{n+1}$ → max norm. $\|y\|_\infty = \max_\ell |y_\ell|$

" result space $\mathcal{P}_n$ → max norm $\|p\|_\infty = \max_{t_0 \le x \le t_n} |p(x)|$

$$\|I_\mathcal{T}(y)\|_\infty = \left\|\sum_{j=0}^{n} y_j L_j\right\|_\infty$$
$$\le \max_x \sum_{j=0}^{n} |y_j|\,|L_j(x)|$$
$$\le \|y\|_\infty \left\|\sum_{j=0}^{n} |L_j|\right\|_\infty , \quad y \neq 0$$
$$\Rightarrow \quad \|I_\mathcal{T}\|_{\infty\to\infty} \le \left\|\sum_{j=0}^{n} |L_j|\right\|_\infty$$

Lemma 3.2.67. Absolute conditioning of polynomial interpolation

Given a mesh $\mathcal{T} \subset \mathbb{R}$ with generalized Lagrange polynomials L_i, $i = 0,\dots,n$, and fixed $I \subset \mathbb{R}$, the norm of the interpolation operator satisfies

$$\left[\ \|I_\mathcal{T}\|_{\infty\to\infty} := \sup_{\mathbf{y}\in\mathbb{R}^{n+1}\setminus\{0\}} \frac{\|I_\mathcal{T}(\mathbf{y})\|_{L^\infty(I)}}{\|\mathbf{y}\|_\infty} = \left\|\sum_{i=0}^{n} |L_i|\right\|_{L^\infty(I)}, \right] \tag{3.2.68}$$

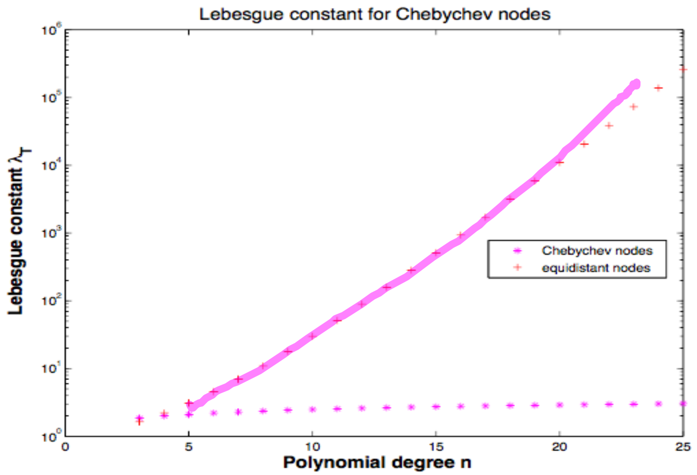
$$\|I_\mathcal{T}\|_{2\to2} := \sup_{\mathbf{y}\in\mathbb{R}^{n+1}\setminus\{0\}} \frac{\|I_\mathcal{T}(\mathbf{y})\|_{L^2(I)}}{\|\mathbf{y}\|_2} \le \left(\sum_{i=0}^{n} \|L_i\|^2_{L^2(I)}\right)^{\frac12}. \tag{3.2.69}$$

= Lebesgue constant (of $\mathcal{T}$) $\lambda_\mathcal{T}$

Note : Lagrange interpolation $I_\mathcal{T}$ is linear :

$L : X \to Y$ linear : $L(x+\delta x) = L(x) + L(\delta x)$

Amplification of perturbation measured by
$$\underbrace{\sup_{\delta x \in X\setminus\{0\}} \frac{\|L(\delta x)\|_Y}{\|\delta x\|_X}}_{\text{Operator norm, cf. matrix norm}} =: \|L\|_{X\to Y}$$



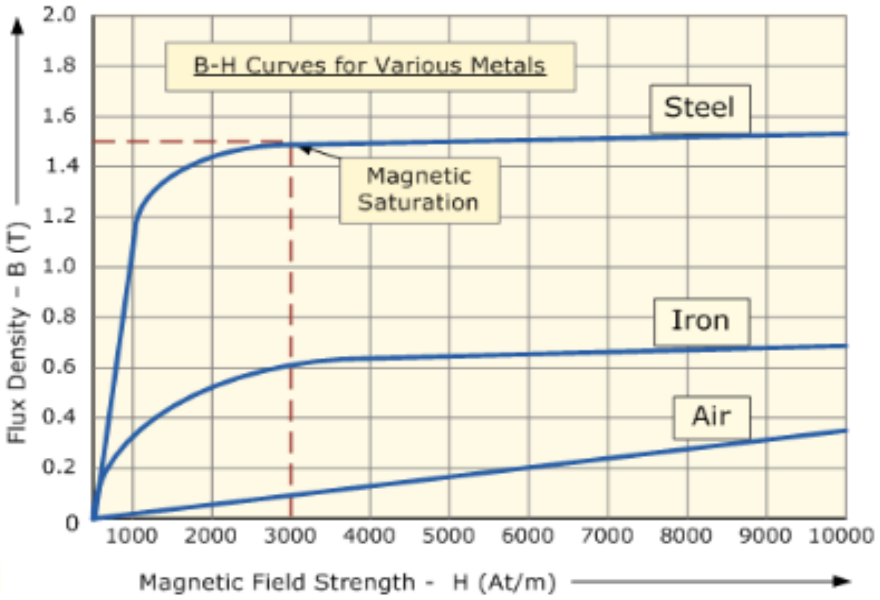
← $\lambda_\mathcal{T}$ for equidistant t_i in $[-1,1]$

$$\lambda_\mathcal{T} \ge C e^{n/2}$$

3.3. Shape preserving interpolation

Given: data points (t_i, y_i), $i = 0, \dots, n$; $t_0 < t_1 < \dots < t_n$

Seek: function $f : I \to \mathbb{R}$: $f(t_i) = y_i$



Magnetization curve

$H \longrightarrow B(H)$

- smooth
- monotonic
- concave

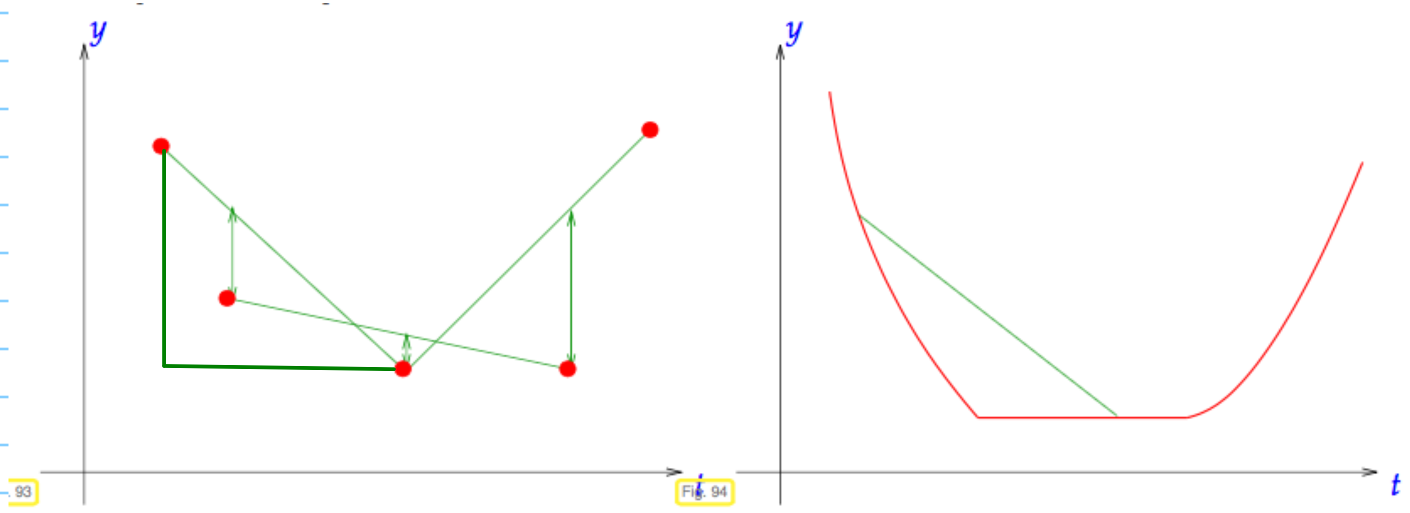
3.3.1. Shape properties of functions & data

- sign
- monotonicity

Definition 3.3.3. monotonic data

The data (t_i, y_i) are called monotonic when $y_i \geq y_{i-1}$ or $y_i \leq y_{i-1}$, $i = 1, \dots, n$.

- curvature



Convex data   Convex function

$$\Delta_j = \frac{y_j - y_{j-1}}{t_j - t_{j-1}} : \text{local slope}$$

f' increasing

Definition 3.3.4. Convex/concave data

The data $\{(t_i, y_i)\}_{i=0}^{n}$ are called convex (concave) if

$$\Delta_j \overset{(\geq)}{\leq} \Delta_{j+1}, \quad j = 1, \dots, n-1, \quad \Delta_j := \frac{y_j - y_{j-1}}{t_j - t_{j-1}}, \quad j = 1, \dots, n.$$

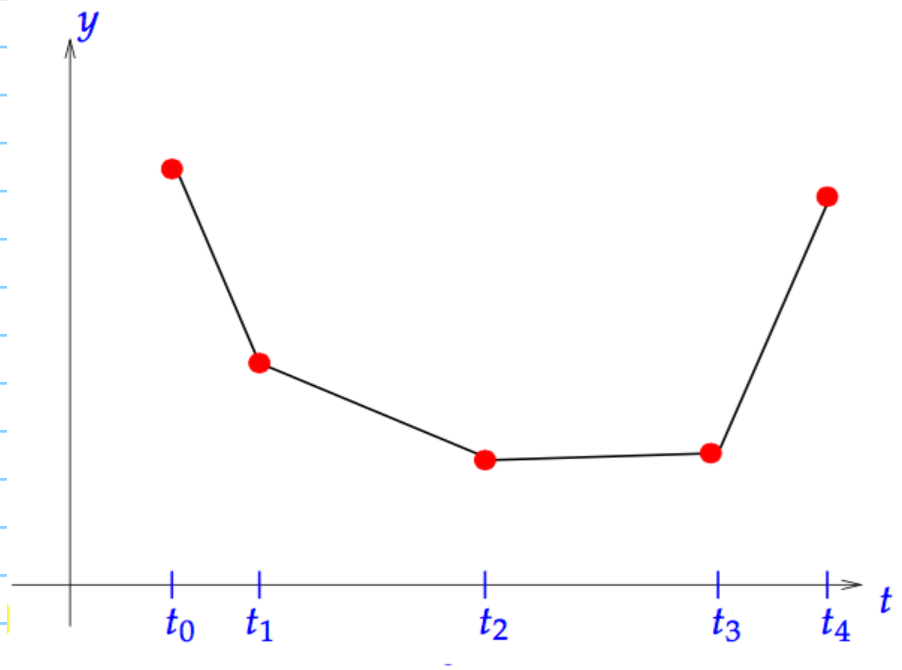
Shape preservation

(Local) Shape properties of data $\Longrightarrow$ shape property of interpolant

$\hookrightarrow$ holds on all $[t_{k-m}, t_k]$

⑪ Shape preservation for Lagrange IP ? NO WAY!

3.3.2. Piecewise linear interpolation



◁ Perfect local shape preservation!

Theorem 3.3.9. Local shape preservation by piecewise linear interpolation

Let $s \in C([t_0, t_n])$ be the piecewise linear interpolant of $(t_i, y_i) \in \mathbb{R}^2$, $i = 0, \dots, n$, for every subinterval $I = [t_j, t_k] \subset [t_0, t_n]$:

if $(t_i, y_i)|_I$ are positive/negative $\Rightarrow$ $s|_I$ is positive/negative,
if $(t_i, y_i)|_I$ are monotonic (increasing/decreasing) $\Rightarrow$ $s|_I$ is monotonic (increasing/decreasing),
if $(t_i, y_i)|_I$ are convex/concave $\Rightarrow$ $s|_I$ is convex/concave.

Remark: Locality of p.w. linear interpolation

$\hat{=}$ change of y_i affect interpolant only on $[t_{i-1}, t_{i+1}]$

Drawback: interpolant merely C^0 (kinks!)

3.4. (Piecewise) cubic Hermite interpolation

$\longrightarrow$ C^1-interpolants

3.4.1. Definition

Definition 3.4.1. Cubic Hermite polynomial interpolant

Given data points $(t_j, y_j) \in \mathbb{R} \times \mathbb{R}$, $j = 0, \dots, n$, with pairwise distinct ordered nodes t_j, and slopes $c_j \in \mathbb{R}$, the piecewise cubic Hermite interpolant $s : [t_0, t_n] \to \mathbb{R}$ is defined by the requirements

$$s_{|[t_{i-1}, t_i]} \in \mathcal{P}_3, \; i = 1, \dots, n \;, \quad \boxed{s(t_i) = y_i} \;, \quad \boxed{s'(t_i) = c_i} \;, \quad i = 0, \dots, n \,.$$

$\longrightarrow$ $s \in C^1$ continuity of s'

Express s locally through generalized cardinal basis functions

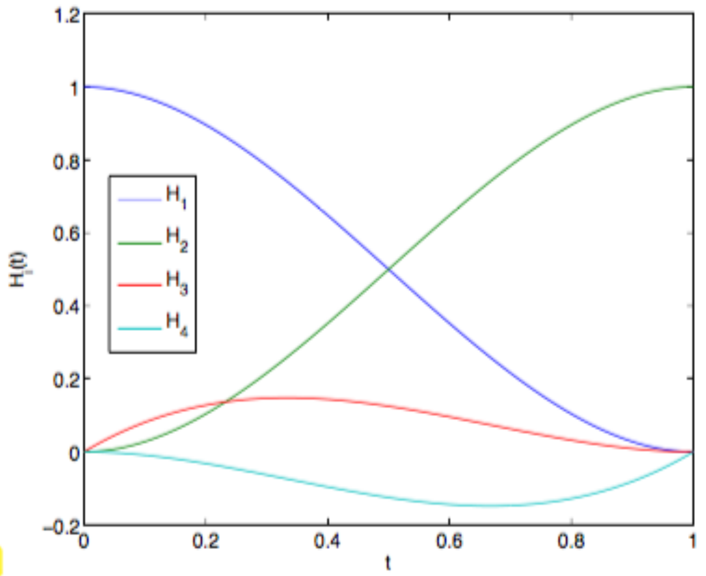
$$s(t) = y_{i-1} H_1(t) + y_i H_2(t) + c_{i-1} H_3(t) + c_i H_4(t)$$

for $t_{i-1} \leq t \leq t_i$

with

	$H(t_{i-1})$	$H(t_i)$	$H'(t_{i-1})$	$H'(t_i)$
H_1	1	0	0	0
H_2	0	1	0	0
H_3	0	0	1	0
H_4	0	0	0	1

$\longrightarrow$ 4x4 LSE for polynomial coefficients.



$$H_1(t) := \phi\left(\tfrac{t_i - t}{h_i}\right), \quad H_2(t) := \phi\left(\tfrac{t - t_{i-1}}{h_i}\right),$$
$$H_3(t) := -h_i\psi\left(\tfrac{t_i - t}{h_i}\right), \quad H_4(t) := h_i\psi\left(\tfrac{t - t_{i-1}}{h_i}\right),$$
$$h_i := t_i - t_{i-1} \;,$$
$$\phi(\tau) := 3\tau^2 - 2\tau^3 \;,$$
$$\psi(\tau) := \tau^3 - \tau^2 \,. \tag{3.4.5}$$

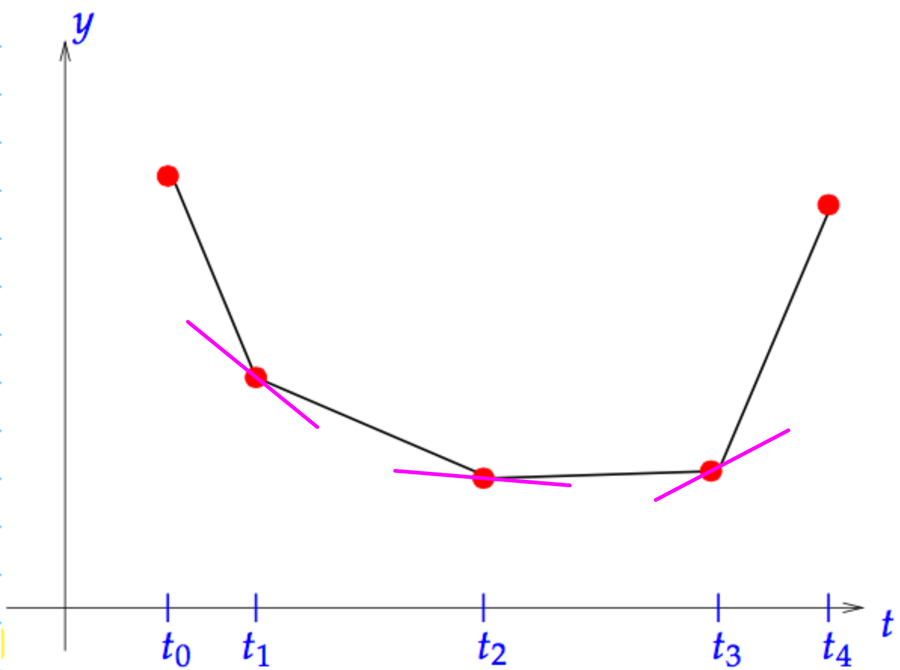
$$\tau := \frac{t - t_{i-1}}{t_i - t_{i-1}}$$

MATLAB-code 3.4.6: Local evaluation of cubic Hermite polynomial

```
function s=hermloceval(t,t1,t2,y1,y2,c1,c2)
% y1, y2: data values, c1, c2: slopes
h = t2-t1; t = (t-t1)/h;
a1 = y2-y1; a2 = a1-h*c1;
a3 = h*c2-a1-a2;
s = y1+(a1+(a2+a3*t).*(t-1)).*t;
```

Task: Find slopes from values : $c_i = c_i(y_0, \dots, y_n)$

Natural : local $c_i = c_i(y_{i-1}, y_i, y_{i+1})$



Idea :

c_i by local averaging of slopes

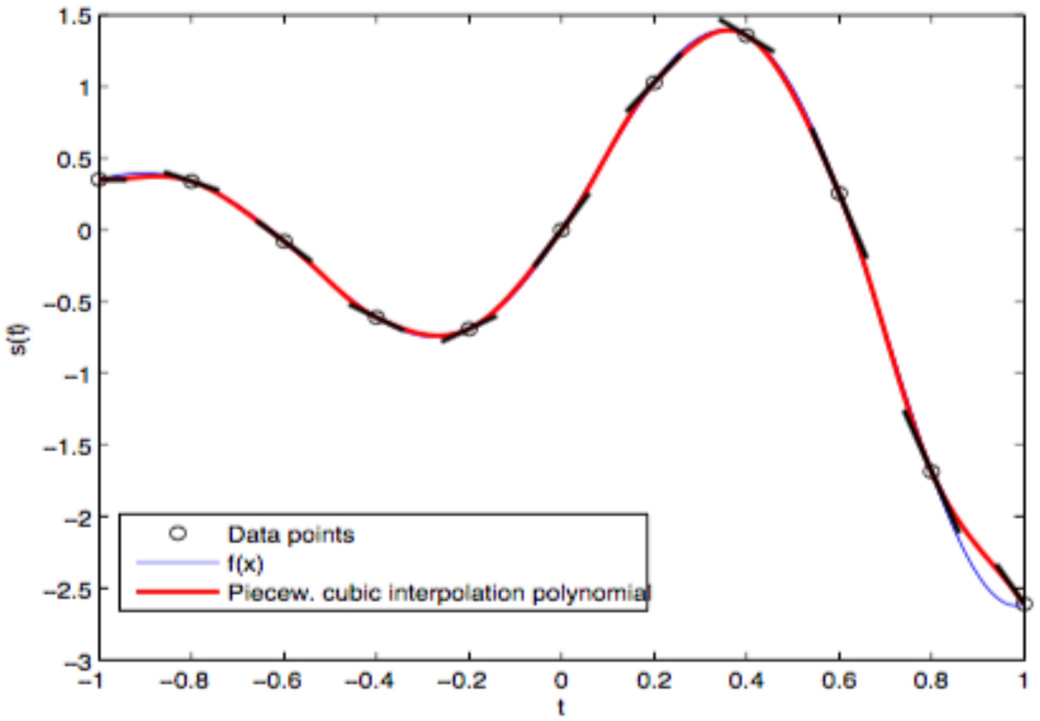
e.g. (generalized) arithmetic averaging :

$$c_i = \begin{cases} \Delta_1 & \text{, for } i = 0\,, \\ \Delta_n & \text{, for } i = n\,, \\ \underbrace{\frac{t_{i+1}-t_i}{t_{i+1}-t_{i-1}}\Delta_i + \frac{t_i - t_{i-1}}{t_{i+1}-t_{i-1}}\Delta_{i+1}} & \text{, if } 1 \le i < n\,. \end{cases} \quad , \quad \Delta_j := \frac{y_j - y_{j-1}}{t_j - t_{j-1}}, j = 1, \dots, n\,.$$

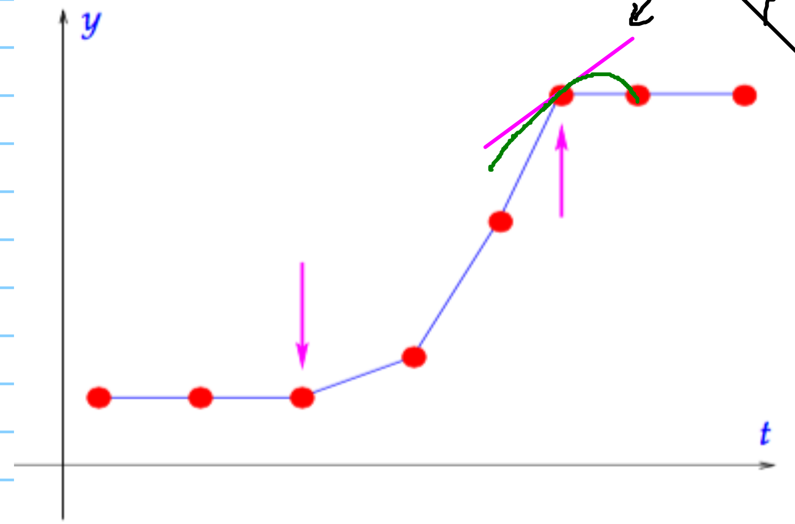
linear combination with non-negative coefficients adding up to 1.

$c_i = \frac{1}{2}(\Delta_i + \Delta_{i+1})$ for equispaced nodes !

Monotonicity preserving ? NO !



3.4.2. Local monotonicity preserving p.w. cubic Hermite intp.



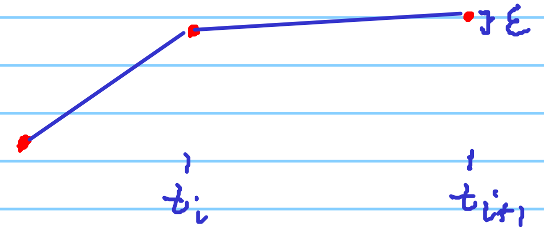
slope computed by arithmetic avg.

⇒ overshoot !

The only possible slope for the interpolant at the marked nodes is 0

▷ Use slope limiting

$$c_i = \begin{cases} 0 & \text{, if } \operatorname{sgn}(\Delta_i) \neq \operatorname{sgn}(\Delta_{i+1}) \,, \\ \text{some "average" of } \Delta_i, \Delta_{i+1} & \text{otherwise} \end{cases} \quad , \quad i = 1, \dots, n-1 \,.$$



⇒ If $\varepsilon \approx 0$, then $c_i \approx 0$

▷ $c_i \rightarrow 0$ as $\Delta_i \rightarrow 0$ or $\Delta_{i+1} \rightarrow 0$

Harmonic mean (weighted) : $c_i = \dfrac{w_i + w_{i+1}}{\frac{w_i}{\Delta_i} + \frac{w_{i+1}}{\Delta_{i+1}}}$, $w_i + w_{i+1} = 1$

$\operatorname{sgn}(\Delta_1) = \operatorname{sgn}(\Delta_2) \rightarrow$

$$c_i = \begin{cases} \Delta_1 & \text{, if } i = 0 \,, \\ \dfrac{3(h_{i+1}+h_i)}{\frac{2h_{i+1}+h_i}{\Delta_i} + \frac{2h_i + h_{i+1}}{\Delta_{i+1}}} & \text{, for } i \in \{1, \dots, n-1\} \,, \\ \Delta_n & \text{, if } i = n \,, \end{cases} \quad h_i := t_i - t_{i-1} \,. \tag{3.4.14}$$

otherwise use limiting

▷ C^1 interpolation scheme : MATLAB pchip

Theorem 3.4.18. Monotonicity preservation of limited cubic Hermite interpolation

The cubic Hermite interpolation polynomial with slopes as in Eq. (3.4.14) provides a local monotonicity-preserving C^1-interpolant.

Note: A non-linear (local) interpolation scheme

3.5. Splines

↳ Piecewise polynomial C^k-interpolants, $k \geq 2$ ($\to$ perceived as smooth)

Definition 3.5.1. Spline space → [21, Def. 8.1]

Given an interval $I := [a,b] \subset \mathbb{R}$ and a knot set/mesh $\mathcal{M} := \{a = t_0 < t_1 < \ldots < t_{n-1} < t_n = b\}$, the vector space $\mathcal{S}_{d,\mathcal{M}}$ of the spline functions of degree d (or order $d+1$) is defined by

$$\mathcal{S}_{d,\mathcal{M}} := \{s \in C^{d-1}(I) \colon s_j := s_{|[t_{j-1},t_j]} \in \mathcal{P}_d \; \forall j = 1,\ldots,n\} .$$

$d-1$-times continuously differentiable

locally polynomial of degree d

- $d = 0$: p.w. constant, discontinuous ("step function")
- $d = 1$: "polygon", $\mathcal{M}$-p.w. linear

Dimension of $\mathcal{S}_{d,\mathcal{M}}$: [counting argument]

$$\underbrace{n \cdot \dim \mathcal{P}_d}_{\text{dim. of } \mathcal{M}\text{-pw } \mathcal{P}_d} - \underbrace{(n-1) \cdot d}_{\#\text{ continuity constraint}} = \boxed{n + d = \dim \mathcal{S}_{\mathcal{M},d}}$$

Remark : $s \in \mathcal{S}_{d,\mathcal{M}} \Rightarrow s' \in \mathcal{S}_{d-1,\mathcal{M}}$

$\Rightarrow \int s \, dt \in \mathcal{S}_{d+1,\mathcal{M}}$

3.5.1 Cubic spline interpolation ($d = 3$)

Given : Knot set $\mathcal{M} = \{t_i\}_{i=0}^{n+1}$ + values $y_0, \ldots, y_n$

Find $s \in \mathcal{S}_{3,\mathcal{M}}$: $\underbrace{s(t_i) = y_i}_{n+1 \text{ conditions}}$ $\leftrightarrow$ $\dim \mathcal{S}_{3,\mathcal{M}} = n+3$

$\triangleright$ 2 degrees of freedom (d.o.f.) remain

Linear intp. $\Rightarrow$ LSE for expansion coefficient

Use same representation as for pchip !

(3.4.4) ▶

$$s_{|[t_{j-1},t_j]}(t) = \underset{y_{j-1}}{s(t_{j-1})} \cdot (1 - 3\tau^2 + 2\tau^3) + \underset{y_j}{s(t_j)} \cdot (3\tau^2 - 2\tau^3) + h_j \underset{c_{j-1}}{s'(t_{j-1})} \cdot (\tau - 2\tau^2 + \tau^3) + h_j \underset{c_j}{s'(t_j)} \cdot (-\tau^2 + \tau^3) , \tag{3.5.5}$$

C^1 already by construction $\Leftarrow$

with $h_j := t_j - t_{j-1}$, $\tau := (t - t_{j-1})/h_j$.

↑ unknown slopes c_j, $j = 0, \ldots, n$

Equations from $\underbrace{C^2\text{-continuity at internal knots}}$ t_j, $j = 1, \ldots, n-1$

$$s''_{|[t_{i-1},t_i]}(t_i) = s''_{|[t_i,t_{i+1}]}(t_i) \;, \quad i = 1, \ldots, n-1$$

$$s''_{|[t_{j-1},t_j]}(t_{j-1}) = -6 \cdot \underset{y_{j-1}}{s(t_{j-1})} h_j^{-2} + 6 \cdot \underset{y_j}{s(t_j)} h_j^{-2} - 4 \cdot h_j^{-1} \underset{c_{j-1}}{s'(t_{j-1})} - 2 \cdot h_j^{-1} \underset{c_j}{s'(t_j)} ,$$

$$s''_{|[t_{j-1},t_j]}(t_j) = 6 \cdot \underset{y_{j-1}}{s(t_{j-1})} h_j^{-2} + -6 \cdot \underset{y_j}{s(t_j)} h_j^{-2} + 2 \cdot h_j^{-1} \underset{c_{j-1}}{s'(t_{j-1})} + 4 \cdot h_j^{-1} \underset{c_j}{s'(t_j)} .$$

(15) Equate both sides: $j = 1, \dots, n-1$

$$\frac{1}{h_j}c_{j-1} + \left(\frac{2}{h_j}+\frac{2}{h_{j+1}}\right)c_j + \frac{1}{h_{j+1}}c_{j+1} = 3\left(\frac{y_j-y_{j-1}}{h_j^2}+\frac{y_{j+1}-y_j}{h_{j+1}^2}\right),$$

$$n-1\left\{\ \overbrace{\begin{bmatrix} b_0 & a_1 & b_1 & 0 & \cdots & \cdots & 0 \\ 0 & b_1 & a_2 & b_2 & & & \\ & 0 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & \ddots & \ddots & \ddots & \\ & & & \ddots & a_{n-2} & b_{n-2} & 0 \\ 0 & \cdots & \cdots & 0 & b_{n-2} & a_{n-1} & b_{n-1} \end{bmatrix}}^{n+1}\right. \begin{bmatrix} c_0 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} 3\left(\frac{y_1-y_0}{h_1^2}+\frac{y_2-y_1}{h_2^2}\right) \\ \vdots \\ 3\left(\frac{y_{n-1}-y_{n-2}}{h_{n-1}^2}+\frac{y_n-y_{n-1}}{h_n^2}\right) \end{bmatrix}.$$

with

$$b_i := \frac{1}{h_{i+1}}, \quad i = 0,1,\dots,n-1,$$
$$a_i := \frac{2}{h_i}+\frac{2}{h_{i+1}}, \quad i = 0,1,\dots,n-1.$$
$$[\ b_i, a_i > 0\ ,\ a_i = 2(b_i + b_{i-1})\ ]$$

Fix missing constraint:

① Complete C.S.I.: prescribe $s'(t_0) = c_0$, $s'(t_n) = c_n$

[→ "drop" first & last column of A]

② Natural C.S.I: requires $s''(t_0) = s''(t_n) = 0$

→ two extra equations

→ $\frac{2}{h_1}c_0 + \frac{1}{h_1}c_1 = 3\frac{y_1-y_0}{h_1^2}$, $\frac{1}{h_n}c_{n-1} + \frac{2}{h_n}c_n = 3\frac{y_n-y_{n-1}}{h_n^2}$.

③ Periodic C.S.I: $s'(t_0) = s'(t_n)$, $s''(t_0) = s''(t_n)$: $c_0 = c_n$ + 1 equ.

→ $n \times n$ LSE

3.5.2. Properties of CSI

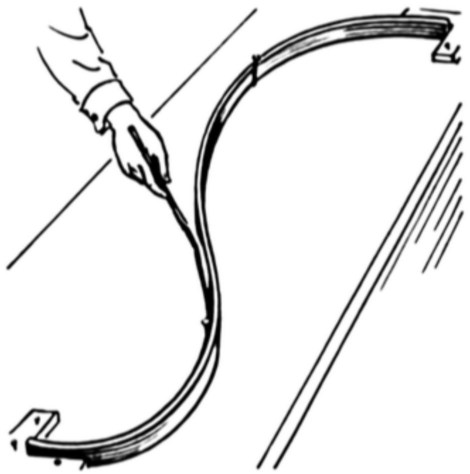
(i) Minimal "bending energy"

$$E_{bd}(f) = \int_a^b |f''(\tau)|^2\, d\tau$$

Theorem 3.5.14. Optimality of natural cubic spline interpolant

The natural cubic spline interpolant s minimizes the elastic curvature energy among all interpolating functions in $C^2([a,b])$, that is

$$E_{bd}(s) \le E_{bd}(f) \quad \forall f \in C^2([a,b]),\ f(t_i) = y_i,\ i = 0,\dots,n.$$



Proof: $d := f - s \in C^2$
$d(t_j) = 0$

$$\int_a^b d''(\tau)s''(\tau)\,d\tau = \sum_{j=1}^{n}\int_{t_{j-1}}^{t_j} d''(\tau)s''(\tau)\,d\tau$$
$$= \sum_{j=1}^{n} -\int_{t_{j-1}}^{t_j} d'(\tau)s'''(\tau)\,d\tau + \underbrace{\cancel{[d's'']_{t_{j-1}}^{t_j}}}_{=0}$$

telescopic sum

16

$$= \sum_{j=1}^{n} \int_{t_{j-1}}^{t_j} d(\tau)\underbrace{s^{(iv)}(\tau)}_{=0}d\tau - \left[\underbrace{d}_{=0} s^{(iii)}\right]_{t_{j-1}}^{t_j} = 0$$

$$\int_a^b |f''|^2 - |s''|^2 d\tau = \int_a^b (f+s-2s)'' d'' d\tau = \int_a^b |d''|^2 d\tau \geq 0 \qquad \square$$

(ii) Monotonicity preservation :

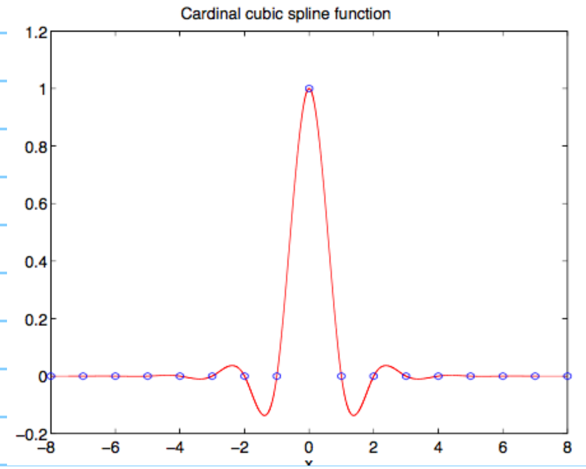
NO, because C.S.I. is a linear I.S.

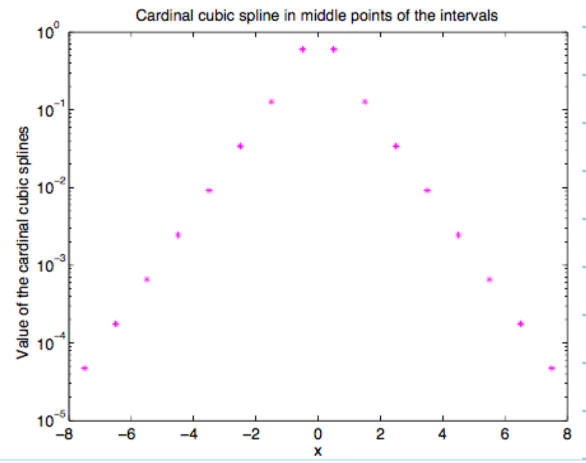
Theorem 3.4.17. Property of linear, monotonicity preserving interpolation into C^1

If, for fixed node set $\{t_j\}_{j=0}^n$, an interpolation scheme $\mathsf{I} : \mathbb{R}^{n+1} \to C^1(I)$ is linear as a mapping from data values to continuous functions on the interval covered by the nodes ($\to$ Def. 3.1.15), and monotonicity preserving, then $\mathsf{I}(\mathbf{y})'(t_j) = 0$ for all $\mathbf{y} \in \mathbb{R}^{n+1}$.

(iii) Locality : NO

C.S.I. linear $\Rightarrow$ Gauge locality by looking at cardinal interpolant $I_S e_j$





C.S.I. is "almost local" : cardinal interpolants decay exponentially