7. Interpolation

7.1 Allgemeine Problemstellung

Interpolation ist die Kunst, zwischen den Zeilen einer Tabelle zu lesen (Rutishauser).

Von $f : \mathbb{R} \to \mathbb{R}$ seien Funktionswerte $(x_j, f(x_j)), \quad j = 0, 1, \ldots, n,$ bekannt. Diese heißen **Stützstellen**, die x_j heißen **Knoten** .

Gesucht ist eine Funktion $p \in \mathrm{Int}_n$, die **interpoliert**, d.h. für die gilt

$$p(x_j) \;=\; f(x_j), \qquad j = 0, 1, \ldots, n\,. \tag{7.1.1}$$

Int_n ist dabei eine vorgegebene Funktionsklasse, z.B. *Polynome* von einem bestimmten Höchstgrad, *rationale Funktionen*, oder *Spline-Funktionen* (stückweise aus Polynomen zusammengesetzte Funktionen).

Häufig ist Int_n ein $(n+1)$-dimensionaler Vektorraum.

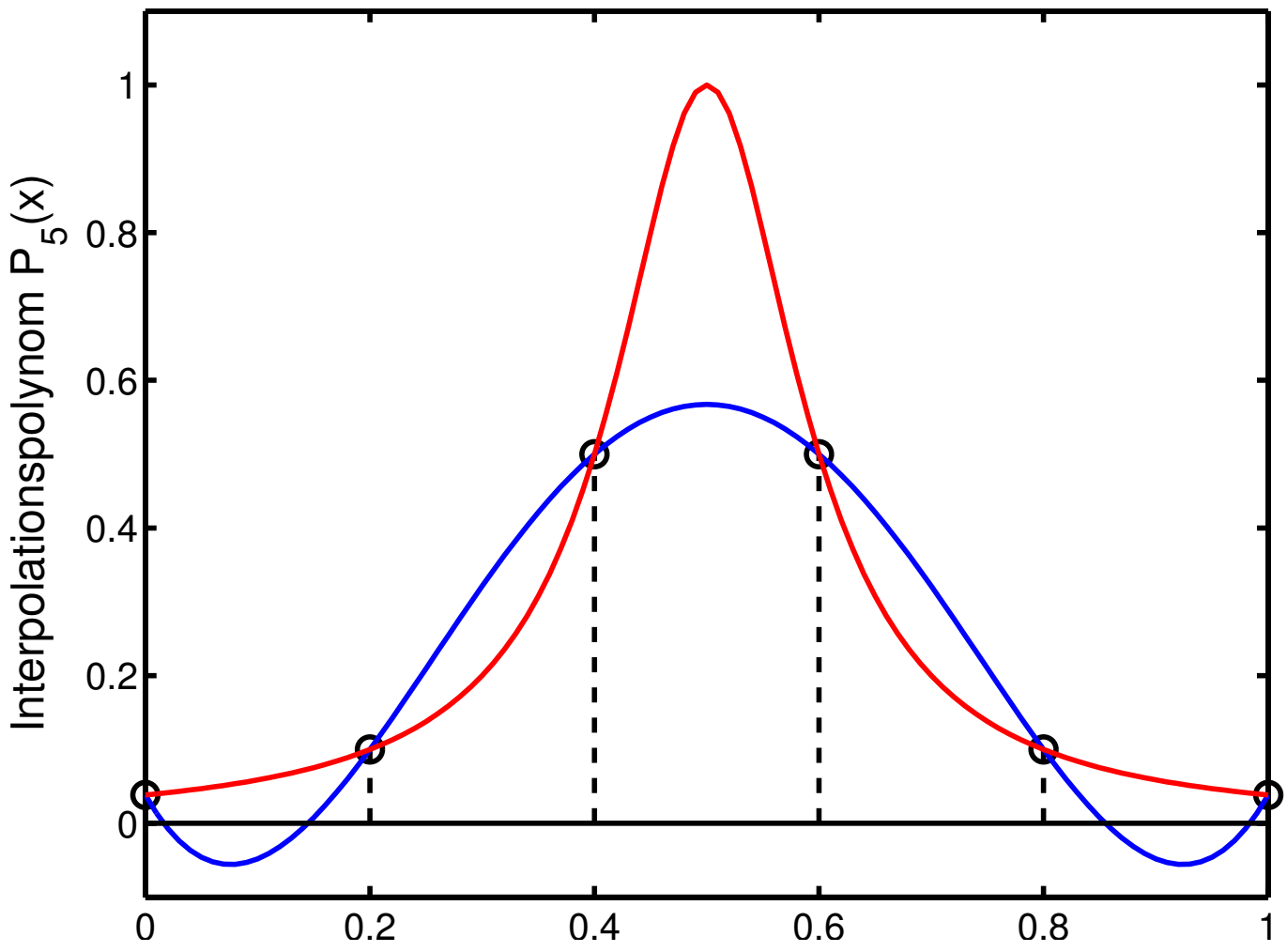
Will man nun den Wert $f(x)$ für ein $x \neq x_j$ approximieren, so wertet man statt dessen $p(x)$ aus.

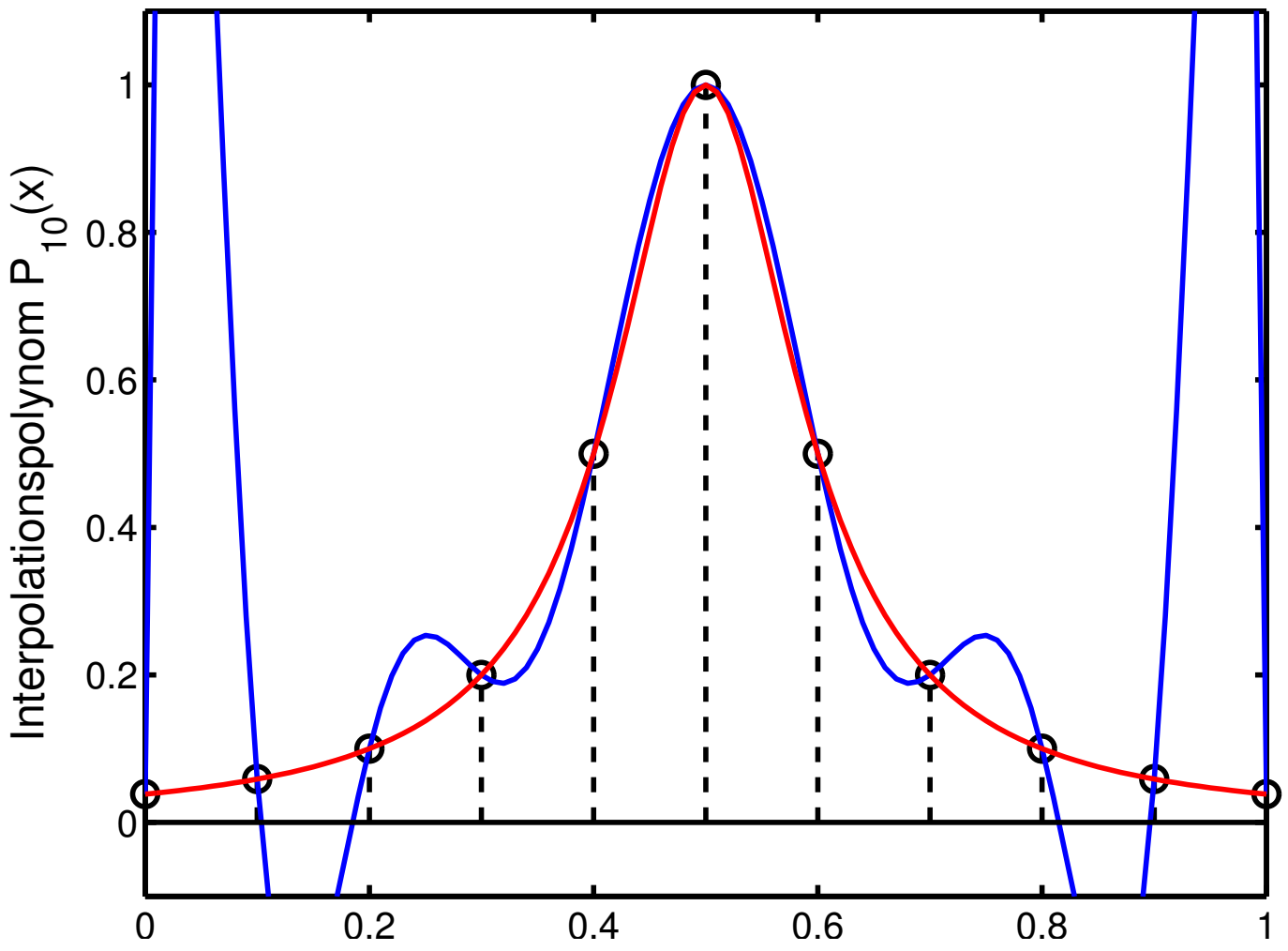
Je nach Wahl von Int_n, der Knoten x_j und der Lage von x (bezogen auf die Knoten) wird man unterschiedliche Approximationsgenauigkeiten zu erwarten haben.

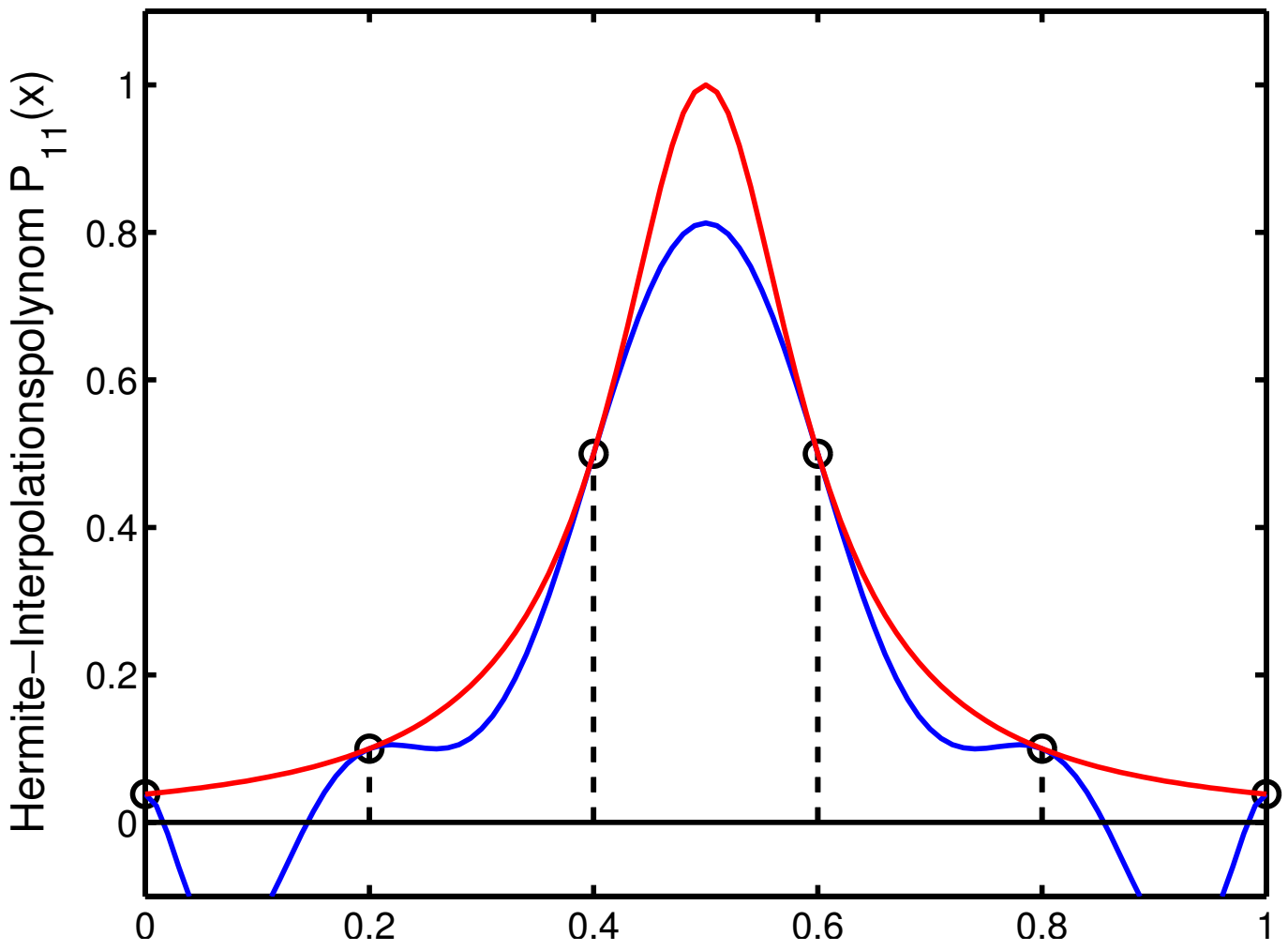
Hermite–Interpolation:

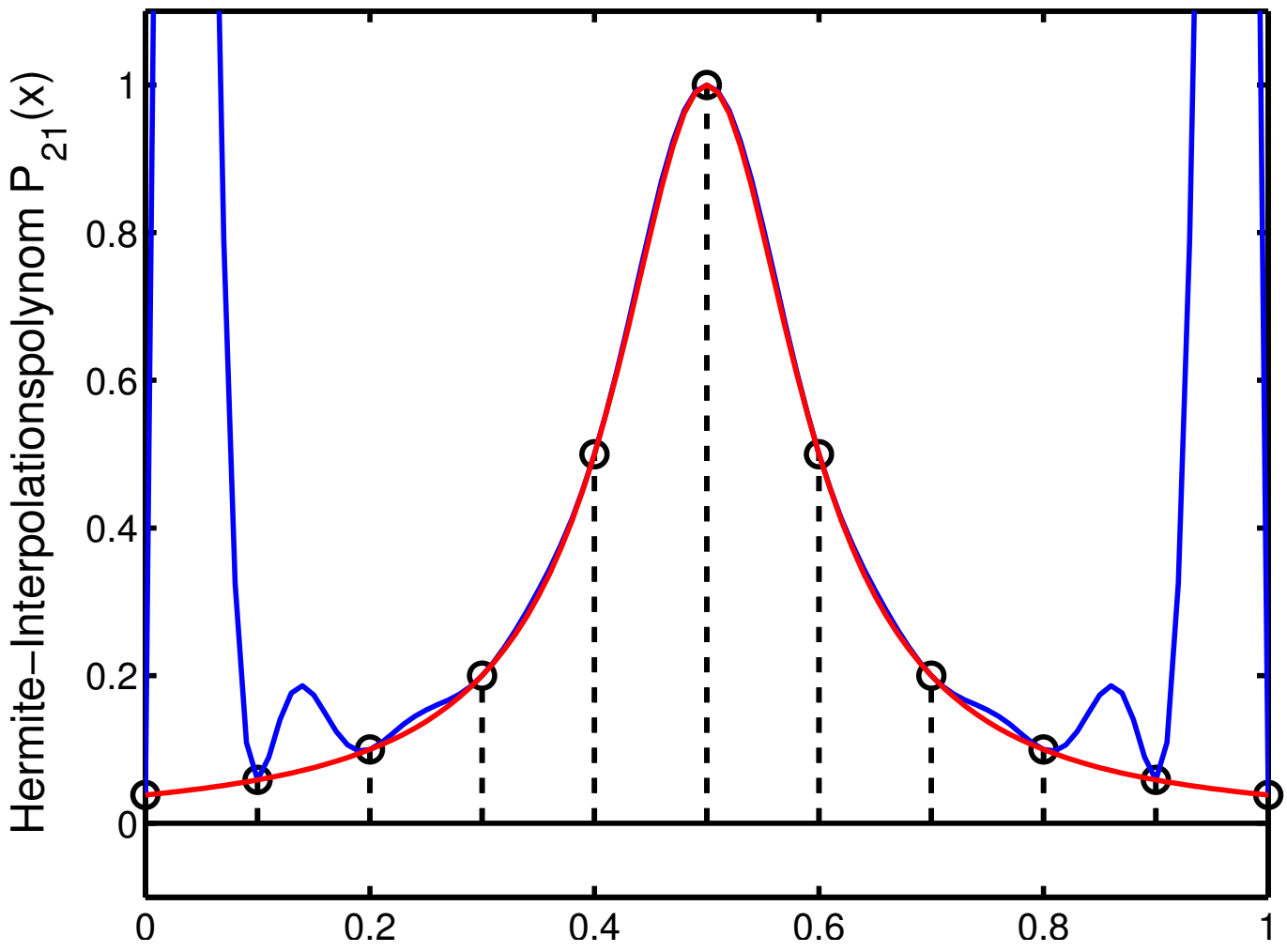
Kennt man neben den Funktionswerte $f(x_j)$ auch die Ableitungswerte $f'(x_j)$, so verlangt man, dass auch diese durch p interpoliert werden, dass also gilt:

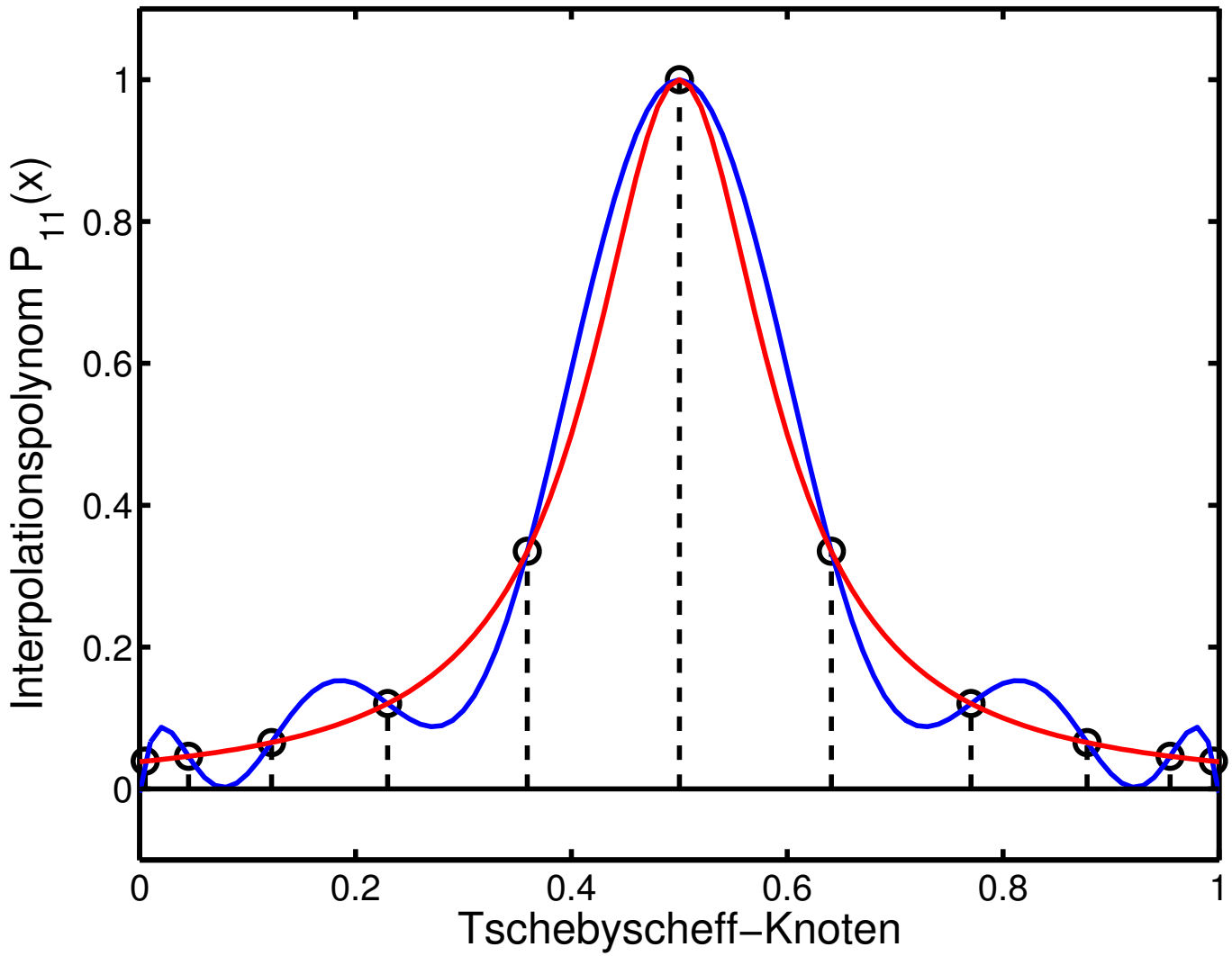
$$p(x_j) = f(x_j), \quad p'(x_j) = f'(x_j), \quad j = 0, 1, \ldots, n\,. \qquad (7.1.2)$$


Interpolationspolynom $P_5(x)$
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1


Interpolationspolynom $P_{10}(x)$
0
0.2
0.4
0.6
0.8
1
0
0.2
0.4
0.6
0.8
1


Hermite–Interpolationspolynom P11(x)
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1


Hermite–Interpolationspolynom P21(x)
0
0.2
0.4
0.6
0.8
1
0
0.2
0.4
0.6
0.8
1


Interpolationspolynom $P_{11}(x)$
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
Tschebyscheff−Knoten

7.2 Polynom–Interpolation

Man bestimme zu vorgegebenen Stützstellen ein Interpolationspolynom $p_n(x) = \sum_{k=0}^{n} a_k\, x^k$, mit

$$\sum_{k=0}^{n} a_k\, x_j^k \;=\; y_j, \qquad j = 0, 1, \ldots, n\,. \tag{7.2.1}$$

In Matrixschreibweise lautet (7.2.1)

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} \;=\; \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}. \tag{7.2.2}$$

Die Koeffizientenmatrix $V(x_0, \ldots, x_n) \in \mathbb{R}^{(n+1,n+1)}$ dieses linearen Gleichungssystems heißt **Vandermonde–Matrix**.

Satz (7.2.3): $\det V(x_0, \ldots, x_n) = \prod_{0 \le i < j \le n} (x_j - x_i)$.

Folgerung: Sind die Knoten x_j paarweise verschiedenen, so existiert genau ein Polynom $p_n \in \Pi_n$, welches (7.1.3) erfüllt.

Beispiel (7.2.4)

x_k	0	1	2	3
y_k	0	2	0	6

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ 0 \\ 6 \end{pmatrix} .$$

$$\Rightarrow \quad a = (0, 8, -8, 2)^T, \quad p_3(x) = 2x^3 - 8x^2 + 8x$$

Bemerkung (7.2.5)

Die Berechnung von p_n mittels (7.2.2) ist i. Allg. nicht zu empfehlen (Aufwand + Stabilität).

A. Darstellung nach Lagrange

Die **Lagrange–Polynome** $L_k \in \Pi_n$, $k = 0, 1, \ldots, n$, werden folgendermaßen definiert:

$$L_k(x) := \prod_{i=0,\, i\neq k}^{n} \left(\frac{x - x_i}{x_k - x_i} \right)$$

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

$$\Rightarrow \quad L_k(x_j) = \delta_{k\,j}, \qquad p_n(x) = \sum_{k=0}^{n} y_k\, L_k(x)\,. \tag{7.2.6}$$

Beispiel (7.2.7)

x_k	0	1	2	3
y_k	0	2	0	6

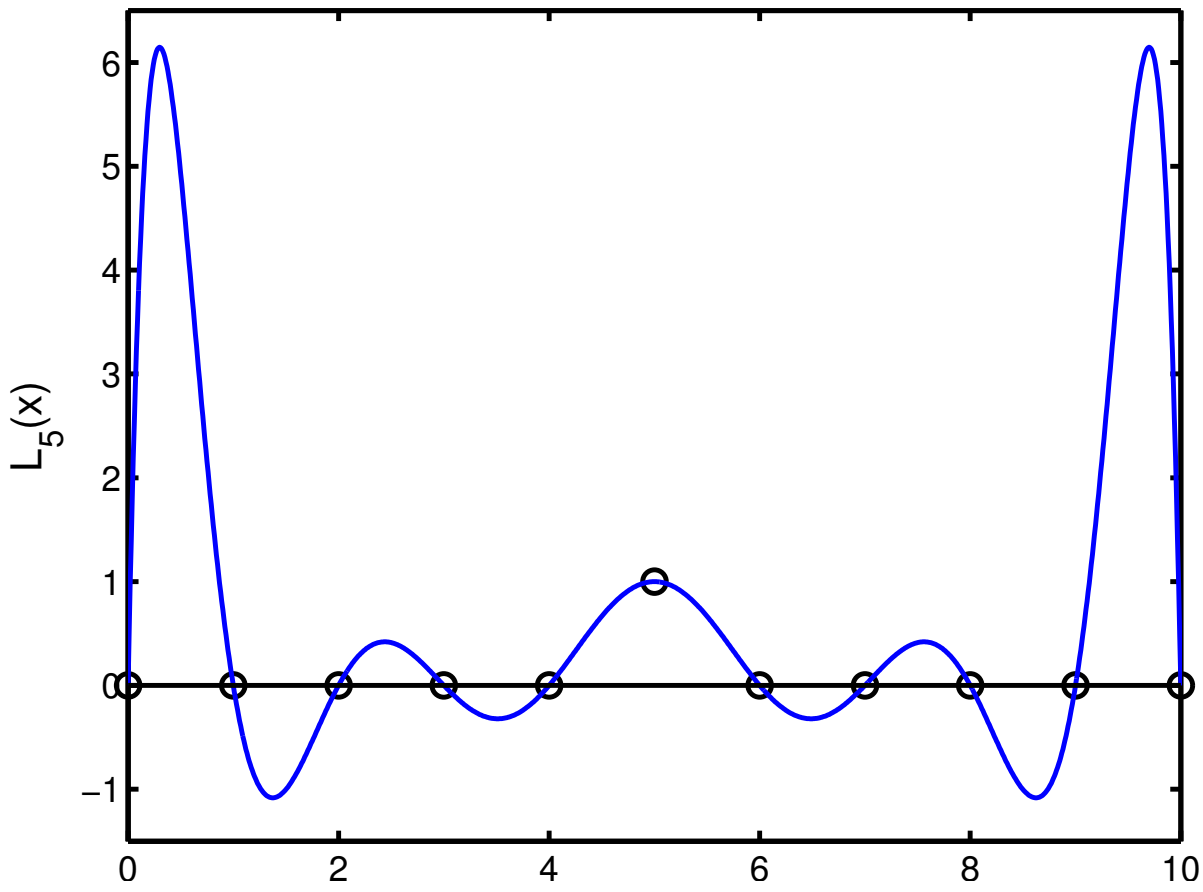
$$\begin{aligned}
\Rightarrow \quad p_3(x) &= 2\,\frac{(x-0)\,(x-2)\,(x-3)}{(1-0)\,(1-2)\,(1-3)} + 6\,\frac{(x-0)\,(x-1)\,(x-2)}{(3-0)\,(3-1)\,(3-2)} \\
&= x\,(x-2)\,[(x-3)+(x-1)] \\
&= x\,(x-2\,)\,(2x-4)\,.
\end{aligned}$$

Bemerkungen (7.2.8)

a) Die Auswertung von $p_n(x)$ mittels (7.2.6) ist ebenfalls i. Allg. zu aufwändig.

b) $L_k(x)$ gibt die (absoluten) Konditionszahlen des Interpolationsproblems bezüglich der Fehler in den y_k wieder.

c) Für große n oszillieren die L_k und werden insbesondere am Rand des Interpolationsbereichs betragsmäßig groß, so dass dort mit großer Verstärkung von Datenfehlern zu rechnen ist.



Lagrange–Polynom $L_5(x)$

Satz (7.2.9) (Interpolationsfehler)

Ist $f \in C^{n+1}[a,b]$ und p_n das Interpolationspolynom zu den Daten $a \leq x_0 < \ldots < x_n \leq b$ und $y_k = f(x_k),\ k = 0,1,\ldots,n$, so existiert zu $x \in [a,b]$ stets ein $\xi \in]a,b[$, so dass der Interpolationsfehler die folgende Darstellung besitzt:

$$f(x) \;=\; p_n(x) \;+\; \frac{f^{(n+1)}(\xi)}{(n+1)!}\,(x-x_0)\ldots(x-x_n)\,.$$

Bemerkungen (7.2.10)

a) Man beachte die Analogie zur Taylorschen Restgliedformel nach Lagrange!

b) $\omega(x) := (x-x_0)\ldots(x-x_n)$ heißt das **Knotenpolynom**. Die Aufgabe, $|\omega(x)|$ durch geschickte Wahl der x_j zu minimieren, führt auf die so genannten **Tschebyscheff–Knoten**.

Beispiel (7.2.11) $f(x) := \sin(\frac{\pi}{4}\, x)$ soll auf $[0,2]$ durch eine interpolierende Parabel approximiert werden. Knoten (äquidistant): $0, 1, 2$.

Eine Fehlerabschätzung nach (7.2.9) ergibt

$$\begin{aligned} |p_2(x) - f(x)| &\leq \frac{(\pi/4)^3 \cos((\pi/4)\xi)}{3!} |x(x-1)(x-2)| \\ &\leq \frac{\pi^3}{384} \| \omega \|_\infty, \quad \omega(x) := x(x-1)(x-2) \end{aligned}$$

Zur Berechnung von $\| \omega \|_\infty$ bestimmt man die lok. Extrema von ω auf $[0,2]$ und findet $\| \omega \|_\infty = \frac{2}{3\sqrt{3}}$.

$$\Rightarrow \quad |p_2(x) - f(x)| \leq \frac{2\,\pi^3}{1152\sqrt{3}} \approx 0.031$$

B. Darstellung nach Aitken, Neville

Idee: Rekursive Berechnung des Interpolationspolynoms $p_n(x)$

$p_{k\,j}(x) \in \Pi_j$: Interpolationspolynom zu den Stützstellen:

$$(x_{k-j},\, y_{k-j}),\ (x_{k-j+1},\, y_{k-j+1}),\ \ldots,\ (x_k,\, y_k)\,.$$

Satz (7.2.12) (Lemma von Aitken)

Es gilt die folgende Rekursion: $(j = 1, \ldots,\, k)$

$$\begin{aligned} p_{k\,0}(x) &= y_k, \qquad k = 0, \ldots,\, n, \\ p_{k\,j}(x) &= p_{k,\,j-1}(x) + \frac{x - x_k}{x_{k-j} - x_k} \left(p_{k-1,\,j-1}(x) - p_{k,\,j-1}(x) \right) \end{aligned}$$

Schema von Neville (7.2.13)

(zeilenweise Berechnung)

$$
\begin{array}{llllll}
p_{0\,0} & & & & & \\
p_{1\,0} & p_{1\,1} & & & & \\
p_{2\,0} & p_{2\,1} & p_{2\,2} & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
p_{n\,0} & p_{n\,1} & p_{n\,2} & \cdots & p_{n\,n}
\end{array}
\qquad
\begin{array}{llllll}
p_0 & & & & & \\
p_1 & p_0 & & & & \\
p_2 & p_1 & p_0 & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
p_n & p_{n-1} & p_{n-2} & \cdots & p_0
\end{array}
$$

Algorithmus von Aitken, Neville (7.2.14)

$p_0 := y_0;$
für $k = 1, 2, \ldots, n$
  $p_k := y_k; \quad z := x - x_k;$
  für $i = k-1,\ k-2,\ \ldots,\ 0$

$$p_i := p_{i+1} + \frac{z}{x_i - x_k}\,(p_i - p_{i+1});$$

  end i
end k

Beispiel (7.2.15)

Gesucht: $p_4(x)$ zu den Stützstellen $(x_k, \sin(x_k))$, $k = 0, 1, \ldots, 4$,
mit: $x_k := (50 + 5\,k) \cdot \pi/180$, und $x := 62 \cdot \pi/180 \approx 1.082104$.

x_k	$\sin(x_k)$				
.8726647	.7660444				
.9599311	.8191520	.8935027			
1.047198	.8660254	.8847748	.8830292		
1.134464	.9063078	.8821384	.8829293	.8829493	
1.221731	.9396926	.8862769	.8829661	.8829465	.8829476

Der Algorithmus von Aitken, Neville ist konkurrenzfähig, falls das Interpolationspolynom nur an einer, bzw. nur an wenigen Stellen berechnet werden soll.

C. Darstellung nach Newton

Idee: Explizite Darstellung von p_n bzgl. der **Newton–Basis** $\omega_0(x), \ldots, \omega_n(x)$, $P_n(x) = \sum_{k=0}^{n} d_k \, \omega_k(x)$, mit

$$\omega_k(x) \;:=\; \prod_{j=0}^{k-1} (x - x_j)\,, \quad k \;=\; 0, \ldots, n\,. \tag{7.2.16}$$

Dividierte Differenzen:

$[y_j, y_{j+1}, \ldots, y_k]$: höchster Koeffizienten (von x^{k-j}) des Interpolationspolynoms $p_{k,\,k-j}$ zu den Stützstellen (x_i, y_i), $i = j, \ldots, k$.

Damit folgt:

$$\begin{aligned} p_n(x) &= p_{nn}(x) \\ &= [y_0, \ldots, y_n]\ (x - x_0) \cdot \ldots \cdot (x - x_{n-1}) \;+\; Q(x) \\ &= [y_0, \ldots, y_n]\, \omega_n(x) + [y_0, \ldots, y_{n-1}]\, \omega_{n-1}(x) + \ldots \\ &= \sum_{k=0}^{n} [y_0, \ldots, y_k]\, \omega_k(x) \end{aligned}$$

Wie kann man nun die dividierten Differenzen berechnen?
Die Anwendung des Lemmas von Aitken liefert die Rekursion:

Satz (7.2.17) (Dividierte Differenzen)

$$\begin{aligned} [y_j] &= y_j \\ [y_j, \ldots, y_k] &= \frac{[y_{j+1}, \ldots, y_k] - [y_j, \ldots, y_{k-1}]}{x_k - x_j}, \quad 0 \le j < k \le n\,. \end{aligned}$$

$$\begin{array}{lllll} y_0 = [y_0] & & & & \\ y_1 = [y_1] & [y_0, y_1] & & & \\ y_2 = [y_2] & [y_1, y_2] & [y_0, y_1, y_2] & & \\ \quad \vdots & \quad \vdots & & & \\ y_n = [y_n] & [y_{n-1}, y_n] & \cdots & & [y_0, y_1, \ldots, y_n] \end{array}$$

Bei eindimensionaler Indizierung (Berechnung spaltenweise)

$$
\begin{array}{llllll}
d_0 & & & & & \\
d_1 & d_1 & & & & \\
d_2 & d_2 & d_2 & & & \\
\uparrow & \uparrow & \uparrow & \ddots & & \\
d_n & d_n & d_n & \ldots & & d_n
\end{array}
$$

ergibt sich der **Algorithmus (7.2.18)**

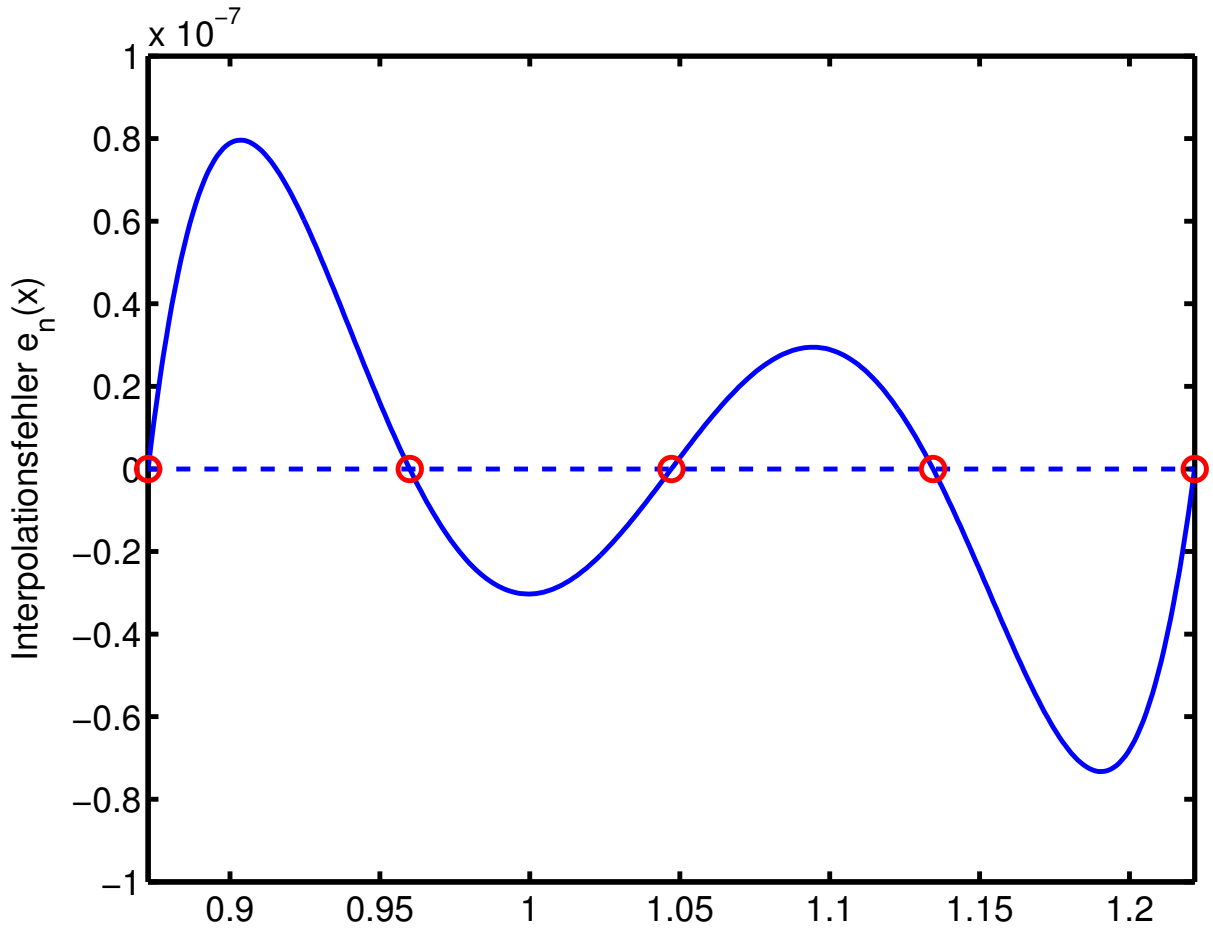
$d_j := y_j \quad (j = 0, 1, \ldots, n)$
für $k = 1, 2, \ldots, n$
  für $j = n, n-1, \ldots, k$
    $d_j := (d_j - d_{j-1})/(x_j - x_{j-k})$
  end j
end k

Die Auswertung von $p_n(x) = \sum_{k=0}^{n} d_k \, (x - x_0) \, \ldots \, (x - x_{k-1})$
erfolgt mit Hilfe des **Horner–Schemas (7.2.19)**:

$$
\begin{aligned}
& p \;:=\; d_n \quad (= [y_0, \ldots, y_n]) \\
& \text{für} \quad k = n-1, \ldots, 0 \\
& \qquad p \;:=\; (x - x_k) \cdot p + d_k \\
& \text{end } k
\end{aligned}
$$

Beispiel (7.2.20) **(sin-Beispiel, vgl. (7.2.15))**
Algorithmus (7.2.18) ergibt die dividierten Differenzen:

k	d_k
0	$7.660444e-01$
1	$6.085683e-01$
2	$-4.093162e-01$
3	$-8.946472e-02$
4	$3.603862e-02$



Interpolationsfehler $e_4(x) := \sin(x) - p_4(x)$

7.3 Spline-Interpolation

Man betrachtet folgende Approximation der Gesamtkrümmung

$$I[y] \;=\; \int_{x_0}^{x_n} (y''(x))^2 \,\mathrm{d}x \qquad (7.3.1)$$

Satz (7.3.2)

Es gibt genau eine Funktion $s(x), a \le x \le b$, die $I[y]$ unter allen interpolierenden C^2–Funktionen minimiert. s ist eine **kubische Spline–Funktion**, d.h.:

a) In jedem Teilintervall $[x_j, x_{j+1}]$ ist s ein Polynom aus Π_3.

b) s ist eine C^2–Funktion auf $[a, b]$.

c) $s(x_j) = y_j, \quad j = 0, 1, \ldots, n.$

d) $s''(x_0) = s''(x_n) = 0$ **(natürliche Randbedingungen)**.

Berechnung der Spline-Funktion (7.3.3) $(x \in [x_j, x_{j+1}])$

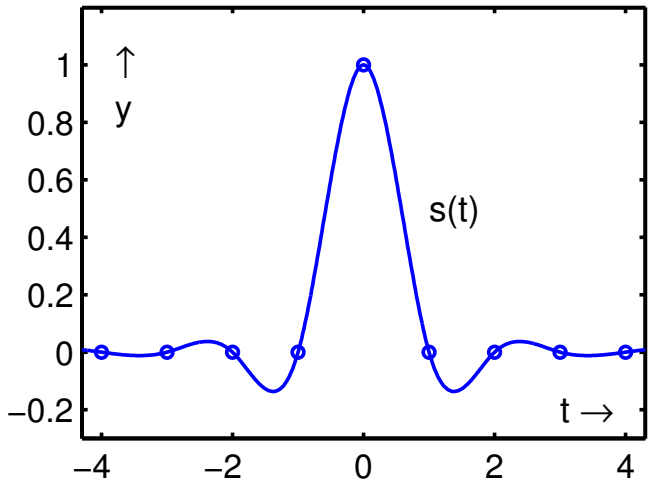
$$
\begin{aligned}
s(x) &= y_j + b_j\,(x - x_j) + c_j\,(x - x_j)^2 + d_j\,(x - x_j)^3 \\
b_j &= \frac{y_{j+1} - y_j}{h_j} - \frac{2\,c_j + c_{j+1}}{3}\,h_j, \qquad h_j := x_{j+1} - x_j \\
d_j &= \frac{c_{j+1} - c_j}{3\,h_j}\,.
\end{aligned}
$$

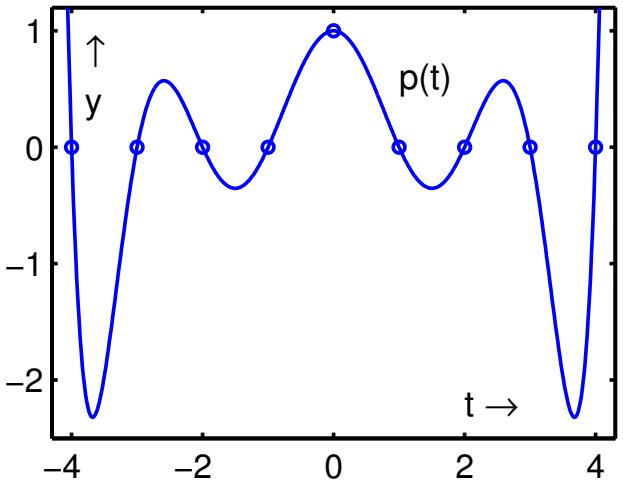
$$
\begin{pmatrix}
2(h_0+h_1) & h_1 & & & 0 \\
h_1 & 2(h_1+h_2) & h_2 & & \\
 & \ddots & \ddots & \ddots & \\
 & & & h_{n-2} & \\
0 & & h_{n-2} & 2(h_{n-2}+h_{n-1})
\end{pmatrix}
\begin{pmatrix} c_1 \\ \vdots \\ \vdots \\ \vdots \\ c_{n-1} \end{pmatrix}
=
\begin{pmatrix} r_1 \\ \vdots \\ \vdots \\ \vdots \\ r_{n-1} \end{pmatrix}
$$

$$
\begin{aligned}
r_j &:= 3\left(\frac{y_{j+1} - y_j}{h_j} - \frac{y_j - y_{j-1}}{h_{j-1}}\right), \quad j = 1, \ldots, n-1\,. \\
c_0 &:= c_n := 0\,.
\end{aligned}
$$

Beispiel (7.3.4)

t_j	-4	-3	-2	-1	0	1	2	3	4
y_j	0	0	0	0	1	0	0	0	0





Bemerkungen (7.3.5)

a) Der interpolierende kubische Spline ist einfach zu berechnen. Aufwand ist vergleichbar mit dem Aufwand zur Interpolation mit Polynomen.

b) Splines neigen aufgrund ihrer Minimaleigenschaft weniger zu Oszillationen als Interpolationspolynome und sind auch bei einer größeren Anzahl von Stützpunkten brauchbar.

c) Datenfehler in y_j wirken sich nur lokal, d.h. in der Nähe von x_j aus. Bei größerem Abstand zu x_j tritt sogar Fehlerdämpfung auf!

d) Für eine C^4–Funktion f gilt die **Fehlerabschätzung**:

$$| f(x) - s_\Delta(x) | \;\leq\; L \cdot K \cdot \|\Delta\|^4, \quad a \leq x \leq b.$$

Dabei ist

$$\|\Delta\| := \max_j |x_{j+1} - x_j|, \quad K := \|\Delta\| / \min_j |x_{j+1} - x_j|, \quad L := \| f^{(4)} \|_\infty$$