

# Einführung in die Numerik (Potschka)

Robin Heinemann

16. Mai 2017

## Inhaltsverzeichnis

<b>0</b>	<b>Einführung</b>	<b>1</b>
<b>1</b>	<b>Fehleranalyse</b>	<b>3</b>
1.1	Zahldarstellung und Rundungsfehler . . . . .	3
1.2	Konditionierung numerischer Aufgaben . . . . .	6
1.2.1	Differentielle Fehleranalyse . . . . .	7
1.2.2	Arithmetische Grundoperationen . . . . .	9
1.3	Stabilität numerischer Algorithmen . . . . .	11
<b>2</b>	<b>Interpolation und Approximation</b>	<b>15</b>
2.1	Auswertung von Polynomen und deren Ableitungen . . . . .	20
2.2	Interpolation von Funktionen . . . . .	22
2.3	Richardsonsche Extrapolation zum Limes . . . . .	25
2.4	Spline-Interpolation . . . . .	27

## 0 Einführung

### Beispiel 0.1 Simulation einer Pendelbewegung

Modellannahmen:

- Masse  $m$  an Stange
- keine Reibung
- Stange: Gewicht 0, starr, Länge  $l$
- Auslenkung  $\phi$

**Erste Fehlerquelle:** Modellierungsfehler

Modellgleichungen:

$$F_T(\phi) = -m \cdot g \sin \phi$$

Konsistenzcheck:

$$\begin{aligned} F_T(0) &= 0 & (\text{Ruhelage}) \\ F_T\left(\frac{\pi}{2}\right) &= F_G = -mg \end{aligned}$$

Bewegungsgleichungen:

- Weg  $s(t)$
- $\frac{ds}{dt} =: v(t)$  Geschwindigkeit
- $\frac{dv}{dt} =: a(t)$  Beschleunigung

Beziehungen:

- Bogenlänge  $s(t) = l\phi(t)$
- 2. Newton'sches Gesetz ( $F = ma$ )

$$-mg \sin \phi(t) = m \frac{d}{dt} v(t) = m \frac{d^2}{dt^2} s(t) = ml \frac{d^2}{dt^2} \phi(t)$$

$\Rightarrow$  DGL 2. Ordnung

$$\frac{d^2}{dt^2} \phi(t) = -\frac{g}{l} \sin \phi(t) \quad t \geq 0$$

Für eindeutige Lösung braucht man zwei Anfangsbedingungen:

$$\phi(0) = \phi_0 \quad \frac{d}{dt} \phi(0) = u_0$$

Lösung bei kleiner Auslenkung: Linearisiere um  $\phi = 0$

$$\begin{aligned} \sin \phi &= \phi - \frac{1}{3!} \phi^3 + \dots \approx \phi \\ \Rightarrow \frac{d^2}{dt^2} \phi(t) &= -\frac{g}{l} \phi(t) \end{aligned}$$

Für  $u_0 = 0$  findet man mit dem Ansatz  $\phi(t) = A \cos(\omega t)$ :

$$-\omega^2 A \cos(\omega t) = -\frac{g}{l} A \cos(\omega t)$$

die Lösung:

$$\phi(t) = \phi_0 \cos\left(\sqrt{\frac{g}{l}} t\right)$$

Fehlerquelle: Abschneidefehler.

Numerische Lösung:

Setze  $u(t) := \frac{d}{dt} \phi(t)$

$$\frac{d}{dt} \begin{pmatrix} \phi \\ u \end{pmatrix} = \begin{pmatrix} u \\ -\frac{g}{l} \sin(\phi) \end{pmatrix}$$

Approximation mit Differenzenquotienten

$$\left( -\frac{g}{l} \sin \phi(t) \right) = \frac{d}{dt} \left( \frac{\phi}{u} \right) = \lim_{\Delta t \rightarrow 0} \frac{1}{\Delta t} \left( \frac{\phi(t + \Delta t) - \phi(t)}{u(t + \Delta t) - u(t)} \right) \approx \frac{1}{\Delta t} \left( \frac{\phi(t + \Delta t) - \phi(t)}{u(t + \Delta t) - u(t)} \right)$$

$\downarrow$   
 $> 0, \text{ klein}$

Fehlerquelle: Diskretisierungsfehler

Auf Gitter  $t_n = n\Delta t$  mit Werten  $\phi_n = \phi(n\Delta t)$ ,  $u_n = u(n\Delta t)$ :

$$\phi_{n+1} = \phi_n + \Delta t u_n, u_{n+1} = u_n - \Delta t \frac{g}{l} \phi_n$$

Kleinerer Diskretisierungsfehler mit zentralen Differenzen:

$$-\frac{g}{l} \sin \phi(t) = \frac{d^2}{dt^2} \phi(t) \approx \frac{\phi(t + \Delta t) - 2\phi(t) + \phi(t - \Delta t)}{\Delta t^2}$$

Rekursionsformel:

$$\phi_{n+1} = 2\phi_n - \phi_{n-1} - \Delta t^2 \frac{g}{l} \sin \phi_n, n \geq 1$$

mit  $\phi_1 = \phi_0 + \Delta t n_0$  (Expliziter Euler)

Letzte Fehlerquelle: Rundungsfehler

## 1 Fehleranalyse

### 1.1 Zahldarstellung und Rundungsfehler

Anforderung: Rechnen mit reellen Zahlen auf dem Computer.

Problem: Speicher endlich ( $\implies$  endliche Genauigkeit).

Lösung: Gleitkommazahlen, ein **Kompromiss** zwischen:

- Umfang darstellbarer Zahlen
- Genauigkeit
- Geschwindigkeit einfacher Rechenoperationen (+, -, ·, /)

Alternativen:

- Fixkommazahlen
- logarithmische Zahlen
- Rationalzahlen

**Definition 1.1** Eine (normalisierte) Gleitkommazahl zur Basis  $b \in \mathbb{N}, b \geq 2$ , ist eine Zahl  $x \in \mathbb{R}$  der Form

$$x = \pm m \cdot b^{\pm e}$$

mit der Mantisse  $m = m_1 b^{-1} + m_2 b^{-2} + \dots \in \mathbb{R}$  und dem Exponenten  $e = e_{s-1} b^{s-1} + \dots + e_0 b^0 \in \mathbb{N}$ , wobei  $m_i, e_i \in \{0, \dots, b-1\}$ . Für  $x \neq 0$  ist die Darstellung durch die Normierungsvorschrift  $m \neq 0$  eindeutig. Für  $x = 0$  setzt man  $m = 0$ .

**Beispiel 1.2 ( $b = 10$ )** •  $m_i$ :  $i$ -te Nachkommastelle der Mantisse

- $e$ : Verschiebt das Komma um  $e$  Stellen.

$$0.314 \times 10^1 = 3.14$$

$$0.123 \times 10^6 = 123\,000$$

Auf dem Rechner stehen nur endlich viele Stellen zur Verfügung:

$r$  Ziffern + 1 Vorzeichen für Mantisse  $m$

$s$  Ziffern + 1 Vorzeichen für Exponenten.

Für  $x = \pm[m_1 b^{-1} + \dots + m_r b^{-r}] \cdot b^{\pm[e_{s-1} b^{s-1} + \dots + e_0 b^0]}$  muss man also nur  $(\pm)[m_1 \dots m_r](\pm)[e_{s-1} \dots e_0]$  abspeichern. Wählt man  $b = 2$ , so gilt  $m_i, e_i \in \{0, 1\}$  und  $x$  kann mit  $2 + r + s$  Bits gespeichert werden (Maschinenzahlen). Maschinenzahlen bilden das numerische Gleitkommagitter  $A = A(b, r, s)$

**Beispiel 1.3 ( $b = 2, r = 3, s = 1$ )**

$$m = \frac{1}{2} + m_2 \frac{1}{4} + m_3 \frac{1}{8} \in \left\{ \frac{4}{8}, \frac{5}{8}, \frac{6}{8}, \frac{7}{8} \right\}$$

$$e = e_0 \in \{0, 1\}$$

Da  $A$  endlich ist, gibt es eine größte/kleinste darstellbare Zahl:

$$\begin{aligned} x_{\{min/max\}} &= \pm(b-1)[b^{-1} + \dots + b^{-r}] \cdot b^{(b-1)[b^{s-1} + \dots + b^0]} \\ &= \pm(1 - b^{-r}) \cdot b^{(b^s - 1)} \end{aligned}$$

sowie eine kleinste positive/größte negative Zahl:

$$\begin{aligned} x_{posmin/negmax} &= \pm b^{-1} \cdot b^{-(b-1)[b^{s-1} + \dots + b^0]} \\ &= b^{-b^s} \end{aligned}$$

Das gängigste Format ist das IEEE-Format, das auch hinter dem Python-Datentyp float steht:

$$x = \pm m \cdot 2^{c-1022}$$

Dieser Datentyp ist 64 Bit (8 Byte) groß (doppelte Genauigkeit, double). Davon speichert 1 Bit das Vorzeichen, 52 Bits die Mantisse  $m = 2^{-1} + m_2 2^{-2} + \dots + m_{53} 2^{-53}$  und 11 Bits die Charakteristik  $c = c_0 2^0 + \dots + c_{10} 2^{10}$ , mit  $m_i, c_i \in \{0, 1\}$ . Es gibt folgende spezielle Werte:

- Alle  $c_i, m_i = 0$  :  $x = \pm 0$
- Alle  $m_i = 0, c_i = 1$  :  $x = \pm \infty$
- Ein  $m_i \neq 0$ , alle  $c_i = 1$  :  $x = \text{NaN}$  (not a number)

Für  $c$  bleibt damit ein Bereich von  $\{0, \dots, 2046\}$  beziehungsweise  $c - 1022 \in \{-1022, \dots, 1024\}$ . Damit gilt:

- $x_{max} \approx 2^{1024} \approx 1.8 \times 10^{308}, x_{min} = -x_{max}$
- $x_{posmin} = 2^{-1022} \approx 2.2 \times 10^{-308}, x_{negmax} = -x_{posmin}$

Ausgangsdaten  $x \in \mathbb{R}$  einer numerischen Aufgabe und die Zwischenergebnisse einer Rechnung müssen durch Maschinenzahlen dargestellt werden. Für Zahlen des „zulässigen“ Bereichs  $D = [x_{min}, x_{negmax}] \cup \{0\} [x_{posmin}, x_{max}]$  wird eine Rundungsoperation  $\text{rd} : D \rightarrow A$  verwendet, die

$$|x - \text{rd } x| = \min_{y \in A} |x - y| \forall x \in D$$

erfüllt.

#### Beispiel 1.4 (Natürliche Rundung im IEEE-Format)

$$\text{rd}(x) = \text{sgn}(x) \cdot \begin{cases} 0, m_1, \dots, m_{53} \cdot 2^e & m_{54} = 0 \\ (0, m_1, \dots, m_{53} + 2^{-53}) \cdot 2^e & m_{54} = 1 \end{cases}$$

Rundungsfehler:

- absolut:

$$|x - \text{rd}(x)| \leq \frac{1}{2} b^{-r} b^e$$

- relativ:

$$\left| \frac{x - \text{rd}(x)}{x} \right| \leq \frac{1}{2} \frac{b^{-r} b^e}{|m| b^e} \leq \frac{1}{2} b^{-r+1}$$

Der relative Fehler ist für  $x \in D \setminus \{0\}$  beschränkt durch die „Maschinenengenauigkeit“

$$\text{eps} = \frac{1}{2} b^{-r+1}$$

Für  $x \in D$  ist  $\text{rd}(x) = x(1 + \varepsilon)$ ,  $|\varepsilon| \leq \text{eps}$ . Für das IEEE-Format (double)

$$\text{eps} = \frac{1}{2} 2^{-52} \approx 10^{-16}$$

Arithmetische Grundoperationen

$$* : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}, * \in \{x, -, +, /\}$$

werden auf dem Rechner ersetzt durch Maschinenoperationen:

$$\circledast : A \times A \rightarrow A$$

Dies ist normalerweise für  $x, y \in A$  und  $x * y \in D$  realisiert durch

$$x \circledast y := \text{rd}(x * y) = (x * y)(1 + \varepsilon), |\varepsilon| \leq \text{eps}$$

Dazu werden die Operationen maschinenintern (unter Verwendung einer längeren Mantisse) ausgeführt, normalisiert und dann gerundet. Im Fall  $x * y \notin D$  gibt es eine Fehlermeldung (overflow, underflow)

oder das Ergebnis NaN. Achtung: Das Assoziativ- und Distributivgesetz gilt dann nur näherungsweise. Im Allgemeinen ist für  $x, y, z \in A$

$$\begin{aligned}(x \oplus y) \oplus z &\neq x \oplus (y \oplus z) \\ (x \oplus y) \odot z &\neq (x \odot z) \oplus (y \odot z)\end{aligned}$$

Insbesondere gilt für  $|y| \leq \frac{|x|}{b} \text{eps}$

$$x \oplus y = x$$

Damit ergibt sich eine alternative Charakterisierung der Maschinengenauigkeit:  $\text{eps}$  ist die kleinste positive Zahl in  $A$ , sodass  $1 \oplus \text{eps} \neq 1$

## 1.2 Konditionierung numerischer Aufgaben

Eine numerische Aufgabe wird als **gut konditioniert** bezeichnet, wenn eine kleine Störung in den Eingangsdaten (Messfehler, Rundungsfehler) auch nur eine kleine Änderung der Ergebnisse zur Folge hat.

**Beispiel 1.5 (Schnittpunkt von Geraden)** Zwei Geraden, die sich (annähernd) rechtwinklig treffen sind gut konditioniert.

Zwei Geraden, die sich unter einem stumpfen, oder spitzen Winkel treffen sind schlecht konditioniert.

**Beispiel 1.6 (Lineares Gleichungssystem)**

$$\begin{pmatrix} 1 & 10^6 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \implies \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 & -10^6 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

$$b = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \implies x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$b = \begin{pmatrix} 1 \\ 10^{-3} \end{pmatrix} \implies x = \begin{pmatrix} -999 \\ 10^{-3} \end{pmatrix} \not\approx \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$\implies$  schlecht konditioniert.

**Definition 1.7** Eine **numerische Aufgabe** berechnet aus Eingangsgrößen  $x_j \in \mathbb{R}, j = 1, \dots, m$  unter der funktionellen Vorschrift  $f(x_1, \dots, x_m), i = 1, \dots, n$  Ausgangsgrößen  $y_i = f_i(x_1, \dots, x_m)$

$$y = f(x), f: \mathbb{R}^m \rightarrow \mathbb{R}^n$$

**Beispiel 1.8 (Lösung eines LGS)**  $Ay = x, f(x) = A^{-1}x$

**Definition 1.9** Fehlerhafte Eingangsgrößen  $x_i + \Delta x_i$  ( $\Delta x_i$ : Rundungsfehler, Maschinengenfehler) ergeben fehlerhafte Resultate  $y_i + \Delta y_i$ . Wir bezeichnen  $|\Delta y_i|$  als den absoluten Fehler und  $\left| \frac{\Delta y_i}{y_i} \right|$  für  $y_i \neq 0$  als den relativen Fehler.

### 1.2.1 Differentielle Fehleranalyse

Annahmen:

- kleine relative Datenfehler  $|\Delta x_i| \ll |x_i|$
- $f_i$  stetig partiell differenzierbar nach allen  $x_i$

Dann gilt:

$$\begin{aligned} y_i &= f_i(x_i), y_i + \Delta y_i = f_i(x + \Delta x) \\ \implies \Delta y_i &= f_i(x + \Delta x) - f_i(x) \end{aligned}$$

Taylorentwicklung

$$= \sum_{j=1}^m \frac{\partial f_i}{\partial x_j} \Delta x_j + R_i^f(x, \Delta x)$$

mit einem Restglied  $R_i^f$ , das für  $|\Delta x| = \max_{j=1, \dots, m} |\Delta x_j| \rightarrow 0$  schneller gegen 0 geht als  $|\Delta x|$ .  
Wenn  $f$  sogar zweimal stetig differenzierbar ist, gilt sogar, dass

$$\left| R_i^f(x, \Delta x) \right| \leq c |\Delta x|^2, c \in \mathbb{R}$$

**Definition 1.10 (Landau-Notation)** Seien  $g, h : \mathbb{R}_+ \rightarrow \mathbb{R}, t \rightarrow 0^+$ . Wir schreiben:

- $g(t) = \mathcal{O}(h(t)) : \iff \exists t_0, c \in \mathbb{R}_+ : \forall t \in (0, t_0] : |g(t)| \leq c|h(t)|$
- $gt = \sigma(ht) : \iff \exists t_0 \in \mathbb{R}_+, c : \mathbb{R}_+ \rightarrow \mathbb{R}, \lim_{t \rightarrow 0^+} c(t) = 0 : \forall t \in (0, t_0] : |g(t)| \leq c(t)|h(t)|$

**Bemerkung 1.11** • Analoge Schreibweise für  $t \rightarrow \infty$

- $\mathcal{O}$  und  $\sigma$  sind Symbole, keine Funktionen

$$\mathcal{O}(t^2) + \mathcal{O}(t^3) + \mathcal{O}(2t^2) = \mathcal{O}(t^2) \not\implies \mathcal{O}(t^3) + \mathcal{O}(2t^2) = 0$$

- $\sigma(t^n)$  ist stärker als  $\mathcal{O}(t^n) : \sigma(t^n) + \mathcal{O}(t^n) = \mathcal{O}(t^n)$
- $\mathcal{O}(t^{n+1})$  ist stärker als  $\sigma(t^n)$ : Wähle  $c(t) = t!$

**Beispiel 1.12** Ist  $g(t)$  zweimal stetig differenzierbar, so gilt mit Taylor

$$\begin{aligned} g(t + \Delta t) &= g(t) + \Delta t g'(t) + \frac{1}{2} \Delta t^2 g''(\tau), \tau \in [t, t + \Delta t] \\ \implies \frac{1}{\Delta t} (g(t + \Delta t) - g(t)) &= g'(t) + \mathcal{O}(\Delta t) \end{aligned}$$

Damit folgt dass  $\Delta y_i$  in erster Näherung, das heißt bis auf eine Größe der Ordnung  $\mathcal{O}(|\Delta x|^2)$  gleich

$$\sum_{j=1}^m \frac{\partial f_i}{\partial x_j}(x) \Delta x_j$$

ist. Schreibweise

$$\Delta y_i \doteq \sum_{j=1}^m \frac{\partial f_i}{\partial x_j}(x) \Delta x_j$$

Für den komponentenweisen relativen Fehler gilt

$$\frac{\Delta y_i}{y_i} \doteq \sum_{j=1}^m \frac{\partial f_i}{\partial x_j}(x) \frac{\Delta x_j}{y_i} = \sum_{j=1}^m \underbrace{\frac{\partial f_i}{\partial x_j}(x) \frac{x_j}{f_i(x)}}_{=: k_{ij}(x)} \frac{\Delta x_j}{x_j}$$

Vernachlässigt haben wir dabei

$$\left| \frac{R_i^f(x_j, \Delta x)}{y_i} \right| = \mathcal{O}\left(\frac{|\Delta x|^2}{|y_i|}\right)$$

Diese Vernachlässigung ist nur zulässig falls

$$|\Delta x| = \sigma(|y_i|), i = 1, \dots, n$$

damit

$$\mathcal{O}\left(\frac{|\Delta x|^2}{|y_i|}\right) = \sigma(|\Delta x|)$$

(stärker als  $\mathcal{O}(|\Delta x|)$ )

**Definition 1.13** Die Größen  $k_{ij}(x)$  heißen (relative) Konditionszahlen von  $f$  im Punkt  $x$ . Sie sind Maß dafür, wie sich kleine relative Fehler in den Ausgangsdaten  $x_j$  auf das Ergebnis  $y_i$  auswirken. Sprechweise:

- $|k_{ij}(x)| \gg 1$ : Die Aufgabe  $y = f(x)$  ist schlecht konditioniert
- sonst: Die Aufgabe  $y = f(x)$  ist gut konditioniert
- $|k_{ij}(x)| < 1$ : Fehlerdämpfung
- $|k_{ij}(x)| > 1$ : Fehlerverstärkung.

**Bemerkung 1.14** Man kann auch Störungen in  $f$  betrachten.



**Beispiel 1.15** Implizit gegebene Aufgaben. Für  $n = m$  sei  $y$  die gegebene Eingangsgröße und ein  $x$  mit  $f(x) = y$  die Ausgabe (zum Beispiel:  $f(x) = Ax + b$ ). Die differentielle Fehleranalyse auf der Umkehrfunktion  $x = f^{-1}(y)$  liefert unter geeigneten Annahmen.

$$\frac{\Delta x_i}{x_i} \doteq \sum_{j=1}^n k_{ij}^{-1}(y) \frac{\Delta y_j}{y_j}, k_{ij}^{-1} = \frac{\partial f_i^{-1}}{\partial y_j}(y) \frac{y_j}{x_i}$$

Wir definieren die Matrizen

$$K^{-1}(y) = \left(k_{ij}^{-1}\right)_{i,j=1}^n, K(x) = (k_{ij}(x))_{i,j=1}^n$$

und betrachten deren Produkt:

$$\begin{aligned} (K^{-1}(y)K(x))_{ij} &= \sum_{l=1}^n k_{il}^{-1}(y) k_{lj}(x) \\ &= \sum_{l=1}^n \frac{\partial f_i^{-1}}{\partial y_l}(y) \frac{y_l}{x_i} \frac{\partial f_l}{\partial x_j}(x) \frac{x_j}{y_l} \\ &= \frac{x_j}{x_i} \sum_{l=1}^n \frac{\partial f_i^{-1}}{\partial y_l} \frac{\partial f_l}{\partial x_j} = \frac{x_j}{x_i} \frac{d}{dx_j} (f_i^{-1}(f(x))) \\ &= \frac{x_j}{x_i} \frac{dx_i}{dx_j} = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & \text{sonst} \end{cases} \end{aligned}$$

$K^{-1}$  ist gerade das Inverse von  $K$ .

Wiederholung: Numerische Aufgabe

$$f : x \in \mathbb{R}^m \mapsto y \in \mathbb{R}$$

Konditionszahlen:

$$\begin{aligned} \frac{\Delta y_i}{y_i} &\doteq \sum_{j=1}^m k_{ij}(x) \frac{\Delta x_j}{x_j} \\ k_{ij}(x) &= \frac{\partial f_i}{\partial x_j}(x) \frac{x_j}{f_i(x)} \end{aligned}$$

### 1.2.2 Arithmetische Grundoperationen

Addition:  $f(x_1, x_2) = x_1 + x_2, x_1, x_2 \in \mathbb{R} \setminus \{0\}$

$$\begin{aligned} k_{1j}(x) &= \frac{\partial f}{\partial x_j} \frac{x_j}{f} = 1 \frac{x_j}{x_1 + x_2} = \frac{1}{1 + \frac{x_j}{x_1}} \\ \bar{j} &= \begin{cases} 2 & j = 1 \\ 1 & j = 2 \end{cases} \end{aligned}$$

Die Addition ist schlecht konditioniert für  $x_1 \approx -x_2$ .

**Definition 1.16 (Auslöschung)** Unter Auslöschung versteht man den Verlust von Genauigkeit bei der Subtraktion von Zahlen gleichen Vorzeichens.

**Beispiel 1.17**  $b = 10, r = 4, s = 1$

$$\begin{aligned} x_1 &= 0.112\,587 \times 10^2 & \text{rd}(x_1) &= 0.1126 \times 10^2 \\ x_2 &= 0.112\,448 \times 10^2 & \text{rd}(x_2) &= 0.1124 \times 10^2 \\ x_1 + x_2 &= 0.225\,035 \times 10^2 & \text{rd}(x_1) \oplus \text{rd}(x_2) &= 0.2250 \times 10^2 \\ x_1 - x_2 &= 0.129 \times 10^{-1} & \text{rd}(x_1) \ominus \text{rd}(x_2) &= -0.2 \times 10^{-1} \quad (\text{Großer Fehler}) \end{aligned}$$

Multiplikation:  $y = f(x_1, x_2) = x_1 x_2$

$$k_{1j}(x) = \frac{\partial f}{\partial x_j} \frac{x_j}{f} = x_j - \frac{x_j}{x_1 x_2} = 1$$

$\implies$  gut konditioniert

**Beispiel 1.18 (Lösungen quadratischer Gleichungen)** Für  $p, q \in \mathbb{R}$  betrachte:

$$\begin{aligned} 0 &= y^2 - py + q \\ y_{1,2} &= y_{1,2}(p, q) = \frac{p}{2} \pm \sqrt{\frac{p^2}{4} - q} \end{aligned}$$

nach Vieta  $p = y_1 + y_2, q = y_1 \cdot y_2$

$$\begin{aligned}
 1 &= \frac{dp}{dp} = \frac{\partial y_1}{\partial p} + \frac{\partial y_2}{\partial p} \\
 0 &= \frac{dq}{dp} = \frac{\partial y_1}{\partial p} y_2 + y_1 \frac{\partial y_2}{\partial p} \\
 \implies (y_2 - y_1) \frac{\partial y_2}{\partial p} &= y_2 \\
 \implies \frac{\partial y_2}{\partial p} &= \frac{y_2}{y_2 - y_1} \\
 \implies \frac{\partial y_1}{\partial p} &= \frac{y_1}{y_1 - y_2} \\
 0 &= \frac{dp}{dq} = \frac{\partial y_1}{\partial q} + \frac{\partial y_2}{\partial q} \\
 1 &= \frac{dq}{dq} = \frac{\partial y_1}{\partial q} y_2 + y_1 \frac{\partial y_2}{\partial q} \\
 \implies 1 &= (y_2 - y_1) \frac{\partial y_1}{\partial q} \\
 \implies \frac{\partial y_1}{\partial q} &= \frac{1}{y_2 - y_1} \\
 \implies \frac{\partial y_2}{\partial q} &= -\frac{1}{y_2 - y_1} \\
 k_{11}(x) &= \frac{\partial y_1}{\partial p} \frac{p}{y_1} = \frac{y_1}{y_1 - y_2} \frac{y_1 + y_2}{y_1} = \frac{1 + y_2/y_1}{1 - y_2/y_1} \\
 k_{12}(x) &= \frac{\partial y_1}{\partial q} \frac{q}{y_1} = \frac{1}{y_2 - y_1} \frac{y_1 y_2}{y_1} = \frac{1}{1 - y_1/y_2}
 \end{aligned}$$

Analog für  $k_{21}, k_{22}$

Die Berechnung von  $y_1, y_2$  ist schlecht konditioniert  $y_1 \approx y_2$ .

Konkretes Beispiel:  $p = 4, q = 33.999, y_{1,2} = 2 \pm 10 \times 10^{-1}$

$$k_{12} = \frac{y_2}{y_2 - y_1} = \frac{2 - 10^{-2}}{-2 \times 10^{-2}} = -99.5$$

$\implies$  100-fache Fehlerverstärkung.

### 1.3 Stabilität numerischer Algorithmen

Gegeben: Numerische Aufgabe  $f : x \in \mathbb{R}^m \mapsto y \in \mathbb{R}^n$

**Definition 1.19 (Verfahren / Algorithmus)** Unter einem Verfahren / Algorithmus zur (gegebenenfalls näherungsweise) Berechnung von  $y$  aus  $x$  verstehen wir eine endliche Folge von elementaren Abbildungen  $\varphi^{(k)}$ , die durch sukzessiv Anwendung einen Näherungswert  $\tilde{y}$  zu  $y$  liefern.

$$x = x^{(0)} \mapsto \varphi^{(1)}(x^{(0)}) = x^{(1)} \mapsto \dots \mapsto \varphi^{(k)}(x^{(k-1)}) \mapsto \tilde{y} \rightarrow y$$

Im einfachsten Fall sind die  $\varphi^{(i)}$  arithmetische Grundoperationen. Bei der Durchführung des Algorithmus auf dem Rechner treten in jedem Schritt Fehler auf (Rundungsfehler, Auswertungsfehler, ...), die sich akkumulieren können.

**Definition 1.20 (Algorithmus)** Ein Algorithmus heißt stabil, wenn die im Verlauf der Rechnung akkumulierten Fehler den durch die Konditionierung der Aufgabe  $y = f(x)$  bedingten unvermeidbaren Problemfehler nicht übersteigen.

**Beispiel 1.21 (Lösung quadratischer Gleichungen)** Annahme:  $0 \neq q < p^2/4$

Für  $\left| \frac{y_1}{y_2} \right| \gg 1$ , das heißt  $q \ll \frac{p^2}{4}$ , ist die Aufgabe gut konditioniert. Algorithmus:  $u = p^2/4, v = u - q, w = \sqrt{v}$ .

Im Fall  $p < 0$  wird zur Vermeidung von Auslöschung zunächst  $\tilde{y}_2 = p/2 - w$  berechnet.

Fehlerfortpflanzung:

$$w = \sqrt{u - q} \begin{cases} \approx \frac{|p|}{2} & q > 0 \\ > \frac{|p|}{2} & q < 0 \end{cases}$$

$$\frac{\Delta y_2}{y_2} \leq \left| \frac{\frac{1}{2}p}{\frac{p}{2} - w} \right| \left| \frac{\Delta p}{p} \right| + \left| \frac{-w}{\frac{p}{2} - w} \right| \left| \frac{\Delta w}{w} \right|$$

$$= \underbrace{\left| \frac{1}{1 - \frac{2w}{p}} \right|}_{\leq \frac{1}{2}} \left| \frac{\Delta p}{p} \right| + \underbrace{\left| \frac{1}{1 - \frac{p}{2w}} \right|}_{< 1} \left| \frac{\Delta w}{w} \right|$$

Die zweite Wurzel kann so bestimmt werden:

$$A : \tilde{y}_1 = \frac{p}{2} + w, \quad B : \tilde{y}_1 = \frac{q}{\tilde{y}_2}$$

Für  $|q| \ll \frac{p^2}{4}$  ist  $w \approx \frac{|p|}{2} \implies$  Auslöschung in Variante A

$$\left| \frac{\Delta y_1}{y_1} \right| \leq \underbrace{\frac{1}{1 + \frac{2w}{p}}}_{\gg 1} \frac{\Delta p}{p} + \underbrace{\frac{1}{1 + \frac{p}{2w}}}_{\gg 1} \frac{\Delta w}{w}$$

$\implies$  Variante A ist instabil. Variante B ist stabil:

$$\left| \frac{\Delta y_1}{y_1} \right| \leq \underbrace{\left| \frac{\Delta q}{q} \right|}_{\leq \epsilon_{ps}} + \underbrace{\left| \frac{\Delta y_2}{y_2} \right|}_{\approx \epsilon_{ps}}$$

Regel: Bei der Lösung quadratischer Gleichungen sollten nicht beide Wurzeln aus der Lösungsformel berechnet werden.

Konkretes Beispiel:  $p = -4, q = 0.01$  (vierstellige Rechnung)

$$u = 4, v = 3.99, w = 1.9974948 \dots, \tilde{y}_2 = -3.997(4981 \dots)$$

$$\tilde{y}_1 = \begin{cases} \text{exakt:} & -0.9925915 \dots \\ A : & -0.003000 \quad (\text{rel. Fehler: } 20\%) \\ B : & -0.002502 \quad (\text{rel. Fehler: } 1.7 \times 10^{-4}) \end{cases}$$

**Auswertung arithmetischer Ausdrücke**

Vorwärtsrundungsfehleranalyse: Akkumulation des Rundungsfehlers ausgehend von Startwert.

**Beispiel 1.22**  $y = f(x_1, x_2) = x_1^2 - x_2^2 = (x_1 - x_2)(x_1 + x_2)$  Konditionierung:

$$\begin{aligned} \left| \frac{\Delta y}{y} \right| &\leq \sum_{i=1}^2 \left| \frac{\partial f}{\partial x_i} \frac{x_i}{f} \right| \left| \frac{\Delta x_i}{x_i} \right| \\ &= \left| 2x_1 \frac{x_1}{x_1^2 - x_2^2} \right| \left| \frac{\Delta x_1}{x_1} \right| + \left| -2x_2 \frac{x_2}{x_1^2 - x_2^2} \right| \left| \frac{\Delta x_2}{x_2} \right| \\ &\leq 2 \frac{x_1^2 + x_2^2}{|x_1^2 - x_2^2|} \text{eps} = 2 \left| \frac{\left(\frac{x_1}{x_2}\right)^2 + 1}{\left(\frac{x_1}{x_2}\right)^2 - 1} \right| \text{eps} \end{aligned}$$

$\Rightarrow$  schlecht konditioniert für  $\left| \frac{x_1}{x_2} \right| \approx 1$

Algorithmus A	Algorithmus B
$u = x_1 \odot x_1$	$u = x_1 \oplus x_1$
$v = x_2 \odot x_2$	$v = x_1 \ominus x_2$
$\tilde{q} = u \ominus v$	$\tilde{q} = u \odot v$

Sei  $x_1, x_2 \in A$ . Für Maschinenoperationen  $\otimes$  und  $a, b \in A$  gilt

$$a \otimes b = (a * b)(1 + \varepsilon), |\varepsilon| \leq \text{eps}.$$

Algorithmus A:

$$\begin{aligned} u &= x_1^2(1 + \varepsilon_1), v = x_2^2(1 + \varepsilon_2) \\ \tilde{y} &= (x_1^2(1 + \varepsilon_1) - x_2^2(1 + \varepsilon_2))(1 + \varepsilon_3) \\ &= \underbrace{x_1^2 - x_2^2}_y + x_1^2 \varepsilon_1 - x_2^2 \varepsilon_2 + \underbrace{(x_1^2 - x_2^2)}_y \varepsilon_3, |\varepsilon| \leq \text{eps} \\ \Rightarrow \left| \frac{\Delta y}{y} \right| &\leq \text{eps} \frac{x_1^2 + x_2^2 + |x_1^2 - x_2^2|}{|x_1^2 - x_2^2|} = \text{eps} \left( 1 + \left| \frac{\left(\frac{x_1}{x_2}\right)^2 + 1}{\left(\frac{x_1}{x_2}\right)^2 - 1} \right| \right) \end{aligned}$$

Wegen der Konditionierung des Problems

$$\left| \frac{\Delta y}{y} \right| \leq 2 \left| \frac{\left(\frac{x_1}{x_2}\right)^2 + 1}{\left(\frac{x_1}{x_2}\right)^2 - 1} \right| \text{eps}$$

ist A stabil. Algorithmus B:

$$u = x_1 \oplus x_2, v = x_1 \ominus x_2, y = u \odot v$$

## Rundungsfehleranalyse

$$\begin{aligned}
u &= (x_1 + x_2)(1 + \varepsilon_1), v = (x_1 - x_2)(1 + \varepsilon_2) \\
\tilde{y} &= (x_1 + x_2)(1 + \varepsilon_1)(x_1 - x_2)(1 + \varepsilon_2)(1 + \varepsilon_3) \\
&= \underbrace{x_1^2 - x_2^2}_y + \underbrace{(x_1^2 - x_2^2)}_{\varepsilon_1 + \varepsilon_2 + \varepsilon_3} + \mathcal{O}(\varepsilon^3) \\
\Rightarrow \left| \frac{\Delta y}{y} \right| &\leq |(\varepsilon_1 + \varepsilon_2 + \varepsilon_3)| \leq 3\varepsilon
\end{aligned}$$

$\Rightarrow$  Algorithmus B ist stabiler als Algorithmus A.

Regel: Bei numerischen Rechnungen sollte man die schlechter konditionierten Operationen möglichst frühzeitig ansetzen.

## Wiederholung

- Konditionierung: Eigenschaften einer numerischen Aufgabe
- Stabilität: Eigenschaft eines Verfahrens
  - Auslöschung
- Rundungsfehleranalyse
  - $y = f(x_1, x_2) = x_1^2 - x_2^2 = (x_1 - x_2)(x_1 + x_2)$

## Auswertung von Polynomen

$$y = p(x) = a_0 + a_1x + \dots + a_nx^n$$

Als Modellfall betrachten wir

$$p(x) = a_1x + a_2x^2 = x(a_1 + a_2x)$$

Zwei Varianten für  $\tilde{y} = p(\xi), \xi \in A$

A:  $u = \xi \odot \xi, v = a_2 \odot u, w = a_1 \odot \xi, \tilde{y} = v + w$

B:  $u = a_2 \odot \xi, v = a_1 \oplus u, \tilde{y} = \xi \odot v$

B spart eine arithmetische Operation.

Rundungsfehleranalyse A:

$$\begin{aligned}
u &= \xi^2(1 + \varepsilon_1), v = a_2\xi^2(1 + \varepsilon_1)(1 + \varepsilon_2), w = a_1\xi(1 + \varepsilon_3) \\
\tilde{y} &= (a_2\xi^2(1 + \varepsilon_1)(1 + \varepsilon_2) + a_1\xi(1 + \varepsilon_3))(1 + \varepsilon_4) \\
&= \underbrace{a_2\xi^2 + a_1\xi}_y + \underbrace{(a_2\xi^2 + a_1\xi)}_y \varepsilon_4 + a_2\xi^2(\varepsilon_1 + \varepsilon_2) + a_1\xi\varepsilon_3 + \mathcal{O}(\varepsilon^2) \\
\frac{\Delta y}{y} &\stackrel{.}{=} \varepsilon_4 \frac{a_2\xi^2(\varepsilon_1\varepsilon_2) + a_1\xi\varepsilon_3}{a_2\xi^2 + a_1\xi} \\
&= \varepsilon_4 + \varepsilon_3 + \frac{a_2\xi^2(\varepsilon_1 + \varepsilon_2 - \varepsilon_3)}{a_2\xi^2 + a_1\xi} \\
&= \varepsilon_3 + \varepsilon_3 + \frac{\xi}{\frac{a_1}{a_2} + \xi}(\varepsilon_1 + \varepsilon_2 - \varepsilon_3)
\end{aligned}$$

Variante B:

$$\begin{aligned}
 u &= x_2 \xi (1 + \varepsilon_1), v = (a_1 + a_2 \xi (1 + \varepsilon_1))(1 + \varepsilon_2) \\
 \tilde{y} &= \xi \cdot [a_1 + a_2 \xi (1 + \varepsilon_1)](1 + \varepsilon_2)(1 + \varepsilon_3) \\
 &= \underbrace{\xi(a_1 + a_2 \xi)}_y + a_1 \xi(\varepsilon_2 + \varepsilon_3) + a_2 \xi^2(\varepsilon_1 + \varepsilon_2 + \varepsilon_3) + \mathcal{O}(\varepsilon^2) \\
 \frac{\Delta y}{y} &= \varepsilon_2 + \varepsilon_3 + \frac{a_2 \xi^2}{a_1 \xi + a_2 \xi} \varepsilon_2 = \varepsilon_2 + \varepsilon_3 + \frac{\xi}{\frac{a_1}{x_2} + \xi} \varepsilon_1
 \end{aligned}$$

$\Rightarrow$  Variante B ist etwas stabiler als A im Fall  $\xi \approx -\frac{a_1}{a_2}$  (nahe bei Nullstelle) Allgemein:

$$\begin{aligned}
 p(x) &= a_0 + a_1 x + \dots + a_n x^n \\
 &= a_0 + x(a_1 + x(\dots + x(a_{n-1} + a_n x) \dots))
 \end{aligned}$$

### Definition 1.23 (Horner-Schema)

$$b_n = a_n, b_k = a_k + \xi b_{k-1}, k = n-1, \dots, 0$$

liefert den Funktionswert  $p(\xi) = b_0$  des Polynoms an der Stelle  $x = \xi$ .

Regel: Die Auswertung von Polynomen sollte mit dem Horner-Schema erfolgen.

## 2 Interpolation und Approximation

Grundproblem:

Darstellung und Auswertung von Funktionen.

Aufgabenstellung:

1. Eine Funktion  $f(x)$  ist nur auf einer diskreten Menge von Argumenten  $x_0, \dots, x_n$  bekannt und soll rekonstruiert werden (zum Beispiel für Graph Ausgabe)
2. Eine analytisch gegebene Funktion  $f(x)$  soll auf dem Rechner so dargestellt werden, dass jederzeit Funktionswerte zu beliebigen Argument  $x$  berechnet werden können.

$\rightarrow$  System mit unendlich vielen Freiheitsgraden  $y = f(x)$ . „Simulation“ durch endlich viele Datensätze in Klassen  $P$  von einfach strukturierten Funktionen

- Polynome:  $p(x) = a_0 + a_1 x + \dots + a_n x^n$

- rationale Funktionen:

$$r(x) = \frac{a_0 + a_1 x + \dots + a_n x^n}{b_0 + b_1 x + \dots + b_m x^m}$$

- trigonometrische Funktionen

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

- Exponentialsummen

$$e(x) = \sum_{k=1}^n a_k \exp(b_k x)$$

**Definition 2.1** Geschieht die Zuordnung eines Elementes  $g \in P$  zur Funktion  $f$  durch Fixieren von Funktionswerten

$$g(x_i) = y_i = f(x_i), i = 0, \dots, n$$

so spricht man von **Interpolation**. Ist  $g$  im gewissen Sinne die beste Darstellung von  $f$ , zum Beispiel:  $\max_{a \leq x \leq b} |f(x) - g(x)|$  minimal für  $g \in P$ , oder

$\left( \int_a^b |f(x) - g(x)|^2 dx \right)^{1/2}$  minimal für  $g \in P$  so spricht man von **Approximation**. Die Wahl der Konstruktion von  $g \in P$  hängt von der zu erfüllenden Aufgabe ab. Offenbar ist die Interpolation eine Approximation mit

$$\max_{i=0, \dots, n} |f(x_i) - g(x_i)|$$

für  $g \in P$

Wiederholung: Interpolation und Approximation

- Stützstellen  $x_i$  mit Werten  $y_i, i = 0, \dots, n$
- Klassen  $P$  von Funktion

### Polynominterpolation

Wir bezeichnen mit  $P_n$  den Vektorraum der Polynome vom Grad  $\leq n$ :

$$P_n = \{p(x) = a_0 + a_1x + \dots + a_nx^n \mid a_i \in \mathbb{R}, i = 0, \dots, n\}$$

**Definition 2.2 (Lagrangsche Interpolationsaufgabe)** Die Lagrangsche Interpolationsaufgabe besteht darin zu  $x + 1$  paarweise verschiedenen Stützstellen (auch Knoten genannt)  $x_0, \dots, x_n \in \mathbb{R}$  und gegebenen Knotenwerten  $y_0, \dots, y_n \in \mathbb{R}$  ein Polynom  $p \in P_n$  zu bestimmen mit der Eigenschaft  $p(x_i) = y_i$

**Satz 2.3** Die Lagrangsche Interpolationsaufgabe ist eindeutig lösbar.

**Beweis Eindeutigkeit:** Sind  $p_1, p_2 \in P_n$  Lösungen, so gilt für  $p = p_1 - p_2$ , dass

$$p(x_i) = p_1(x_i) - p_2(x_i) = y_i - y_i = 0, i = 0, \dots, n$$

Also hat  $p$   $n + 1$  Nullstellen und ist folglich identisch Null.  $\implies p_1 = p_2$

**Existenz:** Wir betrachten die Gleichungen

$$p(x_i) = y_i \quad i = 0, \dots, n$$



Dies ist ein lineares Gleichungssystem mit  $n + 1$  Gleichungen und  $n + 1$  Freiheitsgraden.

$$\begin{pmatrix} x_0^0 & x_0^1 & \dots & x_0^n \\ x_1^0 & x_1^1 & & x_1^n \\ \vdots & & \ddots & \vdots \\ x_n^0 & x_n^1 & \dots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Wegen der Eindeutigkeit von  $p$  ist  $\ker V = \{0\}$ . Mit dem Rangsatz ( $\dim \mathbb{R}^{n+1} = \dim \ker V + \dim \operatorname{im} V$ ) liefert  $V$  eine surjektive Abbildung. Damit existiert eine Lösung.  $\square$

Zur Konstruktion des Interpolationspolynoms  $p \in P_n$  verwenden wir die sogenannten Lagrangschen Basispolynome.

$$L_i^{(n)}(x) = \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j} \in P_n, i = 0, \dots, n$$

**Lemma 2.4**  $\{L_i^{(n)}, i = 0, \dots, n\}$  ist eine Basis von  $P_n$

**Beweis** Übung.  $\square$

Offensichtlich gilt:

$$L_i^{(n)}(x_k) = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

**Definition 2.5** Das Polynom

$$p(x) = \sum_{i=0}^n y_i L_i^{(n)}(x) \in P_n$$

hat die gewünschten Eigenschaften

$$p(x_j) = y_j, j = 0, \dots, n$$

und wird die Lagrangsche Darstellung des (Lagrangschen) Interpolationspolynoms zu dem Stützpunkten  $(x_i, y_i), i = 0, \dots, n$  genannt.

Nachteil: Bei Hinzunahme eines weiteren Stützpunktes  $(x_{n+1}, y_{n+1})$  ändern sich die Basispolynome völlig.

Abhilfe: Newtonsche Basispolynome

$$N_0(x) = 1, N_i(x) = (x - x_{i-1})N_{i-1}(x) = \prod_{j=0}^{i-1} (x - x_j)$$

Für den Ansatz

$$p(x) = \sum_{i=0}^n a_i N_i(x)$$

erhält man durch Auswertung von  $x_0, \dots, x_n$  das gestaffelte Gleichungssystem

$$\begin{aligned} y_0 &= p(x_0) = a_0 \\ y_1 &= p(x_1) = a_0 + a_1(x_1 - x_0) \\ &\vdots \\ y_n &= p(x_n) = a_0 + a_1(x_1 - x_0) + \dots + a_n(x_n - x_0) \cdot \dots \cdot (x_n - x_{n-1}) \end{aligned}$$

aus dem sich die Koeffizienten  $a_i$  rekursiv berechnen lassen. Bei Hinzunahme eines weiteren Stützpunktes  $(x_{n+1}, y_{n+1})$  setzt man den Prozess mit der Basisfunktion  $N_{n+1}$  fort. In der Praxis verwendet man folgende stabilere und effizientere Methode

**Satz 2.6 (Newtonsche Darstellung)** Das Lagrangsche Interpolationspolynom zu den Punkten  $(x_0, y_0), \dots, (x_n, y_n)$  lässt sich bezüglich der Newtonschen Polynombasis schreiben in der Form

$$p(x) = \sum_{i=0}^n y[x_0, \dots, x_i] N_i(x)$$

Dabei bezeichnen  $y[x_0, \dots, x_i]$  die zu den Punkten  $(x_i, y_i)$  gehörenden „dividierten Differenzen“, welche rekursiv definiert sind durch

$$L \text{ für } k = 1, \dots, j : i = k - j : y[x_i, \dots, x_{i+k}] = \frac{\overbrace{y[x_{i+1}, \dots, x_{i+k}]}^k - \overbrace{y[x_i, \dots, x_{i+k-1}]}^k}{x_{i+k} - x_i} \quad \text{für } j = 0, \dots, n : y[x_j] = y_j$$

**Beweis** Es bezeichne  $p_{i,i+k} \in P_k$  das Polynom, welches die Punkte  $(x_i, y_i), \dots, (x_{i+k}, y_{i+k})$  interpoliert. Speziell ist  $p_{0,n} = p$  das gesuchte Interpolationspolynom. Wir zeigen

$$p_{i,i+k}(x) = y[x_i] + y[x_i, x_{i+1}](x - x_i) + \dots + y[x_i, \dots, x_{i+k}](x - x_i) \cdot \dots \cdot (x - x_{i+k-1})$$

was für  $i = 0$  und  $k = n$  den Satz beweist. Der Beweis wird durch Induktion über die Indextdifferenz  $k$  geführt. Für  $k = 0$  ist  $p_{i,i} = y_i = y[x_i]$ ,  $i = 0, \dots, n$ . Sei die Behauptung richtig für  $k - 1 \geq 0$ . Nach Konstruktion gilt für ein  $a \in \mathbb{R}$

$$p_{i,i+k}(x) = p_{i,i+k-1}(x) + a(x - x_1) \cdot \dots \cdot (x - x_{i+k-1}) = 0$$

für  $x = x_j$ ,  $j = i, \dots, i + k - 1$ . Zu zeigen:  $a = y[x_i, \dots, x_{i+k}]$ . Offenbar ist  $a$  der Koeffizient von  $x^k$  in  $p_{0,i+k}$ . Nach Induktionsannahme ist also

$$\begin{aligned} p_{i,i+k-1}(x) &= \dots + y[x_i, \dots, x_{i+k-1}]x^{k-1} \\ p_{i+1,i+k-1}(x) &= \underbrace{\dots}_{\text{Grad} \leq k-2} + y[x_{i+1}, \dots, x_{i+k}]x^{k-1} \end{aligned}$$

Ansatz:

$$\begin{aligned} q(x) &= \frac{(x - x_i)p_{i+1,i+k}(x) - (x - x_{i+k})p_{i,i+k-1}(x)}{x_{i+k} - x_i} \\ &= p_{i,i+k-1}(x) + \frac{(x - x_i)p_{i+1,i+k}(x) - (x - x_{i+k} + x_{i+k} - x_i)p_{i,i+k-1}(x)}{x_{i+k} - x_i} \\ &= p_{i,i+k-1}(x) + (x - x_i) \frac{p_{i+1,i+k}(x) - p_{i,i+k-1}(x)}{x_{i+k} - x_i} \end{aligned}$$

Es gilt:

$$\begin{aligned} q(x_i) &= y_i, q(x_{i+k}) = \frac{(x_{i+k} - x_i)y_{i+k} + 0}{x_{i+k} - x_i} = y_{i+k} \\ q(x_j) &= \frac{(x_j - x_i)y_j - (x_j - x_{i+k})y_j}{x_{i+k} - x_i} = y_j, j = i + 1, \dots, i + k - 1 \end{aligned}$$

$\Rightarrow q$  interpoliert die Stützpunkte  $(x_i, y_i), \dots, (x_{i+k}, y_{i+k}) \Rightarrow q \equiv p_{i,i+k}$  (Eindeutigkeit des Interpolationspolynoms). Der führende Koeffizient in  $p_{i,i+k}(x)$  ist demnach

$$\begin{aligned} q &= \frac{y[x_{i+1}, \dots, x_{i+k}] - y[x_i, \dots, x_{i+k-1}]}{x_{i+k} - x_i} \\ &= y[x_i, \dots, x_{i+k}] \quad \square \end{aligned}$$

**Korollar 2.7** Sei  $\sigma : \{0, \dots, n\} \rightarrow \{0, \dots, n\}$  eine beliebige Permutation. Dann gilt für die Stützpunkte  $(\tilde{x}_i, \tilde{y}_i) = (x_{\sigma(j)}, y_{\sigma(j)})$

$$y[\tilde{x}_0, \dots, \tilde{x}_n] = y[x_0, \dots, x_n]$$

**Beweis** Koeffizient des Monoms  $x^n$  ist  $y[x_0, \dots, x_n]$  unabhängig von der Reihenfolge.  $\square$

Wiederholung: Lagrange-Interpolation:

Gegeben:  $(x_i, y_i), i = 0, \dots, n$

Suche  $p \in P_n : p(x_i) = y_i, i = 0, \dots, n$

Lösung:

$$\begin{aligned} p(x) &= \sum_{i=0}^n y_i L_i^{(n)}(x) \\ &= L_i^{(n)}(x) \qquad \qquad \qquad = \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j} \in P_n \end{aligned}$$

$$\Rightarrow L_i^{(n)}(x_j) = \delta_{ij}$$

Andere Darstellung: Newton-Neville

$$N_i(x) = \prod_{j=0}^{n-1} (x - x_j)$$

$$p(x) = \sum_{i=0}^N y[x_0, \dots, x_i] D_i(x)$$

$$y[x_i] = q_i$$

$$y[x_i, \dots, x_{i+k}] = \frac{y[x_{i+1}, \dots, x_{i+k}] - y[x_i, \dots, x_{i+k-1}]}{x_{i+k} - x_i}$$

**Definition 2.8** Das durch die Rekursion  $j = 0, \dots, n, p_{j,j}(x) = y_j$  für  $k = 1, \dots, j : i = k - j$

$$p_{i,i+k}(x) = p_{i,i+k-1}(x) + (x - x_i) \frac{p_{i+1,i+k}(x) - p_{i,i+k-1}(x)}{x_{i+k} - x_i}$$

erzeugte Polynom  $p_{0,1}$  ist die sogenannte Nevellsche Darstellung des Interpolationspolynom zu den Stützstellen  $(x_0, y_0), \dots, (x_n, y_n)$

Schema:

	$k = 0$		$k = 1$		$k = 2$	$\dots$	$k = n - 1$		$k = n$
$x_0$	$y_0$	$\rightarrow$	$p_{0,1}$	$\rightarrow$	$p_{0,2}$	$\dots$	$p_{0,n-1}$	$\rightarrow$	$p_{0,n}$
$x_1$	$y_1$	$\nearrow$	$p_{1,2}$	$\nearrow$	$p_{1,3}$	$\dots$	$p_{1,n}$	$\nearrow$	
$x_2$	$y_2$	$\nearrow$							
$\vdots$		$\vdots$	$\ddots$						
$x_{n-1}$	$y_{n-1}$	$\rightarrow$	$p_{n-1,n}$						
$x_n$	$y_n$	$\nearrow$							

Die Hinzunahme eines weiteren Stützpunktes ist problemlos. Die Auswertung von  $p_{0,n}(x)$  an einer Stelle  $\xi \neq x_i$  ohne vorige Bestimmung der Koeffizienten der Newton-Darstellung ist damit sehr einfach und numerisch effizient und stabil möglich. Dazu wird im Schema  $x$  mit  $\xi$  ersetzt.

## 2.1 Auswertung von Polynomen und deren Ableitungen

Sei  $p \in P_n$  gegeben in der Darstellung

$$p(x) = a_0 + a_1 x + \dots + a_n x^n$$

Wiederholung: Auswertung von  $p(\xi)$  mittels Horner-Schema

$$b_k = \begin{cases} a_n & k = n \\ a_k + \xi b_{k+1} & k = n - 1, \dots, 0 \end{cases}$$

$$\implies p(\xi) = b_0.$$

Zu  $p_n = p \in P_n$  und festem  $\xi$  wird durch

$$p_{n-1}(x) = b_1 + b_2x + \dots + b_nx^{n-1}$$

ein Polynom  $p_{n-1} \in P_{n-1}$  definiert. Wegen  $a_k = b_k - \xi b_{k+1}, k = 0, \dots, n-1, a_n = b_n$ :

$$\begin{aligned} p_n(x) &= \sum_{k=0}^n b_k x^k - \xi \sum_{k=0}^{n-1} b_{k+1} x^k \\ &= b_0 + x \sum_{k=1}^n b_k x^{k-1} - \xi \sum_{k=1}^n b_k x^{k-1} \\ &= r_0 + (x - \xi) p_{n-1}(x) \quad r_0 = p(\xi) = b_0 \end{aligned}$$

$\implies$  Für eine Nullstelle  $\xi$  von  $p_n$  leistet das Horner-Schema die Abspaltung des Linearfaktors  $(x - \xi)$  vom Polynom  $p_n$ . Weiter ist dann für  $x \neq \xi$

$$\frac{p_n(x) - p_n(\xi)}{x - \xi} = p_{n-1}(x)$$

$$x \rightarrow \xi$$

$$p'_n(\xi) = p_{n-1}(\xi)$$

Zur Berechnung von  $p'_n(\xi)$  wird das Horner-Schema auf  $p_{n-1}$  angewendet.

$$p_{n-2} \in P_{n-2}, p_{n-1}(x) = r_1 + (x - \xi)p_{n-2}(x), r_1 = p_{n-1}(\xi)$$

Fortsetzen  $\rightarrow$  endliche Folge von Polynomen  $p_n, p_{n-1}, \dots, p_0$  mit

$$\begin{aligned} p_{n-j}(x) &= (x - \xi)p_{n-j-1}(x) + r_j, \quad j = 0, \dots, n \\ p_n(x) &= r_0 + r_1(x - \xi) + \dots + r_n(x - \xi)^n \end{aligned}$$

Vergleich mit der Taylorentwicklung von  $p_n$  um  $\xi$  ergibt

$$r_j = \frac{1}{j!} p_n^{(j)}(\xi)$$

Die Koeffizienten von  $p_{n-j}$  seien

$$p_{n-j}(x) = a_j^{(j)} + a_{j+1}^{(j)}x + \dots + a_n^{(j)}x^{n-j}, j = 0, \dots, n$$

Es gilt die Rekursion:

$$a_k^{(j+1)} = \begin{cases} a_n^{(j)} & k = n \\ a_k^{(j)} + \xi a_{k+1}^{(j)} & \end{cases}$$

und es gilt

$$p^{(j)}(\xi) = j! a_j^{(j+1)}, j = 0, \dots, n$$

Dieses „vollständige Horner-Schema“ kann leicht zur Auswertung von Polynomen in Newton-Darstellung modifiziert werden:

$$p(x) = a_0 + a_1(x - x_0) + \dots + a_n(x - x_0) \cdot \dots \cdot (x - x_{n-1})$$

## 2.2 Interpolation von Funktionen

Stützstellen  $x_0, \dots, x_n \in [a, b]$ . Werte gegeben durch Funktion  $y_i = f(x_i), i = 0, \dots, n$

**Frage:** Wie gut approximiert das Interpolationspolynom  $p \in P_n$  die Funktion  $f$  auf  $[a, b]$ ?

**Bezeichnungen:**

- $\overline{(x_0, \dots, x_n)}$  = kleinstes Intervall, das alle  $x_i$  enthält.
- $C[a, b]$  : Vektorraum der über  $[a, b]$  stetigen Funktionen
- $C^k[a, b]$  : Vektorraum über  $[a, b]$  k-mal stetig differenzierbarer Funktionen.

**Satz 2.9 (Interpolationsfehler 1)** Sei  $f \in C^{n+1}[a, b]$ . Dann gibt es zu jedem  $x \in [a, b]$  ein  $\xi_x \in \overline{(x_0, \dots, x_n, x)}$ , sodass gilt

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{j=0}^n (x - x_j)$$

**Beweis** Für  $x \in \{x_0, \dots, x_n\}$  ist alles klar. Sei  $x \in [a, b] \setminus \{x_0, \dots, x_n\}$ . Wir setzen

$$l(t) = \prod_{j=0}^n (t - x_j), \quad c(x) = \frac{f(x) - p(x)}{l(x)}$$

Die Funktion

$$F(t) = f(t) - p(t) - c(x)l(t)$$

besitzt dann mindestens die  $n+2$  Nullstellen  $x_0, \dots, x_n, x$  in  $[a, b]$ . Durch wiederholte Anwendung des Satzes von Rolle schließt man, dass die Ableitung  $F^{n+1}$  eine Nullstelle  $\xi_x \in \overline{(x_0, \dots, x_n, x)}$ . Es

$$\begin{aligned} 0 &= F^{(n+1)}(\xi_x) = f^{(n+1)}(\xi_x) - p^{(n+1)}(\xi_x) - c(x)l^{(n+1)}(\xi_x) \\ &= f^{(n+1)}(\xi_x) - c(x)(n+1)! \end{aligned} \quad \square$$

Wiederholung:

- Neville-Schema für  $p \in P_n$ :  
$$p(x_i) = y_i, i = 0, \dots, n$$
- Vollständiges Horner-Schema
- Interpolation von Funktionen  $y_i = f(x_i)$

Interpolationsfehler 1: Sei  $f \in C^{n+1}[a, b] \implies \forall x \in [a, b] \exists \xi_x \in \overline{(x_0, \dots, x_n, x)}$ :

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{j=0}^n (x - x_j)$$

**Satz 2.10 (Interpolationsfehler 2)** Sei  $f \in C^{n+1}[a, b]$ . Dann gilt für  $x \in [a, b] \setminus \{x_0, \dots, x_n\}$ :

$$f(x) - p(x) = f[x_0, \dots, x_n, x] \prod_{j=0}^n (x - x_j)$$

mit der Notation

$$f[x_i, \dots, x_{i+k}] = y[x_i, \dots, x_{i+k}]$$

und es ist

$$f[x_0, \dots, x_n, x] = \int_0^1 \int_0^{t_1} \dots \int_0^{t_n} f^{(n+1)}(x_0 + t_1(x_1 - x_0) + \dots + t_n(x_n - x_{n-1}) + t(x - x_n)) dt dt_n \dots dt_1$$

**Beweis** Per Induktion über  $n$ .

IA:  $n = 0$ :

$$f(x) - p_0(x) = f(x) - f(x_0) = \begin{cases} f[x_0, x](x - x_0) \\ (x - x_0) \int_0^1 f'(x_0 + t(x - x_0)) dt \end{cases}$$

wobei ein

$$\int_0^1 g'(t) dt = g(1) - g(0)$$

für  $g(t) = f(x_0 + t(x - x_0)) \implies g'(t) = f'(t)(x - x_0)$

Sei die Behauptung richtig für  $n - 1 \geq 0$ . Dann ist

$$\begin{aligned} f(x) - p_n(x) &= f(x) - \sum_{i=0}^n f[x_0, \dots, x_n] \prod_{j=0}^{i-1} (x - x_j) \\ &= f(x) - p_{n-1}(x) - f[x_0, \dots, x_n] \prod_{j=0}^{n-1} (x - x_j) \\ &= f[x_0, \dots, x_{n-1}, x] \prod_{j=0}^{n-1} (x - x_j) - f[x_0, \dots, x_n] \prod_{j=0}^{n-1} (x - x_j) \\ &= (f[x_0, \dots, x_{n-1}, x] - f[x_0, \dots, x_n]) \prod_{j=0}^{n-1} (x - x_j) \\ &= \frac{f[x, x_0, \dots, x_n] - f[x_0, \dots, x_n]}{x - x_n} \prod_{j=0}^{n-1} (x - x_j) \\ &= f[x_0, \dots, x_n, x] \prod_{j=0}^{n-1} (x - x_j) \end{aligned}$$

Weiterhin gilt:

$$f[x_0, \dots, x_{n-1}, x] - f[x_0, \dots, x_n] = \int_0^1 \int_0^{t_1} \dots \int_0^{t_{n-1}} [f^{(n)}(x_0 + t_1(x_1 - x_0) + \dots + t_n(x - x_{n+1})) - f^{(n)}(x_0)] dt dt_{n-1} \dots dt_1$$

Setze  $g(t) = f^{(n)}(x_0 + t_1(x_1 - x_0) + \dots + t_n(x_n - x_{n-1}) + t_{x-x_n})$

$$\begin{aligned}
 &= \int_0^1 \int_0^{t_1} \dots \int_0^{t_{n-1}} [g(t_n) - g(0)] dt_n \dots dt_1 \\
 &= \int_0^1 \int_0^{t_1} \dots \int_0^{t_{n-1}} \int_0^{t_n} f^{(n+1)}(x_0 + t_1(x_1 - x_0) + \dots + t_n(x_n - x_{n-1}) + t_{x-x_n}) dt_n \dots dt_1 \\
 \implies f[x_0, \dots, x_n, x] &= \int_0^1 \int_0^{t_1} \dots \int_0^{t_n} f^{(n+1)}(\dots) dt_n \dots dt_1 \quad \square
 \end{aligned}$$

Die Integraldarstellung der dividierten Differenzen gestattet ihre stetige Fortsetzung für den Fall, das Stützstellen zusammenfallen:

$$f[x_0, \dots, x_r, x_r, \dots, x_n] = \lim_{\varepsilon \rightarrow 0} f[x_0, \dots, x_r, x_r + \varepsilon, \dots, x_n]$$

Im Extremfall  $x_0 = x_1 = \dots = x_n$  wird

$$\begin{aligned}
 f[x_0, \dots, x_n] &= \int_0^1 \int_0^{t_1} \dots \int_0^{t_{n-1}} f^{(n)}(x_0) dt_n \dots dt_1 \\
 &= \int_0^1 \int_0^{t_1} \dots \int_0^{t_{n-1}} 1 dt_n \dots dt_1 f^{(n)}(x_0) \\
 &= \frac{1}{n!} f^{(n)}(x_0)
 \end{aligned}$$

Damit geht das Newtonsche Interpolationspolynom über in das Taytorpolynom n-ten Grades von  $f$  in  $x_0$ . Konstruieren wir die Fehlerdarstellung so erhalten wir für ein  $\xi_x \in (x_0, \dots, x_n, x)$

$$\begin{aligned}
 \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{j=0}^n (x - x_j) &= f(x) - p(x) \\
 &= f[x_0, \dots, x_n, x] \prod_{j=0}^n (x - x_j) \\
 \implies f[x_0, \dots, x_n, x] &= \frac{f^{(n+1)}(\xi_x)}{(n+1)!}
 \end{aligned}$$

**Definition 2.11 (Hermit-Interpolation)** Die Hermite'sche Interpolationsaufgabe lautet:

Gegeben  $x_i, i = 0, \dots, m$  (paarweise verschieden),  $y_i^{(k)}, i = 0, \dots, m, k = 0, \dots, \mu_i, \mu_i \geq 0$ .

Gesucht:  $p \in P_n, n = m + \sum_{i=0}^m \mu_i, p^{(k)}(x_j) = y_j^{(k)}, i = 0, \dots, m, k = 0, \dots, \mu_i, (\mu_i + 1)$ -fache Stützstellen.

**Beispiel 2.12**  $x_0 = -1, x_1 = 1, m = 1, y_0^{(0)} = 0, y_1^{(0)} = 1, y_1^{(1)} = 2 \implies \mu_0 = 0, \mu_1 = 1 \implies n = 1 + 0 + 1 = 2 \implies p(x) = x^2$

Analog zur Lagrange-Interpolation:



- Existenz + Eindeutigkeit
- Darstellung des Interpolationsfehlers

Wiederholung: Fehlerdarstellung Lagrange-Interpolation. Sei  $f \in C^{n+1}[a, b]$ .  $\exists \xi_x \in \overline{(x_0, \dots, x_n, x)}$

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{j=0}^n (x - x_j)$$

$$f(x) - p(x) = f[x_0, \dots, x_n, x] \prod_{j=0}^n (x - x_j)$$

$$f[x_0, \dots, x_n, x] = \int_0^1 \int_0^{t_1} \dots \int_0^{t_n} f^{(n+1)}(x_0 + t_1(x_1 - x_0) + \dots + t_n(x_n - x_{n-1}) + t(x_n - x)) dt dt_n \dots dt_1$$

Hermite-Interpolation: Such  $p \in P_n$ ,  $n = m + \sum_{i=0}^m \mu_i$

$$p^{(k)}(x_i) = y_i^{(k)}, i = 0, \dots, m, k = 0, \dots, \mu_i$$

### 2.3 Richardsonsche Extrapolation zum Limes

Gegeben: Numerischer Prozess mit Werten  $a(h)$ ,  $h \in \mathbb{R}_+$ ,  $h \rightarrow 0$ .

Gesucht:  $a(0) = \lim_{h \rightarrow 0} a(h)$

Idee: Für  $h_i > 0$ ,  $i = 0, \dots, n$ , interpoliere  $(h_i, a(h_i))$  und berechne  $p_n(0)$

**Beispiel 2.13 (Numerische Differentiation)** Sei  $f \in C^\infty[a, b]$ ,  $x \in (a, b)$ . Nach Taylor gilt

$$a(h) = \frac{f(x+h) - f(x-h)}{2h} = f'(x) + \sum_{i=1}^{\infty} \frac{f^{(2i+1)}(x)}{(2i)!} h^{2i}$$

**Satz 2.14 (Extrapolationsfehler)** Für  $h \in \mathbb{R}_+$  habe  $a(n)$  die Entwicklung

$$a(h) = a_0 + \sum_{j=1}^n a_j h^{jq} + a_{n+1}(h) h^{(n+1)q}$$

mit  $q > 0$ , Koeffizienten  $a_j$  und

$$a_{n+1}(h) = a_{n+1} + \mathcal{O}(1)$$

Die Folge  $(h_i)_{i \in \mathbb{N}}$  erfülle

$$0 \leq \frac{h_{k+1}}{h_k} \leq \rho < 1$$

( $\implies h_k$  positiv, monoton fallend). Dann gilt für das Interpolationspolynom  $p_1^{(k)} \in P_n$  (in  $h^q$ ) durch

$$(h_k^q, a(h_k)), \dots, (h_{k+n}^q, a(h_{k+n}))$$

$$a(0) - p_n^{(k)}(0) = \mathcal{O}(h_k^{(n+1)q})$$

( $k \rightarrow \infty$ )

**Beweis** Abkürzungen  $z = h^q$ ,  $z_k = h_k^q$ . Interpoliere  $(z_{k+i}, a(h_{k+i}))$ ,  $i = 0, \dots, n$ .

$$p_n(z) = \sum_{i=0}^n a(h_{k+i}) L_{k+i}^{(n)}(z)$$

$$L_{k+1}^{(n)}(z) = \prod_{\substack{l=0 \\ l \neq i}}^n \frac{z - z_{k+l}}{z_{k+1} - z_{k+l}}$$

Übung:

$$\sum_{i=0}^n x_{k+1}^n(0) = \begin{cases} 1 & r = 0 \\ 0 & r = 1, \dots, n \\ (-1)^n \prod_{j=0}^n z_{k+j} & r = n+1 \end{cases}$$

$$\begin{aligned} p_n(0) &= \sum_{i=0}^n \left( a_0 + \sum_{j=1}^n a_j z_{k+i}^j + a_{n+1}(h_{k+1}) z_{k+i}^{n+1} \right) L_{k+i}^{(n)}(0) \\ &= a_0 \underbrace{\sum_{i=0}^n L_{k+1}^{(n)}}_{=1} + \sum_{j=1}^n a_j \underbrace{\sum_{i=0}^n z_{k+1}^j L_{k+i}^{(n)}(0)}_0 \\ &= +a_{n+1} \underbrace{\sum_{i=0}^n z_{k+1}^{n+1} L_{k+1}^{(n)}}_{=(-1)^n \prod_{i=0}^n z_{k+i}} + \sum_{i=0}^n \imath(1) z_{k+i}^{n+1} L_{k+i}^{(n)}(0) \end{aligned}$$

Da man Landau-Symbole nicht ausklammern darf, schätzen wir ab:

$$\begin{aligned} \left| L_{k+i}^{(n)}(0) \right| &= \prod_{\substack{l=0 \\ l \neq i}}^n \left| \frac{z_k + l}{z_{k+i} - z_{k+l}} \right| \\ &\leq \prod_{l=0}^{i-1} \left| \frac{z_{n+l}}{z_{k+i} - z_{k+l}} \right| \prod_{l=1+i}^n \left| \frac{z_{k+i}}{z_{k+i} - z_{k+l}} \right| \\ &= \prod_{l=0}^{i-1} \frac{1}{\left| \frac{z_{k+i}}{z_{k+l}} - 1 \right|} \prod_{l=i+1}^n \frac{1}{\left| 1 - \frac{z_{k+l}}{z_{k+i}} \right|} \\ &\leq \frac{1}{(1 - \rho^q)^n} \\ \implies p_n(0) &= a_0 + a_{n+1} (-1)^n \prod_{i=0}^n z_{k+i} + \imath(z_k^{n+1}) \\ &= a_0 + \mathcal{O}(h_k^{(n+1)q}) \end{aligned}$$

□

## 2.4 Spline-Interpolation

Problem: Oszillationen des Interpolationspolynoms, wenn man Stützstellen nicht geeignet wählen kann. Abhilfe: Stückweise polynomielle Interpolation:

- Zerlegung:  $a = x_0 < x_1 < \dots < x_n = b$
- Teilintervalle:  $I_i = [x_{i-1}, x_i], i = 1, \dots, n$
- Feinheit:  $h = \max_{i=1, \dots, n} h_i$  mit  $h_i = |I_i| = x_i - x_{i-1}$
- Vektorräume stückweise polynomieller Funktionen

$$S_n^{k,r}[a, b] = \{p \in C^r[a, b] \mid p|_{I_i} \in P_k(i_i)\}, i = 1, \dots, n$$

**Beispiel 2.15 (Stückweise lineare Interpolation)**  $\implies p \in S_n^{(1,0)}[a, b]$ . Fehlerabschätzung:

$$\max_{x \in [a, b]} |f(x) - p(x)| \leq \frac{1}{2} h^2 \max_{x \in [a, b]} |f''(x)|$$

**Beispiel 2.16 (Splines)** Zweimal stetig differenzierbare, stückweise kubische Polynome. Motivation: Biegestab. Minimiere Biegeenergie

$$\int_{x_0}^{x_n} s''(x)^2 dx$$

**Definition 2.17 (Kubischer Spline)** Eine Funktion  $s_n : [a, b] \rightarrow \mathbb{R}$  heißt kubischer Spline bezüglich  $a = x_0 < x_1 < \dots < x_n = b$ , wenn gilt

1.  $s_n \in C^2[a, b]$
2.  $s_n|_{I_i} \in P_3, i = 1, \dots, n$

Gilt zusätzlich

3.  $s_n''(a) = s_n''(b) = 0$  so heißt  $s_n$  natürlicher Spline.

Existenz des interpolierenden kubischen Spline zu Knotenwerten  $s_n(x_i) = y_i, i = 0, \dots, n$

**Satz 2.18 (Spline-Interpolation)** Der interpolierende kubische Spline existiert und ist eindeutig bestimmt durch zusätzliche Vorgabe von  $s_n''(a), s_n''(b)$

**Beweis**  $s$  hat die Form

$$s(x)|_{I_i} = p_i(x), i = 1, \dots, n, p_i \in P_3(I_i)$$

4 Koeffizienten auf jedem der  $n$  Intervalle ergeben  $4n$  Freiheitsgrade. Zur Bestimmung:

$s(x_i) = y_i, i = 0, \dots, n$	2n Gleichungen
$s' \in C[a, b]$	$n - 1$
$s'' \in C[a, b]$	$n - 1$
Zusatzbedingung für $s_n''(a), s_n''(b)$	2
<hr/>	
	4n

$\Rightarrow$  quadratisches lineares Gleichungssystem,  $4n \times 4n$

$$N = \{\omega \in C^2[a, b] \mid \omega_{x_i} = 0, i = 0, \dots, n\}$$

Seien  $s_n^{(1)}$  und  $s_n^{(2)}$  interpolierende Splines  $\Rightarrow s = s_n^{(1)} - s_n^{(2)} \in N$ . Für  $\omega \in N$  beliebig:

$$\begin{aligned} \int_a^b s''(x)\omega''(x)dx &= \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} s''(x)\omega''(x)dx \\ &= \sum_{i=0}^{n-1} \left[ - \int_{x_i}^{x_{i+1}} s^{(3)}\omega' dx + s''\omega' \Big|_{x_i}^{x_{i+1}} \right] \\ &= \sum_{i=0}^{n-1} \left[ - \int_{x_i}^{x_{i+1}} s^{(4)}\omega dx - s^{(3)}\omega \Big|_{x_i}^{x_{i+1}} + s''\omega' \Big|_{x_i}^{x_{i+1}} \right] \\ &= \sum_{i=0}^{n-1} s''\omega' \Big|_{x_i}^{x_{i+1}} = s''(x)\omega'(x) - s''(a)\omega'(a) \\ &= 0 \end{aligned}$$

Speziell für  $\omega = s$

$$\int_a^b |s''(x)|^2 dx = 0$$

$\Rightarrow s$  ist linear  $0 = s(a) = s(b) = 0$

□