5 Interpolation & Approximation

5.1 Einführung

Worum es geht: Darstellung und Auswertung von Funktionen im Rechner.

Anwendungen:

- Rekonstruktion eines funktionalen Zusammenhangs aus „gemessenen" Funktionswerten; Auswertung an Zwischenstellen.
- Teuer auszuwertende Funktionen effizienter auswerten
- Darstellung von Fonts (2D), Körpern (3D) im Rechner. Voraussetzung für Simulation; Szenen in der Computergrafik.
- Lösungen von Differential- und Integralgleichungen
- Datenkompression

  $\Rightarrow$ Beispiel auf den Folien.

Wir beschränken uns hier weitgehend auf Funktionen in einer Variablen, also etwa

$$f \in C^r[a,b].$$

$C^r[a,b]$ ist ein unendlichdimensionaler (Vektor-)raum von Funktionen. Im Rechner betrachten wir Funktionenklassen die durch eine endliche Zahl von Parametern bestimmt sind (müssen keine linearen (!) Teilräume sein!). z. B.

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

b) $r(x) = \dfrac{a_0 + a_1 x + \dots + a_n x^n}{b_0 + b_1 x + \dots + a_m x^m}$ (rationale Funktionen)

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

(trigonometrische Polynome)

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

Grundaufgabe der <u>Approximation</u>:

Gegeben eine Menge von Funktionen P (siehe oben) sowie eine Funktion f (z.B. in $C[a,b]$).

Finde $g \in P$ so dass der Fehler $f-g$ in geeigneter Weise minimiert wird.

Beispiele:

a) $\left(\int_a^b (f-g)^2 dx \right)^{1/2} \to \min$

b) $\max_{a \leq x \leq b} |f(x) - g(x)| \to \min$

c) $\max_{i=0,\dots,n} |f(x_i) - g(x_i)| \to \min$ für $a \leq x_i \leq b$, $i = 0, \dots, n$.

Man spricht von <u>Interpolation</u> falls g durch

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

festgelegt wird.

Dies ist ein Spezialfall der Approximationsaufgabe.

5.2 Polynominterpolation

P_n sei die Menge der Polynome über $\mathbb{R}$ vom Grad kleiner gleich $n \in \mathbb{N}$

$$P_n := \left\{ p(x) = \sum_{i=0}^{n} a_i x^i \;\middle|\; a_i \in \mathbb{R} \right\}.$$

P_n ist ein $n+1$-dimensionaler Vektorraum.

Die Monome $1, x, x^2, \dots, x^n$ bilden eine Basis von P_n

Zu gegebenen $n+1$ $\underbrace{\text{Stützstellen}}_{}$ ((paarweise verschiedenen)) $x_0, x_1, \dots, x_n$ ist die Interpolationsaufgabe

$$p \in P_n \; : \quad p(x_i) = y_i := f(x_i) \qquad i = 0, \dots, n$$

äquivalent zu dem linearen Gleichungssystem

$$\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ & & & & \\ & & & & \\ 1 & x_n & x_n^2 & & x_n^n \end{bmatrix}}_{V[x_0, \dots, x_n]} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

- $V[x_0, \dots, x_n]$ heißt Vandermonde-Matrix
- $x_i = x_j \Rightarrow$ Zeile i = Zeile $j \Rightarrow V$ nicht regulär
- aber ist V für paarweise verschiedene x_i regulär? Ja! Siehe unten. (Determinante lässt sich auch direkt ausrechnen).
- V ist schlecht konditioniert für große n $\operatorname{cond}_\infty(V) = \mathcal{O}(2^n)$! für x_i positiv.

Wie macht man es besser und einfacher?

Lagrange-Interpolation

Definition 5.1

Zu den paarweise verschiedenen Stützstellen x_i, $i=0,\dots,n$, definiere die sog. Lagrange-Polynome:

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

Diese Polynome haben die Eigenschaften:

a) $L_i^{(n)}$ hat Grad n.

klar wg: $\prod_{\substack{j=0\\j\neq i}}^{n}(x-x_j)$ hat n Faktoren. $\dots x^n + \dots$

b) Es gilt

Kronecker-Delta

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

$i\neq k$: Im Produkt $\prod_{\substack{j=0\\j\neq i}}^{n}(x-x_j)$ kommt $j=k\neq i$ vor und damit $(x_k-x_k)=0$ als ein Faktor.

$i=k$: $L_i^{(n)}(x_i) = \prod_{\substack{j=0\\j\neq i}}^{n} \underbrace{\frac{x_i-x_j}{x_i-x_j}}_{=1} = 1.$

c) Die $L_i^{(n)}$ bilden eine Basis von P_n

$L_k^{(n)}$ ist das einzige Polynom unter den $L_i^{(n)}$ mit $L_i(x_k)=1$, daher ist es unabhängig von den $L_i^{(n)}$, $i\neq k$.

Da P_n $n+1$ dimensional bilden die $L_i^{(n)}$, $i=0,\dots,n$ eine Basis.

Lösung der Interpolationsaufgabe:

Für gegebene (x_i, y_i), $i=0,\dots,n$, erfüllt

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

die Interpolationsaufgabe: $p(x_k) = \sum_{i=0}^{n} y_i L_i^{(n)}(x_k) = \sum_{i=0}^{n} y_i \delta_{ik} = y_k$.

Satz 5.2 (Eindeutigkeit der Polynominterpolation)

Zu gegebenen paarweise verschiedenen Stützstellen $x_0, \dots, x_n$ gibt es genau ein Polynom vom Grad n mit

$$p(x_i) = y_i \qquad i = 0, \dots, n, \quad y_i \in \mathbb{R}.$$

Beweis!

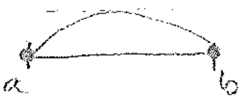
kurz: $p = \sum_{i=0}^{n} y_i L_i^{(n)}$ interpoliert die gegebenen Werte. Da die $L_i^{(n)}$ eine Basis von P_n bilden ist diese Darstellung eindeutig.

lang: $p = \sum_{i=0}^{n} y_i L_i^{(n)}$ zeigt konstruktiv die Existenz des Interpolationspolynoms

Eindeutigkeit: Ang. es gibt zwei Polynome p_1, p_2, $p_1 \neq p_2$, aber $p_1(x_i) = p_2(x_i) = y_i \quad i = 0, \dots, n$.

$p := p_1 - p_2 \in P_n$ (Differenzpolynom) erfüllt $p(x_i) = 0$, $i = 0, \dots, n$, hat also $n+1$ Nullstellen, damit muss $p \equiv 0$ gelten. Dies folgt aus

Satz von Rolle: $u(x)$ sei auf $[a,b]$ stetig und in (a,b) differenzierbar sowie $u(a) = u(b) = 0$. Dann existiert mindestens ein $x \in (a,b)$ mit $u'(x) = 0$ ($u(x) = 0$ ist auch möglich).



Angenommen es sei p nicht das Nullpolynom, also $p = \alpha_m x^m$ mit $0 \leq m \leq n$, $\alpha_m \neq 0$. Dann gilt

$p = p^{(0)} = \alpha_m x^m + \dots$ hat $n+1$ Nullstellen

$p' = p^{(1)} = \alpha_m m x^{m-1} + \dots$ hat n Nullstellen (Rolle!)

$p'' = p^{(2)} = \alpha_m m (m-1) x^{m-2} + \dots$ hat $n-1$ Nullstellen

$\vdots \quad p^{(m)} = \alpha_m m!$ hat $n+1-m \geq 1$ Nullstellen (da $m \leq n$)

Dies geht nur für $\alpha_m = 0$ im Widerspruch zur Annahme. ☒

Alternative Existenz (Rannacher Skript).

- Eindeutigkeit wie oben gezeigt.
- Ex: I-Polynom wird über LGS $Va = y$ für Koeffiz. bestimmt. z.z. ist, dass V regulär. Ang. V ist nicht regulär. Dann gibt es a, a' so dass

Wähle $z = 0$!

$Va = Va' = z$, d.h. die zugeh. Polynome $\sum a_i x^i$ bzw $\sum a'_i x^i$ interpolieren dieselbe Wertetabelle. Dies ist ein ↯ zur Eindeutigkeit.

Newton-Darstellung

Nachteil der Lagrange-Polynome: !

- Hinzufügen einer Stützstelle ändert alle bisherigen Basispolynome
- Eignet sich also nicht zur „inkrementellen" Erstellung des Interpolationspolynoms.

Besser ist hier die Newton-Darstellung mit den Basis-Polynomen:

$$N_0(x) := 1; \qquad i = 1, \dots, n: \quad N_i(x) = \prod_{j=0}^{i-1} (x - x_j)$$

a) $N_i(x)$ ist ein Polynom vom Grad i

b) $N_0, \dots, N_n$ bilden eine Basis von P_n.

c) $N_i(x_k) = 0$ für alle $k < i$, denn $(x - x_k)$ kommt in N_i für $k < i$ vor.

Gestaffelte Berechnung: Bestimme $p(x) = \sum_{i=0}^{n} a_i N_i(x)$ über

$$p(x_k) = \sum_{i=0}^{n} a_i N_i(x_k) = \sum_{i=0}^{k} a_i N_i(x_k) \overset{!}{=} y_i \qquad i = 0, \dots, n$$

also

$$a_0 = y_0;$$

$$a_k = \left(y_k - \sum_{i=0}^{k-1} a_i N_i(x_k) \right) \Big/ N_k(x_k);$$

Alternativ lassen sich die Koeffizienten auch berechnen über

Satz 5.3 (Dividierte Differenzen)

Man definiert rekursiv die sog. „Dividierten Differenzen"

$$\forall i = 0, \dots, n \qquad y[x_i] := y_i \quad \text{(die } n+1 \text{ Werte an Stützstellen)},$$

$$\forall k = 1, \dots, n-i \qquad y[\underbrace{x_i, \dots, x_{i+k}}_{k+1 \text{ Stück}}] := \frac{y[\overbrace{x_{i+1}, \dots, x_{i+k}}^{k}] - y[\overbrace{x_i, \dots, x_{i+k-1}}^{k}]}{x_{i+k} - x_i}$$

Dann gilt

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

Beweis: [Rannacher, Satz 2.2]. ▢

Die dividierten Differenzen ordnet man in folgendem Tableau an:

$$\begin{tikzcd}[column sep=small]
 & a_0 \arrow[d, phantom, "\parallel"] & a_1 \arrow[d, phantom, "\parallel"] & a_2 \arrow[d, phantom, "\parallel"] & a_3 \arrow[d, phantom, "\parallel"] \\
y_0 = & y[x_0] \arrow[r] & y[x_0,x_1] \arrow[r] & y[x_0,x_1,x_2] \arrow[r] & y[x_0,x_1,x_2,x_3] \\
y_1 = & y[x_1] \arrow[r] \arrow[ur] & y[x_1,x_2] \arrow[r] \arrow[ur] & y[x_1,x_2,x_3] \arrow[ur] & \\
y_2 = & y[x_2] \arrow[r] \arrow[ur] & y[x_2,x_3] \arrow[ur] & & \\
y_3 = & y[x_3] \arrow[ur] & & &
\end{tikzcd}$$

Hinzufügen von y_4 erfordert Berechnung aller Koeffizienten in der Diagonale.

Diese Berechnung der Koeffizienten der Newton-Darstellung ist numerisch stabiler.

Effiziente und numerisch stabile Auswertung des Interpolationspolynoms an einzelnen Stellen (also $p(x)$) gelingt mit der Methode von Neville. Siehe [Rannacher, S. 27].

Interpolationsfehler

$y_i = f(x_i)$, $i=0,\dots,n$ sei die Auswertung einer Funktion f an $n+1$ paarweise verschiedenen Stützstellen

$p(x)$ sei das Polynom vom Grad n welches (x_i, y_i), $i=0,\dots,n$ interpoliert.

Die Differenz erfüllt

$$e(x) = f(x) - p(x) = \begin{cases} 0 & x = x_i \\ ? & x \neq x_i \quad i=0,\dots,n. \end{cases}$$

Frage: Wie groß kann die Differenz werden?

Satz 5.4 Sei $f(x)$ $n+1$ mal stetig differenzierbar auf $[a,b]$ und es sei $a \leq x_0 < x_1 < \dots < x_n \leq b$. (lässt auch Extrapolation zu) Dann gibt es zu jedem $x \in [a,b]$ ein $\xi_x \in \overline{(x_0, \dots, x_n, x)}$ (= kleinstes Intervall welches alle Punkte enthält) sodass

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

Beweis. Für $x \in \{x_0, \dots, x_n\}$ liefert $\prod_{j=0} (x - x_j) = 0$ somit ist ξ_x beliebig wählbar. Sei also $x \in [a,b] \setminus \{x_0, \dots, x_n\}$. Zu diesem x definiere die Funktion

$$\underset{\substack{\uparrow \\ \text{neue freie} \\ \text{Variable}}}{F_x(t)} = f(t) - p(t) - \underbrace{\frac{f(x) - p(x)}{l(x)}}_{\substack{\text{eine Zahl} \\ \text{abhängig von } x}} l(t) \quad \text{mit } l(t) = \prod_{j=0}^{n} (t - x_j)$$

$F_x(t)$ hat die $n+2$ Nullstellen $\{x_0, \dots, x_n, x\}$, denn

$$i=0,\dots,n: \quad F_x(x_i) = \underbrace{f(x_i) - p(x_i)}_{=0} - \frac{f(x)-p(x)}{l(x)} \underbrace{l(x_i)}_{=0}$$

$$x: \quad F_x(x) = f(x) - p(x) - \frac{f(x)-p(x)}{l(x)} l(x) = 0$$

Satz von Rolle: (siehe oben)

$F_x(t)$ $n+2$ Nullstellen (mindestens)

nach t ableiten! $F_x^{(1)}(t)$ $n+1$ Nullstellen – " –

$F_x^{(n+1)}(t)$ 1 Nullstelle (– " –)

Zusätzlich gilt: Diese Nullstellen sind in $\overline{(x_0, \dots, x_n, x)}$

Diese Nullstelle von $F_x^{(n+1)}$ sei $\xi_x \in \overline{(x_0, \dots, x_n, x)}$ und es gilt

$$F_x^{(n+1)}(\xi_x) = f^{(n+1)}(\xi_x) - \underbrace{p^{(n+1)}(\xi_x)}_{\substack{=0 \text{ da } p \\ \text{Grad } n}} - \frac{f(x)-p(x)}{\ell(x)} \underbrace{\ell^{(n+1)}(\xi_x)}_{}$$

$\ell(t) = t^{n+1} + \dots$

$n+1$ mal differenzieren:

$\ell^{(n+1)}(t) = (n+1)!$

$$= f^{(n+1)}(\xi_x) - \frac{f(x)-p(x)}{\ell(x)} (n+1)! \overset{!}{=} 0.$$

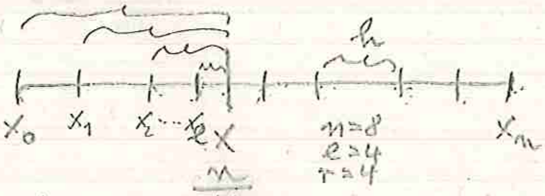
Auflösen nach $f(x)-p(x)$ liefert die Behauptung. $\boxtimes$

<u>Diskussion des Interpolationsfehlers</u>

Betrag bilden:

$$|f(x)-p(x)| \leq \frac{|f^{(n+1)}(\xi_x)|}{n+1!} \prod_{j=0}^{n} |x-x_j| \quad \text{für ein } \xi_x.$$

- Die Stützstellen seien äquidistant gewählt: $|x_i - x_{i+1}| = h$ und $x_0 \leq x \leq x_n$:



ℓ: # Punkte links von x.
r: # Punkte rechts von x
$\Rightarrow \ell + r = n+1$ (Gesamtzahl der Pkte!)

und damit $\displaystyle\prod_{j=0}^{n} |x-x_j| \leq \underbrace{1h \cdot 2h \cdot \ldots \cdot \ell h}_{\text{links von } x} \cdot \underbrace{1h \cdot 2h \cdot \ldots \cdot rh}_{\text{rechts von } x}$

$$= \ell!\, r!\, h^{n+1}$$

also $\displaystyle |f(x)-p(x)| \leq |f^{(n+1)}(\xi_x)| \underbrace{\frac{\ell!\, r!}{n+1!}}_{\leq 1 \text{ da } \ell + r = n+1} h^{n+1}$

Für $|f^{(n+1)}|$ beschränkt und $n \to \infty$ geht $|f(x)-p(x)| \to 0$.

Allerdings sind die höheren Ableitungen auch einfacher Funktionen für $n \to \infty$ oft <u>nicht</u> beschränkt sondern wachsen sehr schnell.

- Raumacher: (Runges Gegenbeispiel)
$$f(x) = \frac{1}{1+x^2} \qquad |f^{(n)}(x)| \approx 2^n n!\, O\left(\frac{1}{|x|^{n+2}}\right)$$
$\Rightarrow$ „Siehe Folien", insbesondere am Rand des Intervalls.

- ~~Beispiel 4.6. aus dem Blumstock-Skript als Ü-Aufgabe!~~

Bemerkung 5.5

Approximationssatz von Weierstraß: Jede Funktion $f \in C[a,b]$ kann beliebig gut gleichmäßig auf $[a,b]$ durch Polynome approximiert werden.

Obige Beobachtung ist kein Widerspruch, denn

-- Approximation muss nicht durch Interpolation erfolgen (Der Beweis nutzt sog. Bernstein-Polynome).

-- Mit nichtäquidistanten Stützstellen wird es auch schon sehr viel besser.

Bemerkung 5.6

Allgemein gilt für „Methoden hoher (Polynom-) Ordnung", dass entsprechende Differenzierbarkeit von f vorliegen muss.

Konditionierung:

$p(x;y)$ bezeichne das Interpolationspolynom zu den Ordinatenwerten $y = (y_0, \dots, y_n)^T$ und festen Abszissenwerte $(x_0, \dots, x_n)^T$.

$$\frac{p(x; y+\Delta y) - p(x;y)}{p(x;y)} = \left(\sum_{i=0}^{n} (y_i + \Delta y_i) L_i^{(n)}(x) - \sum_{i=0}^{n} y_i L_i^{(n)}(x) \right) \Big/ p(x;y)$$

$$= \sum_{i=0}^{n} \underbrace{\frac{L_i^{(n)}(x)\, y_i}{p(x;y)}}_{\text{Verstärk.-faktor}} \cdot \underbrace{\frac{\Delta y_i}{y_i}}_{\text{relativer Eingabefehler}}$$

Für große n kann $L_i^{(n)}(x)$ sehr groß ~~wachsen~~ werden, dann ist die Interpolationsaufgabe schlecht konditioniert!

5.3 Anwendungen der Polynominterpolation

Numerische Differentiation

Problem:

Berechne die Ableitung (der Ordnung k) einer tabellarisch gegebenen Funktion oder einer ~~einer~~ im Rechner als Prozedur gegebenen Funktion.

Idee: Erstelle Interpolationspolynom zu bestimmten Stützstellen, leite dieses ab und werte es aus.

Zunächst: Ableitungsordnung = Polynomgrad.

Lagrange-Polynome sind

$$L_i^{(n)}(x) = \prod_{\substack{j=0\\ j\neq i}}^{n} \frac{(x-x_j)}{(x_i-x_j)} = \underbrace{\left(\prod_{\substack{j=0\\ j\neq i}}^{n} \frac{1}{(x_i-x_j)}\right)}_{=:\lambda_i \in \mathbb{R}} x^n + \alpha_{n-1} x^{n-1} + \ldots + \alpha_0$$

n-maliges differenzieren liefert:

$$\frac{d^n}{dx^n} L_i^{(n)}(x) = \lambda_i\, n! \qquad \text{(konstant, unabhängig von } x\text{)}$$

Damit gilt für die n-te Ableitung eines Interpolationspolynoms vom Grad n:

$$\frac{d^n}{dx^n}\left(\sum_{j=0}^{n} y_i L_i^{(n)}\right)(x) = n! \sum_{j=0}^{n} y_i \lambda_i . \qquad \text{(unabh. von } x\text{)}$$

Eine Aussage über den Fehler liefert:

Satz 5.7 Sei $f \in C^n[a,b]$ und $a = x_0 < \ldots < x_n = b$. Dann gibt es ein $\xi \in (a,b)$ sodass

$$f^{(n)}(\xi) = n! \sum_{i=0}^{n} y_i \lambda_i .$$

Beweisskizze: n-malige Anwendung des Satzes von Rolle auf $g(x) = f(x) - p(x)$.

$g(x) = f(x) - p(x)$ $n+1$ Nullst.
$g'(x)$ n.
$g^{(n)}(x)$ 1 Nullst $\to \xi$

Um einfachere Formeln zu bekommen verwenden wir nun <u>äquidistante</u> Stützstellen, d.h. $x_i = x_0 + ih$, $0 \le i \le n$.

Damit ist

$$\lambda_i = \frac{1}{\underbrace{\underbrace{(x_i - x_0)}_{ih} \cdots \underbrace{(x_i - x_{i-1})}_{h}}_{i \text{ Stück positiv}} \underbrace{\underbrace{(x_i - x_{i+1})}_{-h} \cdots \underbrace{(x_i - x_n)}_{x_0 + ih - x_0 - nh = -(n-i)h}}_{n-i \text{ Stück negativ}}}$$

$$= \frac{1}{h^n (-1)^{n-i}\, i!\,(n-i)!} = \frac{(-1)^{n-i}}{h^n\, n!} \binom{n}{i}$$

← Binomialkoeffizient $\frac{n!}{i!\,(n-i)!}$

und damit

$$f^{(n)}(x) \approx \frac{d^n}{dx^n}\left(\sum_{j=0}^{n} y_i L_i^{(n)}(x)\right) = \frac{1}{h^n} \sum_{i=0}^{n} (-1)^{n-i} \binom{n}{i} y_i .$$

$n!$ kürzt sich heraus!

Speziell:

$$f^{(1)}(x) \approx \frac{y_1 - y_0}{h}, \quad f^{(2)}(x) \approx \frac{y_2 - 2y_1 + y_0}{h^2}, \quad f^{(3)}(x) \approx \frac{y_3 - 3y_2 + 3y_1 - y_0}{h^3}.$$

Bisher: n-te Ableitung aus Polynom vom Grad n (d.h. $n+1$ Werten).

Es geht auch: m-te Ableitung aus Polynom vom Grad $n > m$.
Wert hängt dann aber von der Auswertestelle ab.

Beispiel: $m = 1$, $n = 2$. Äquidistante Stützstellen x_0, $x_1 = x_0 + h$, $x_2 = x_0 + 2h$.

$$p'(x_1) = \frac{y_2 - y_0}{2h} \quad \text{„zentraler Differenzenquotient"}$$

Taylorreihenentwicklung (Übung!) zeigt

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2) \quad \text{für } f \in C^3,$$

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2) \quad \text{für } f \in C^4.$$

<u>Beispiel 5.8</u> → Beispiel 4.9 aus Numstoch zur Auslöschung bei numerischer Differentiation und zur Motivation der Extrapolation!

→ Übung! ?

Extrapolation zum Limes

Eine Größe $a(h)$ sei im Rechner für $h > 0$ berechenbar, nicht jedoch für $h = 0$. Man möchte

$$a(0) = \lim_{h \to 0} a(h)$$

mit guter Genauigkeit berechnen.

Beispiel 5.9

a) $$a(0) = \lim_{x \to +0} \frac{\cos(x) - 1}{\sin(x)} \qquad (=0)$$

b) Numerische Differentiation

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$ (für kleine h tritt ~~Auslösg~~ Auslöschung ein)

c) Numerische Integration

$$\int_a^b f(x) = \lim_{N \to \infty} \sum_{i=1}^{N} \frac{1}{N} f\left(\frac{i-1}{2N} + \frac{i}{2N}\right) = \lim_{N \to \infty} \sum_{i=1}^{N} h f\left(\frac{2i-1}{2} h\right)$$

wobei $h := \frac{1}{N}$.

$h \to 0$ nicht möglich wg. Aufwand $\to \infty$

d) Numerische Lösg des Anfangswertproblems

$$y'(t) = f(t, y(t)); \quad y(0) = y_0$$

$y(T) \approx y_N$ und $y_n = y_{n-1} + h f(t, y_{n-1})$; $h = 1/N$;

$h \to 0$ bedeutet $N \to \infty$ und damit Aufwand $\to \infty$.

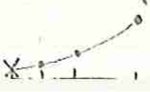
Idee der Extrapolation:

Zu $h_0 > h_1 > \dots > h_n > 0$ bestimme Interpolationspolynom

$$p(h_i) = a(h_i) \qquad i = 0, \dots, n$$

und berechne

$$a(0) \approx p(0) \qquad (\text{Extrapolation, da } 0 \notin [h_n, \dots, h_0]$$

graphisch: 

Beispiel 5.10.

[Rannacher]. Für $a(h) = \dfrac{\cos(h)-1}{\sin(h)}$ erhält man für

$$\left.\begin{aligned} h_0 &= 1/8 & a(h_0) &= -6{,}258151\cdot 10^{-2} \\ h_1 &= 1/16 & a(h_1) &= -3{,}126018\cdot 10^{-2} \\ h_2 &= 1/32 & a(h_2) &= -1{,}562627\cdot 10^{-2} \end{aligned}\right\}$$ halbiert sich, da h halbiert wird.

und bei Extrapolation mit einem Polynom vom Grad 2:

$$a(0) \approx p_2(0) = -1{,}02\cdot 10^{-5}$$

also sehr viel besser! $\boxtimes$

Übung: Extrapolation bei der numerischen Differentiation!

Warum funktioniert das so gut?

Die Funktion $a(x)$ sei $n+1$ mal stetig differenzierbar in einer genügend großen Umgebung von 0. Dann gibt es zu jedem $h>0$ (in dieser Umgebung) ein $\xi_h \in [0,h]$ so dass: (Taylorreihe mit Restglied n. Lagrange)

$$\underbrace{a(h)}_{\text{Für } h>0 \text{ numerisch berechenbar.}} = a(0+h) = a(0) + h\,a'(0) + \dots + \frac{h^n}{n!}a^{(n)}(0) + \frac{h^{n+1}}{(n+1)!}a^{(n+1)}(\xi_h)$$

gleich dorthinschreiben!

$$= \underbrace{\underbrace{a(0)}_{=:a_0} + \underbrace{a'(0)}_{=:a_1} h + \dots + \underbrace{\frac{a^{(n)}(0)}{n!}}_{=:a_n} h^n}_{\text{Polynom in } h \text{ vom Grad } n} + \underbrace{\frac{a^{(n+1)}(\xi_h)}{(n+1)!}}_{\text{abhängig von } h!} h^{n+1} \qquad (5.1)$$

$a^{(k)}(0)$ hängt nicht von h ab!

Idee: Für verschiedene h_i bilde Linearkombination:

$$\sum_{i=0}^n c_i\, a(h_i) = \sum_{i=0}^n c_i \left(\sum_{j=0}^n a_j h_i^j\right) + \sum_{i=0}^n c_i \underbrace{\frac{a^{(n+1)}(\xi_{h_i})}{(n+1)!}}_{=:\alpha_i} h_i^{n+1}$$

$$= \sum_{j=0}^n \underbrace{a_j}_{\text{unbekannt}} \underbrace{\left(\sum_{i=0}^n c_i h_i^j\right)}_{\text{Bestimmungsgleichg für die } c_i} + \text{Fehler} = a_0 + \underbrace{\text{Fehler}}_{\to \text{ klein wg } h_i^{n+1}}$$

$$\sum_{i=0}^n c_i h_i^j = \begin{cases} 1 & j=0 \\ 0 & \text{sonst} \end{cases}$$

Als Gleichungssystem lauten die Bestimmungsgleichungen

$$V^T c = e^{(0)}$$

mit $e^{(0)} = (1, 0, \dots, 0)^T$

und $V = V[h_0, \dots, h_n]$ der Vandermondematrix.

Die Auswertung mit diesen Koeffizienten ist dann:

$$A := \sum_{i=0}^{n} c_i \underbrace{a(h_i)}_{y_i} \underset{\substack{\uparrow \\ y=(y_0,\dots,y_n)}}{=} (\underbrace{V^{-T} e^{(0)}}_{=c})^T y = (e^{(0)})^T V^{-1} y.$$

$$V = \begin{bmatrix} 1 & h_0 & h_0^2 & \cdots & h_0^n \\ & & & & \\ 1 & h_n & \cdots & & h_n^n \end{bmatrix} \overset{h_i = h r^i}{=} \begin{bmatrix} 1 & hr & (hr)^2 & & (hr)^n \\ \vdots & & & & \\ 1 & hr^n & (hr^n)^2 & & (hr^n)^n \end{bmatrix} = \begin{bmatrix} 1 & r & r^2 & \cdots & r^n \\ 1 & r^n & r^{2n} & & r^{n \cdot n} \end{bmatrix} \begin{bmatrix} 1 & & 0 \\ & h & \\ & & \ddots \\ 0 & & h^n \end{bmatrix}$$

Für den Fehler gilt dann

$$|A - \underbrace{a(0)}_{\text{Best. gl. für } c!}| = \left| \sum_{i=0}^{n} c_i \frac{a^{(n+1)}(\xi_i)}{(n+1)!} h_i^{(n+1)} \right|$$

$$h_i = h r^i \rightarrow \leq \sum_{i=0}^{n} \overbrace{\|V^{-T} e^{(0)}\|_\infty}^{\max_i c_i} \frac{|a^{(n+1)}(\xi_i)|}{(n+1)!} h^{n+1} r^{i(n+1)}$$

z.B. $r = 1/2$,

$$\leq \|V^{-T}\|_\infty \underbrace{\|e^{(0)}\|_\infty}_{=1} |a^{(n+1)}(\xi_{i_{max}})| \frac{h^{n+1}}{(n+1)!} \underbrace{\sum_{i=0}^{n} r^{i(n+1)}}_{\leq r^{n+1}+1}$$ geometrische Reihe

$$\leq \|V^{-T}\|_\infty \underbrace{|a^{(n+1)}(\xi_{i_{max}})|}_{\text{beschränkt}} \frac{h^{n+1}}{(n+1)!} \underbrace{(1 + r^{n+1})}_{\leq \frac{5}{4} \text{ für } r \leq \frac{1}{2} \wedge n \geq 1}$$

Nun betrachte das Interpolationspolynom

$$p(h_i) = \sum_{j=0}^{n} b_j h_i^j \overset{!}{=} a(h_i) = y_i$$

und somit $Vb = y$ für dessen Koeffizienten.

Auswerten an der Stelle Null bedeutet

$$p(0) = b_0 = (e^{(0)})^T b = (e^{(0)})^T V^{-1} y = A \text{ von oben!}$$

Oben beschriebene Elimination der Fehlerterme entspricht also genau der Extrapolationsmethode!

Entscheidend ist also die Fehlerdarstellung (5.1).

Und es gilt noch besser! Wir betrachten die Näherung für $f''(x)$:
(n gerade:)

$$f(x+h) = f(x) + hf^{(1)}(x) + \frac{h^2}{2} f''(x) + \frac{h^3}{3!} f^{(3)}(x) + \dots + \frac{h^{2n+2}}{(2n+2)!} f^{(2n+2)}(x) + \frac{h^{2n+4}}{(2n+4)!} f^{(2n+4)}(\xi_+)$$

$$f(x-h) = f(x) - hf^{(1)}(x) + \frac{h^2}{2} f''(x) - \frac{h^3}{3!} f^{(3)}(x) + \dots + \frac{h^{2n+2}}{(2n+2)!} f^{(2n+2)}(x) + \frac{h^{2n+4}}{(2n+4)!} f^{(2n+4)}(\xi_-)$$

$$f(x+h) + f(x-h) = 2f(x) + h^2 f''(x) + \dots + \frac{h^{2n+2}}{2\,(2n+2)!} f^{(2n+2)}(x) + \frac{h^{2n+4}}{(2n+4)!}\left[f^{(2n+4)}(\xi_+) + f^{(2n+4)}(\xi_-)\right]$$

$$g(h) := \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} =$$

$$= f''(x) + \frac{h^2}{2(4!)} f^{(4)}(x) + \dots + \frac{h^{2n}}{2\,(2n+2)!} f^{(2n+2)}(x) + \frac{h^{2n+2}}{(2n+4)!}\left[f^{(2n+4)}(\xi_+) + f^{(2n+4)}(\xi_-)\right]$$

$$= p_x(h^2) + O(h^{2n+2})$$

Da alle ungeraden Terme „von selber" wegfallen kann man mit der gleichen Anzahl von Auswertungen doppelt soviele Fehlerterme eliminieren!

Natürlich muss f dazu entsprechend oft differenzierbar sein.