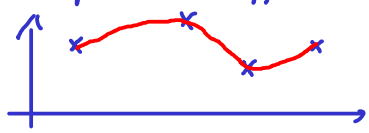
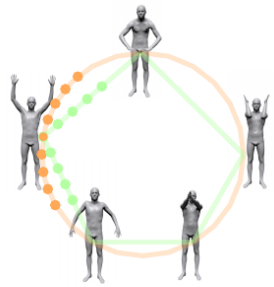
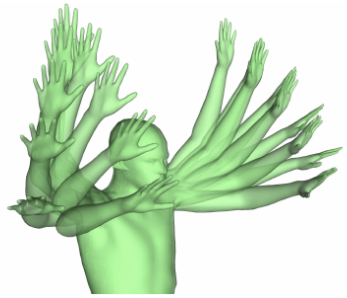
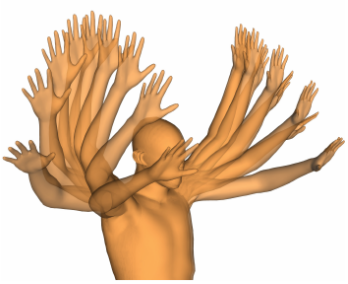
Numerische Analysis

Themen: · Interpolation / Approximation



⇓ Grundlage für

· Numerische Integration / Differentiation

z.B. Keplersche Fassregel zur Bestimmung des Weinfass-Volumens anhand weniger Messwerte (1615)

oder Basis von FEM-Simulationen (google Bildersuche: Finite Elemente)

⇓ Grundlage für

· Numerik gewöhnlicher Differentialgleichungen

z.B. „Mehrkörperproblem": Berechnung / Vorhersage von Planetenpositionen (nicht analytisch möglich)

⇓ Grundlage für

· kurzer Einblick in Numerik partieller Differentialgleichungen

Polynome & rationale Funktionen

heute: Polynome

Polynome

Polynome sind einfache Funktionen, erlauben die Approximation von komplizierten Funktionen (Taylorscher Satz) und lassen sich mit nur $+, -, \cdot$ auswerten ⇒ wichtige Numerik-Bausteine

Def: (Polynom) Sei $\mathbb{K} = \mathbb{R}$ oder $\mathbb{K} = \mathbb{C}$. Ein Polynom n-ten Grades ist eine Funktion

$p_n : \mathbb{K} \to \mathbb{K}$, $p_n(x) = \sum_{i=0}^{n} a_i x^i$ mit $a_0, \dots, a_n \in \mathbb{K}$, $a_n \neq 0$.

$P_n := \{ p \mid p \text{ ist Polynom mit Grad} \leq n \}$

Thm.: 1) P_n ist ein $\mathbb{K}$-Vektorraum (mit $(\alpha p_n + \beta q_n)(x) = \alpha p_n(x) + \beta q_n(x)$ $\forall \alpha, \beta \in \mathbb{K}, p_n, q_n \in P_n$)

2) Die Monome $m_i(x) = x^i$, $i = 0, \dots, n$, bilden eine Basis von P_n.

Bew.: 1) „einfach ausrechnen"

2) Sei $\varphi(x) = a_0 + a_1 x + \dots + a_n x^n = 0$ $\forall x$. Sei k minimal mit $a_k \neq 0$.

⇒ $\frac{\varphi(x)}{x^k} = a_k + a_{k+1} x + \dots + a_n x^{n-k} = 0$ $\forall x \neq 0$, aber auch $\lim_{x \to 0} \frac{\varphi(x)}{x^k} = a_k \neq 0$ ↯

$\Rightarrow m_i$ sind linear unabhängig & spannen P_n auf. □

Thm.: (Polynomdivision) Seien $p, q \neq 0$ Polynome, Grad $q \leq$ Grad p, so gibt es eindeutige Polynome s, r mit $p = sq + r$, Grad $r <$ Grad q (Division mit Rest).

Bew.: „Euklidscher Divisionsalgorithmus": Sei $p(x) = \sum_{i=0}^{n} a_i x^i$, $q(x) = \sum_{i=0}^{m} b_i x^i$, $s(x) = \sum_{i=0}^{n-m} c_i x^i$, $r(x) = \sum_{i=0}^{m-1} d_i x^i$

Koeffizientenvergleich in $p = sq + r$ liefert

$$\left.\begin{aligned}
x^n:\ & a_n = b_m c_{n-m} \\
x^{n-1}:\ & a_{n-1} = b_{m-1} c_{n-m} + b_m c_{n-m-1} \\
& \vdots \\
x^m:\ & a_m = b_0 c_m + b_1 c_{m-1} + \dots + b_m c_0 \\
x^{m-1}:\ & a_{m-1} = b_0 c_{m-1} + \dots + b_{m-1} c_0 + d_{m-1} \\
& \vdots \\
x^0:\ & a_0 = b_0 c_0 + d_0
\end{aligned}\right\}$$

eindeutig lösbar nach c_i & d_i:

(von oben der Reihe nach $c_{n-m}, c_{n-m-1}, \dots, c_0$, $d_{m-1}, d_{m-2}, \dots, d_0$) □

Bsp: $(\overset{a_3}{1}x^3 - \overset{a_2}{9}x^2 + 26x - 24) : (\overset{b_2}{1}x^2 - \overset{b_1}{7}x + 10) = \overset{c_1}{1}x \overset{c_0}{-2}$ ① ② ③

$$\frac{x^3 - 7x^2 + 10x}{-2x^2 + 16x - 24}$$

$a_2 - c_1 b_1 = c_0 b_2$

$$\frac{-2x^2 + 14x - 20}{2x - 4} \leftarrow \text{Rest}$$

Bem: Mit dem Euklidschen Divisionsalgorithmus kann man auch einen größten gemeinsamen Teiler von p_1, p_2 mit Grad $p_2 \leq$ Grad p_1 bestimmen:

Löse $p_1 = q_1 p_2 + p_3$ Grad $p_3 <$ Grad p_2

$p_2 = q_2 p_3 + p_4$ Grad $p_4 <$ Grad p_3

$\vdots$

$p_\ell = q_\ell p_{\ell+1} + p_{\ell+2}$ $\quad p_{\ell+2} = 0$

Dies terminiert nach endlich vielen Schritten, da der Grad monoton abnimmt.

$p_{\ell+1}$ teilt alle Polynome $p_1, \dots, p_\ell$ (sieht durch Einsetzen von unten) und ist ein ggT von p_1, p_2.

Bsp: $\overbrace{(x^3 - 9x^2 + 26x - 24)}^{p_1} = (x - 2)\overbrace{(x^2 - 7x + 10)}^{p_2} + (2x - 4)$

$(x^2 - 7x + 10) = \left(\frac{x}{2} - \frac{5}{2}\right)(2x - 4) + 0$

$\Rightarrow 2x - 4$ ist ggT

In der Numerik an verschiedenen Stellen wichtig sind folgende Polynome.

Def: (Čebyšev-Polynome) Die Čebyšev-Polynome sind rekursiv definiert durch

1. Art:
$T_0(x) = 1$
$T_1(x) = x$
$T_{n+1}(x) = 2x T_n(x) - T_{n-1}(x)$, $n = 1, 2, \dots$

2. Art:
$U_0(x) = 1$
$U_1(x) = 2x$
$U_{n+1}(x) = 2x U_n(x) - U_{n-1}(x)$, $n = 1, 2, \dots$

Bem: T_n hat höchsten Koeffizienten 2^{n-1}, U_n hat 2^n.

Thm: Auf $\mathbb{R}$ gilt $T_n(x) = \begin{cases} \cos(n \arccos x), & x \in [-1,1] \\ \cosh(n \operatorname{arccosh} x), & x \geq 1 \\ (-1)^n \cosh(n \operatorname{arccosh}(-x)), & x \leq 1 \end{cases}$, $U_n(x) = \frac{\sin((n+1)\arccos x)}{\sin \arccos x}$ auf $[-1,1]$

Bew: T_n) Additionstheoremen: $\left.\begin{aligned} \cos(n+1)t &= \cos nt \cos t - \sin nt \sin t \\ \cos(n-1)t &= \cos nt \cos t + \sin nt \sin t \end{aligned}\right\} \Rightarrow \cos(n+1)t + \cos(n-1)t = 2 \cos t \cos nt$

$\Rightarrow \varphi_n(t) = \cos nt$ erfüllt $\varphi_0(t) = 1$
$\varphi_1(t) = \cos t$
$\varphi_{n+1}(t) = 2 \cos t \, \varphi_n(t) - \varphi_{n-1}(t)$, $n = 1, 2, \ldots$

$\Rightarrow \Psi_n := \varphi_n \circ \arccos$ erfüllt gleiche Rekursion wie T_n & ist definiert auf $[-1,1]$.

Fall $x \notin [-1,1]$: Hausaufgabe

U_n) $\left.\begin{aligned} \sin(n+1)t &= \sin nt \cos t + \cos nt \sin t \\ \sin(n-1)t &= \sin nt \cos t - \cos nt \sin t \end{aligned}\right\} \Rightarrow \sin(n+1)t + \sin(n-1)t = 2 \sin nt \cos t$

$\Rightarrow \Psi_n = \varphi_n \circ \arccos$ für $\varphi_n(t) = \frac{\sin(n+1)t}{\sin t}$ erfüllt gleiche Rekursion wie U_n. □

Thm: 1) $T_n'(x) = n U_{n-1}(x)$

2) $|T_n(x)| \leq 1$ für $x \in [-1,1]$ und $|T_n(x)| > 1$ sonst

3) T_n hat auf $[-1,1]$ Extrema an $x_k^{(n)} = \cos\left(k \frac{\pi}{n}\right)$, $k = 0, \ldots, n$, mit $T_n(x_k^{(n)}) = (-1)^k$

4) T_n hat einfache Nullstellen $\tilde{x}_k^{(n)} = \cos\left(\frac{2k-1}{2} \frac{\pi}{n}\right)$, $k = 1, \ldots, n$

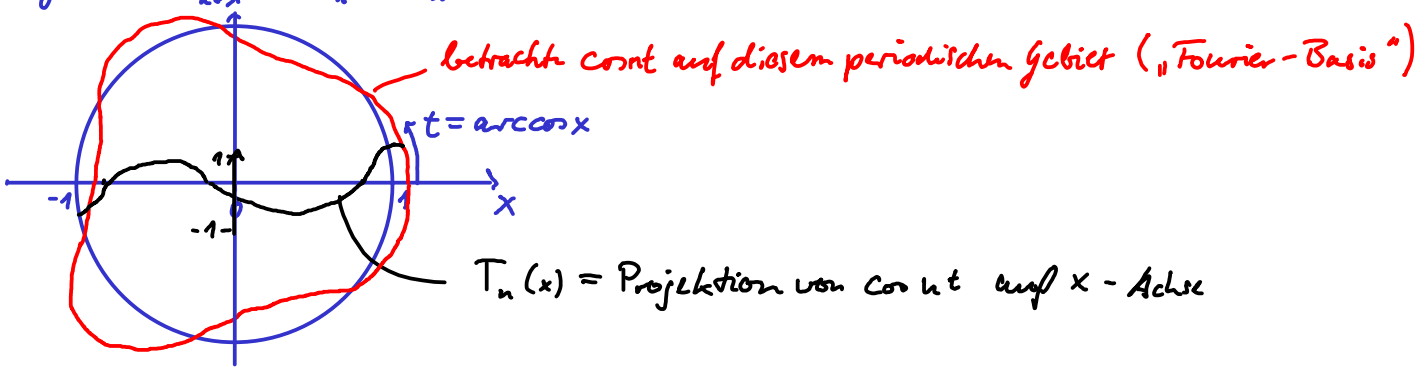
5) Zwischen je zwei Nullstellen von T_{n+1} liegt eine von T_n

Bew: 1) Ausrechnen in trigonometrischer Darstellung

2) folgt aus $|\cos t| \leq 1$, $\cosh t \geq 1$

3) & 4) folgt durch Einsetzen der $x_k^{(n)}$, $\tilde{x}_k^{(n)}$ in T_n & T_n'

5) folgt aus $\tilde{x}_{k+1}^{(n+1)} < \tilde{x}_k^{(n)} < x_k^{(n+1)}$ □




```
x = linspace(-1,1,100);
for n = 0:5
  Tn = cos(n*acos(x));
  plot(x,Tn,'Linewidth',3); hold on;
end
```

Ebenso wichtig sind die Bernstein-Polynome.

Def: (Bernstein-Polynom) $b_{in}(t) = \binom{n}{i} t^i (1-t)^{n-i}$, $i = 0, \ldots, n$, heißt Bernstein-Polynom vom Grad n.

Thm: (Eigenschaften) (1) $b_{i,n}$ hat eine i-fache Nullstelle in 0 und eine $(n-i)$-fache in 1.

(2) $b_{i,n} > 0$ auf $(0,1)$ mit genau einem Maximum in $\frac{i}{n}$.

(3) Die $b_{i,n}$ bilden eine Partition der 1, $\sum_{i=0}^{n} b_{i,n}(t) = 1$.

(4) Die $b_{i,n}$ bilden eine Basis von P_n.

(5) $b_{i,n}'(t) = n\left[b_{i-1,n-1}(t) - b_{i,n-1}(t)\right]$ (für $b_{i,k} = 0$ falls $i<0$ oder $i>k$)

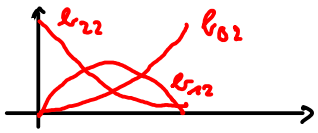
(6) $b_{i,n}^{(k)}(t) = \prod_{j=0}^{k-1}(n-j) \cdot \sum_{\ell=0}^{k} (-1)^{\ell} \binom{k}{\ell} b_{i-k+\ell,\,n-k}(t)$

Bew: (1) & (2) & (5) trivial

(3) folgt aus $1 = (t+(1-t))^n = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i}$

(4) & (6) Hausaufgabe □

Bsp: $b_{0,2}(t) = (1-t)^2$, $b_{1,2}(t) = 2t(1-t)$, $b_{2,2}(t) = t^2$



Heute: · Polynomauswertung
· Kondition & Stabilität

Polynomauswertung

Polynome müssen in numerischen Algorithmen oft ausgewertet werden. Dies kann auf verschiedene Arten gemacht werden, abhängig von der gegebenen Polynom-Darstellung.

Alg.: (Horner-Schema)

$p(x) = \sum_{i=0}^{n} a_i x^i$ kann ausgewertet werden durch $p(x) = (\dots((a_n x + a_{n-1})x + a_{n-2})x + \dots + a_1)x + a_0$

als Schema:

	a_n	a_{n-1}	a_{n-2}	...	a_1	a_0
x	–	$x\tilde{a}_n$	$x\tilde{a}_{n-1}$	...	$x\tilde{a}_2$	$x\tilde{a}_1$
	$\tilde{a}_n$	$\tilde{a}_{n-1}$	$\tilde{a}_{n-2}$	...	$\tilde{a}_1$	$\tilde{a}_0 = p(x)$

Bem: Der Euklidsche Divisionsalgorithmus für $p(x)/(x-y)$ liefert genau das Horner-Schema für $x=y$

denn $p(x) = \sum_{i=0}^{n} a_i x^i \overset{!}{=} \tilde{a}_n x^n + \sum_{i=0}^{n-1} (\tilde{a}_i - \tilde{a}_{i+1} y) x^i = \left(\sum_{i=0}^{n-1} \tilde{a}_i x^{i-1}\right)(y-x) + \tilde{a}_0$.

($\tilde{a}_i = \tilde{a}_{i+1} y + a_i$)

Bsp: $p(x) = x^3 - 9x^2 + 26x - 24$, $x = 2$:

	1	−9	26	−24
2		2	−14	24
	1	−7	12	0 $= p(x)$

$\Rightarrow \quad p(x) = (x-2)(1x^2 - 7x + 12)$

Alg.: (Clenshaw-Schema)

$p(x) = \sum_{i=0}^{n} a_i T_i(x) = \tilde{a}_0$ mit

$$\begin{cases} \tilde{a}_n = a_n \\ \tilde{a}_{n-1} = 2x\tilde{a}_n + a_{n-1} \\ \tilde{a}_{n-i} = 2x\tilde{a}_{n-i+1} + a_{n-i} - \tilde{a}_{n-i+2} & , i = 2, \dots, n-1 \\ \tilde{a}_0 = x\tilde{a}_1 + a_0 - \tilde{a}_2 \end{cases}$$

als Schema:

	a_n	a_{n-1}	a_{n-2}	...	a_2	a_1	a_0
	–	–	$-\tilde{a}_n$	...	$-\tilde{a}_4$	$-\tilde{a}_3$	$-\tilde{a}_2$
x	–	$2x\tilde{a}_n$	$2x\tilde{a}_{n-1}$	...	$2x\tilde{a}_3$	$2x\tilde{a}_2$	$x\tilde{a}_1$
	$\tilde{a}_n$	$\tilde{a}_{n-1}$	$\tilde{a}_{n-2}$	...	$\tilde{a}_2$	$\tilde{a}_1$	$\tilde{a}_0 = p(x)$

Der Algorithmus folgt aus der Rekursion für T_n:

$$p(x) = a_n T_n(x) + a_{n-1} T_{n-1}(x) + \dots + a_1 T_1(x) + a_0 T_0(x)$$

$$= (2x a_n + a_{n-1})\, T_{n-1}(x) + (-a_n + a_{n-2})\, T_{n-2}(x) + \dots \qquad (2x T_{n-1}(x) - T_{n-2}(x))$$

$$= \tilde{a}_{n-1} T_{n-1}(x) + (-\tilde{a}_n + a_{n-2})\, T_{n-2}(x) + \dots \qquad (2x T_{n-2}(x) - T_{n-3}(x))$$

$$= (2x \hat{a}_{n-1} - \hat{a}_n + a_{n-2})\, T_{n-2}(x) + (-\tilde{a}_{n-1} + a_{n-3}) T_{n-3}(x) + \dots$$

$$= \hat{a}_{n-2} T_{n-2}(x) + (-\tilde{a}_{n-1} + a_{n-3})\, T_{n-3}(x) + \dots$$

$$\vdots$$

$$= \hat{a}_1 T_1(x) + (-\hat{a}_2 + a_0)\, T_0(x)$$

$$= x \tilde{a}_1 + (-\tilde{a}_2 + a_0) = \tilde{a}_0$$

Bsp: $p(x) = x^3 - 9x^2 + 26x - 24$

$$= \tfrac{1}{4} T_3(x) - \tfrac{9}{2} T_2(x) + \tfrac{107}{4} T_1(x) - \tfrac{57}{2} T_0(x)$$

$x = 2$

	1/4	−9/2	107/4	−57/2
			−1/4	7/2
2		1	−14	−25
	1/4	−7/2	−25/2	$0 = p(2)$

Die Auswertung eines Polynoms in Bézier-Darstellung geschieht nach de Casteljaus Algorithmus.

Def: (Bézier-Darstellung) Sei $p \in \mathcal{P}_n$. Die Form $p = \sum_{i=0}^{n} \beta_i B_{in}$ heißt Bézier-Darstellung von p.

Bézier-Koeffizienten $= \beta_0, \dots, \beta_n$, Bézier-Punkte $= \begin{pmatrix} 0/n \\ \beta_0 \end{pmatrix}, \dots, \begin{pmatrix} n/n \\ \beta_n \end{pmatrix}$

Alg: (de Casteljau) $p(t) = \sum_{i=0}^{n} \beta_i B_{in}(t)$ kann rekursiv berechnet werden durch

$$\beta_i^{(0)} := \beta_i, \quad i = 0, \dots, n$$

$$\beta_i^{(k)} := (1-t)\, \beta_i^{(k-1)} + t\, \beta_{i+1}^{(k-1)}, \quad k = 1, \dots, n, \quad i = 0, \dots, n-k$$

$$p(t) = \beta_0^{(n)}$$

Übersichtlich geschieht dies im Dreiecksschema

$$\begin{array}{ccccc} \beta_0^{(0)} & \beta_0^{(1)} & \beta_0^{(2)} & \cdots & \\ \beta_1^{(0)} & \beta_1^{(1)} & \vdots & \ddots & \\ \beta_2^{(0)} & \vdots & \beta_{n-2}^{(2)} & \iddots & \beta_0^{(n)} \\ \vdots & \beta_{n-1}^{(1)} & & & \\ \beta_n^{(0)} & & & & \end{array}$$

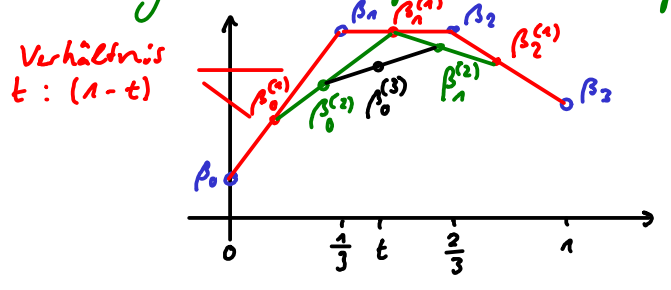
Beweis: Dies folgt aus $B_{in}(t) = \binom{n-1}{i} t^i (1-t)^n + \binom{n-1}{i-1} t^i (1-t)^n = (1-t) B_{i,n-1}(t) + t\, B_{i-1,n-1}(t)$:

$$p(t) = \beta_0 B_{0n}(t) + \beta_1 B_{1n}(t) + \dots + \beta_n B_{nn}(t) = \beta_0 \left[(1-t) B_{0,n-1}(t) + t B_{0,n-1}(t)\right] + \dots + \beta_n \left[(1-t) B_{n-1,n-1}(t) + t B_{n-1,n-1}(t)\right]$$

$$= \left[\beta_0 (1-t) + \beta_1 t\right] B_{0,n-1}(t) + \dots + \left[\beta_{n-1}(1-t) + \beta_n t\right] B_{n-1,n-1}(t) = \beta_0^{(1)} B_{0,n-1}(t) + \dots + \beta_{n-1}^{(1)} B_{n-1,n-1}(t)$$

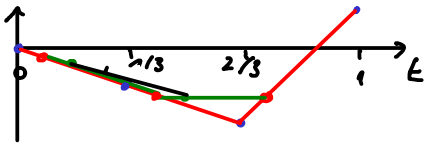
Nun iteriere dies bis zu $\beta_0^{(n)}$. □

Bem: Die Auswertung nach de Casteljau kann auch graphisch erfolgen:



Bsp: $(\beta_0, \dots, \beta_3) = (0, -1, -2, 1)$, $t = \frac{1}{4}$

$0 \quad -1/4 \quad -\frac{1}{2} \quad -\frac{11}{4} = p(t)$
$-1 \quad -5/4 \quad -5/4$
$-2 \quad -5/4$
1



Da Polynome in numerischen Algorithmen häufig ausgewertet werden müssen, ist ihre Kondition von Interesse, d.h. wie empfindlich das Ergebnis bzgl. kleiner numerischen Eingabefehlern ist.

Thm. (Markow-Ungleichung) Sei $p \in P_n$, dann ist $\max_{x \in [-1,1]} |p'(x)| \le n^2 \max_{x \in [-1,1]} |p(x)|$.

Bem: Für $p = T_n$ gilt $p'(1) = n^2$, $|p(x)| \le 1$ für $x \in [-1,1]$, somit ist die Ungleichung scharf.

Thm. (Kondition) Sei $p \in P_n$, $|p(x)| \le 1$ für $x \in [-1,1]$,

- $p = \sum_{i=0}^{n} a_i Q_i$ für $Q_0, \dots, Q_n$ eine Basis von P_n,
- $\hat{x}$ eine Näherung für x mit $|x - \hat{x}| \le \varepsilon$, $x, \hat{x} \in [-1,1]$,
- $\hat{p} = \sum_{i=0}^{n} \hat{a}_i Q_i$ mit $|a_i - \hat{a}_i| \le \delta$, $i = 0, \dots, n$.

Dann (1) $|p(x) - p(\hat{x})| \le n^2 \varepsilon + O(\varepsilon^2)$,

(2) $|p(x) - \hat{p}(x)| \le \delta \sum_{i=0}^{n} |Q_i(x)|$.

Bew: (1) $|p(x) - p(\hat{x})| = |p'(x)||x - \hat{x}| + O(|x - \hat{x}|^2) \le n^2 \max_{y \in [-1,1]} |p(y)| \, \varepsilon + O(\varepsilon^2) \le n^2 \varepsilon + O(\varepsilon^2)$

(2) $|p(x) - \hat{p}(x)| = \left| \sum_{i=0}^{n} (a_i - \hat{a}_i) Q_i(x) \right| \le \delta \sum_{i=0}^{n} |Q_i(x)|$ □

Bem:
- Für $Q_i = T_i$ oder $Q_i(x) = x^i$ oder $Q_i(x) = b_{in}(x)$ ist somit der Gesamtfehler
$$|p(x) - \hat{p}(\hat{x})| \le |p(x) - p(\hat{x})| + |p(\hat{x}) - \hat{p}(\hat{x})| \le n^2 \varepsilon + (n+1)\delta + O(\varepsilon^2).$$
- Ein Eingabefehler ε kann um den Faktor n^2 verstärkt werden! Für Polynome aus der Taylorentwicklung analytischer Funktionen (z.B. exp, sin, log(1+·), ...) schrumpfen die Koeffizienten jedoch so schnell, dass typischerweise $\max_{x \in [-1,1]} |p'(x)| \le c_n \max_{x \in [-1,1]} |p(x)|$ für $c_n \ll n^2$ – die Fehlerverstärkung n^2 ist also ein worst case.

Weiter ist von Interesse, wie groß die durch Rundung und Maschinenoperationen verursachten Fehler bei der Polynomauswertung auf dem Computer sind, also wie stabil die Auswertungsalgorithmen sind.

Thm. (Stabilität) Durch Maschinenoperationen im Horner-Schema für $p(x) = \sum_{i=0}^{n} a_i x^i$ entsteht der Fehler

$$\sum_{i=0}^{n} \hat{a}_{n-i} (\varepsilon_i + \eta_{i+1}) x^{n-i} + \text{höhere Ordnungsterme in } \varepsilon_i, \eta_i \qquad (\eta_{n+1} := 0)$$

mit $|\varepsilon_i| \le \varepsilon_m$ und $|\eta_i| \le \varepsilon_m$ den relativen Additions- und Multiplikationsfehler im i-ten Schritt.
(ε_m: Maschinengenauigkeit)

Bew.: $(\,)'$ bezeichne Maschinen-berechnete Größen.

Horner: $\tilde a_n' = a_n$, $\tilde a_{n-i}' = (\tilde a_{n-i+1}' \odot x) \oplus a_{n-i} = \big(\tilde a_{n-i+1}' \times (1+\eta_i) + a_{n-i}\big)(1+\varepsilon_i)$

$$\Rightarrow p(x)' = \tilde a_0' = \big(\tilde a_1' \times (1+\eta_n) + a_0\big)(1+\varepsilon_n) = \big((\tilde a_2' \times (1+\eta_{n-1}) + a_1)(1+\varepsilon_{n-1}) \times (1+\eta_n) + a_0\big)(1+\varepsilon_n)$$

$$= \ldots = \sum_{i=0}^{n} a_{n-i} x^{n-i} \prod_{j=i}^{n} (1+\varepsilon_j) \prod_{j=i+1}^{n} (1+\eta_j)$$

definiere $\tilde a_{n+1} = 0$

$$= \sum_{i=0}^{n} \Big(\tilde a_{n-i} - \tilde a_{n-i+1} x\Big) x^{n-i} \prod_{j=i}^{n} (1+\varepsilon_j) \prod_{j=i+1}^{n} (1+\eta_j)$$

$$= \tilde a_0 (1+\varepsilon_n) + \sum_{i=0}^{n-1} \Big(\tilde a_{n-i}(1+\varepsilon_i)(1+\eta_{i+1}) - \tilde a_{n-i}\Big) x^{n-i} \prod_{j=i+1}^{n} (1+\varepsilon_j) \prod_{j=i+2}^{n} (1+\eta_j)$$

$$= \underbrace{\tilde a_0}_{\tilde p(x)} + \tilde a_0 \varepsilon_n + \sum_{i=0}^{n-1} \tilde a_{n-i} (\varepsilon_i + \eta_{i+1}) x^{n-i} + \text{höhere Ordnungsterme} \qquad \square$$

Je größer die Zwischensummen $\tilde a_i$, desto größer der verursachte Fehler. Für $p(x) = T_n(x)$ und $x = 1$ kann man z.B. zeigen, dass $\max_{i=0,\ldots,n} \tilde a_i > \frac{(1+\sqrt{2})^n}{2(n+2)}$ $\Rightarrow$ für $n = 10$ muss man mit dem Verlust von 3 Dezimal-Stellen rechnen.

```
T10 = @(x) cos(10*acos(x));
T10a = @(x) 512*x^10-1280*x^8+1120*x^6-400*x^4+50*x^2-1;
T10b = @(x) (((((((((512*x+0)*x-1280)*x+0)*x+1120)*x+0)*x-400)*x...
             +0)*x+50)*x+0)*x-1;
eps, T10a(1-eps)-T10(1-eps), T10b(1-eps)-T10(1-eps)
```

Auch beim Clenshaw-Algorithmus hängt die Stabilität von den Zwischensummen ab, und es können ähnliche Stabilitätsprobleme beobachtet werden. Da die Koeffizienten der Čebyšev-Polynome T_n in der Monombasis für hohe Potenzen sehr groß sind ($\sim 2^{n-1}$), sind die Koeffizienten eines gegebenen Polynoms p in der Čebyšev-Basis üblicherweise sehr klein (etwa 2^{n-1} mal kleiner als in Monom-Basis). Somit sind auch die Zwischensummen im Clenshaw-Algorithmus erheblich kleiner und dieser somit viel stabiler. Wenn das Hornerschema für ein häufig auszuwertendes Polynom instabil ist, kann es sich also lohnen, zur Čebyšev-Darstellung mit Clenshaw-Algorithmus überzugehen. Für die meisten Polynome (insb. Taylorapproximationen an analytische Funktionen) reicht das Horner-schema jedoch aus. De Casteljaus Algorithmus ist i.A. gutmütig:

<u>Thm:</u> (Stabilität) Wird $p(t)$ für $t \in [0,1]$ via de Casteljaus Algorithmus mittels Maschinenoperationen ausgewertet, so erfüllt das berechnete $\hat p(t)$: $|\hat p(t) - p(t)| \le 2n B \varepsilon_m + O(\varepsilon_m^2)$ mit $B = \max_{i=0,\ldots,n} |\beta_i|$.

Bew.: $\hat{\ }$ bezeichne Maschinen-berechnete Größen

Sei t und $1-t$ exakt gegeben. Wir zeigen per vollständiger Induktion in k, dass

$$|\beta_i^{(k)}| \le B \quad , \quad |\hat\beta_i^{(k)} - \beta_i^{(k)}| \le 2k B \varepsilon_m + O(\varepsilon_m^2), \qquad i = 0, \ldots, n-k.$$

Induktionsanfang: $k=0$ ist trivial

Induktionsschritt: $\beta_i^{(k)} = \beta_i^{(k-1)}(1-t) + \beta_{i+1}^{(k-1)} t \le B(1-t) + Bt = B$

$\hat\beta_i^{(k)} = \hat\beta_i^{(k-1)} \odot (1-t) \oplus \hat\beta_{i+1}^{(k-1)} \odot t = \hat\beta_i^{(k-1)}(1-t)(1+\varepsilon_1)(1+\delta) + \hat\beta_{i+1}^{(k-1)} t (1+\varepsilon_2)(1+\delta)$ mit $|\varepsilon_1|, |\varepsilon_2|, |\delta| \le \varepsilon_m$

$= \beta_i^{(k)} + \underbrace{(\hat\beta_i^{(k-1)} - \beta_i^{(k-1)})}_{\le B(k-1)2\varepsilon_m}(1-t) + \underbrace{(\hat\beta_{i+1}^{(k-1)} - \beta_{i+1}^{(k-1)})}_{\le B(k-1)2\varepsilon_m} t + \underbrace{\hat\beta_i^{(k-1)}}_{\le B + B(k-1)2\varepsilon_m}(1-t)(\varepsilon_1+\delta) + \hat\beta_{i+1}^{(k-1)} t(\varepsilon_2+\delta) + O(\varepsilon_m^2)$

$\Rightarrow |\hat\beta_i^{(k)} - \beta_i^{(k)}| \le B(k-1)2\varepsilon_m + B\,2\varepsilon_m + O(\varepsilon_m^2) = Bk2\varepsilon_m + O(\varepsilon_m^2)$ □

heute: · rationale Funktionen
· Polynominterpolation

Rationale Funktionen

Nimmt man zu $+, -, \cdot$ noch $/$ hinzu, lassen sich rationale Funktionen auswerten.

Def. (rationale Funktion) Eine rationale Funktion vom Typ (m,n) ist ein Quotient $r = \frac{p}{q}$ zweier teilerfremder Polynome $p \in P_m$, $q \in P_n \setminus \{0\}$. (Nennergrad, Zählergrad)

Eine rationale Funktion kann auf verschiedene Arten dargestellt und somit auch ausgewertet werden, z.B. können p und q in Monombasis geschrieben und mit dem Hornerschema ausgewertet werden $\Rightarrow$ insgesamt $(m+n)$ mal $\cdot$, $(m+n)$ mal $+$ bzw. $-$, 1 mal $/$

Eine Alternative ist Folgendes:

Thm. (Kettenbruch) Eine rationale Funktion $r = \frac{p}{q}$ vom Typ (m,n) lässt sich darstellen durch

$$r = p_0 + \cfrac{1}{p_1 + \cfrac{1}{p_2 + \ddots + \cfrac{1}{p_s}}} = p_0 + \left\lfloor \frac{1}{p_1} \right\rfloor + \left\lfloor \frac{1}{p_2} \right\rfloor + \ldots + \left\lfloor \frac{1}{p_s} \right\rfloor$$

für Polynome $p_i \in P_{n_i}$ mit $\sum_{i=1}^{s} n_i = n$, $n_0 = \max(0, m-n)$.

Bem: Die Auswertung kostet s mal $/$ und $\sum_{i=0}^{s} n_i = \max(m,n)$ mal $\cdot$ und $+$ bzw. $-$.

Beweis: Setze $r_{-1} := p$, $r_0 := q$, d.h. $r = \frac{r_{-1}}{r_0}$.

Euklidischer Divisionsalgorithmus $\Rightarrow$ $r_{-1} = p_0 r_0 + r_1 \Rightarrow r = p_0 + \frac{r_1}{r_0} = p_0 + \frac{1}{r_0/r_1}$

$r_0 = p_1 r_1 + r_2 \Rightarrow r = p_0 + \frac{1}{p_1 + \frac{r_2}{r_1}}$

$\vdots$

$r_{i-1} = p_i r_i + r_{i+1} \Rightarrow r = p_0 + \left\lfloor \frac{1}{p_1} \right\rfloor + \ldots + \left\lfloor \frac{1}{p_i} \right\rfloor + \left\lfloor \frac{r_{i+1}}{r_i} \right\rfloor$

$n_i = \text{Grad}\, p_i = \text{Grad}\, r_{i-1} - \text{Grad}\, r_i > 0$ für $i = 1, 2, \ldots$

$\Rightarrow$ Grad r_i nimmt monoton ab $\Rightarrow$ Iteration endet nach s Schritten mit $r_s = 0$ und

$\sum_{i=1}^{s} n_i = (\text{Grad}\, r_0 - \text{Grad}\, r_1) + (\text{Grad}\, r_1 - \text{Grad}\, r_2) + \ldots + (\text{Grad}\, r_{s-1} - \text{Grad}\, r_s) = \text{Grad}\, r_0 - \text{Grad}\, r_s = \text{Grad}\, r_0 = n$. □

Auch andere Darstellungen sind denkbar wie z.B. $r = p_0 + \left\lfloor \frac{p_1}{1} \right\rfloor + \left\lfloor \frac{p_2}{1} \right\rfloor + \ldots + \left\lfloor \frac{p_s}{1} \right\rfloor$, daher untersuchen wir nun allgemeine Kettenbrüche.

Def. (Kettenbruch) Seien $A=(a_1,a_2,\dots)$, $B=(b_0,b_1,\dots)$ Folgen in $\mathbb{R}$, $\mathbb{C}$ oder P_m. Der von A und B erzeugte Kettenbruch der Länge n ist $k_n = b_0 + \cfrac{a_1|}{b_1} + \cfrac{a_2|}{b_2} + \dots + \cfrac{a_n|}{b_n}$ (falls kein Nenner verschwindet).

Setzen wir $k_n = \frac{P_n}{Q_n}$ (d.h. z.B. $P_0=b_0$, $Q_0=1$) so können P_n, Q_n rekursiv berechnet werden:

Alg: (Euler & Wallis)

$k_n = \frac{P_n}{Q_n}$ für $\quad P_{-1}=1,\ P_0=b_0,\ P_i = b_i P_{i-1} + a_i P_{i-2}$

$\qquad\qquad\qquad\quad Q_{-1}=0,\ Q_0=1,\ Q_i = b_i Q_{i-1} + a_i Q_{i-2}$

Bew.: Induktionsanfang: $k_0 = \frac{P_0}{Q_0} = b_0$

Induktionsschritt: $k_n = b_0 + \cfrac{a_1|}{b_1} + \dots + \cfrac{a_{n-1}|}{\tilde b_{n-1}}$ mit $\tilde b_{n-1} = b_{n-1} + \frac{a_n}{b_n}$.

Nach Induktionsvoraussetzung also $k_n = \frac{\tilde P_{n-1}}{\tilde Q_{n-1}}$ für $\tilde P_{n-1} = \tilde b_{n-1} P_{n-2} + a_{n-1} P_{n-3}$, $\tilde Q_{n-1} = \tilde b_{n-1} Q_{n-2} + a_{n-1} Q_{n-3}$

$$\Rightarrow k_n = \frac{b_{n-1}P_{n-2} + \frac{a_n}{b_n}P_{n-2} + a_{n-1}P_{n-3}}{b_{n-1}Q_{n-2} + \frac{a_n}{b_n}Q_{n-2} + a_{n-1}Q_{n-3}} = \frac{P_{n-1} + \frac{a_n}{b_n}P_{n-2}}{Q_{n-1} + \frac{a_n}{b_n}Q_{n-2}} = \frac{1/b_n}{1/b_n}\,\frac{P_n}{Q_n}$$ □

Eine weitere wichtige Darstellung erhält man durch Partialbruchzerlegung.

Thm. (Partialbruchzerlegung) Sei $r=\frac{p}{q}$ eine rationale Funktion vom Typ (m,n), $m<n$, mit

$$q = \prod_{i=1}^{s} (x-z_i)^{n_i},\quad z_i\in\mathbb{C},\ z_i\neq z_j \text{ für } i\neq j,\ n_i\in\mathbb{N},\ \sum_{i=1}^{s} n_i = n$$

(nach dem Fundamentalsatz der Algebra existiert eine solche Darstellung), dann gibt es $a_{i\lambda}\in\mathbb{C}$, $\lambda=1,\dots,n_i$, $a_{in_i}\neq 0$, sodass $r=\sum_{i=1}^{s} H_i$ für $H_i(x) = \sum_{\lambda=1}^{n_i} \frac{a_{i\lambda}}{(x-z_i)^\lambda}$.

Bew: Es ist $q=(x-z_s)^{n_s}\tilde q$ für $\tilde q\in P_{n-n_s}$, $\tilde q(z_s)\neq 0$.

- für $\alpha = \frac{p(z_s)}{\tilde q(z_s)}$ gilt $r = \frac{\alpha}{(x-z_s)^{n_s}} + \tilde r$ mit $\tilde r = \frac{p(x)-\alpha\tilde q(x)}{(x-z_s)^{n_s}\tilde q(x)} =: \frac{\tilde p(x)}{q(x)}$
- $\tilde p(z_s)=0$, d.h. aus $\frac{\tilde p(x)}{q(x)}$ kann $(x-z_s)$ gekürzt werden und man erhält $\tilde r = \frac{\hat p}{\hat q}$ mit $\operatorname{Grad}\hat p < \operatorname{Grad} p$, $\operatorname{Grad}\hat q < \operatorname{Grad} q$
- Wiederhole Prozedur für $\frac{\hat p}{\hat q}$ und iteriere bis $\hat q = 1$. □

Bem: Der Beweis ist konstruktiv und gibt somit einen Algorithmus an. Allerdings können die Pole $z_i\in\mathbb{C}$ i.A. nur approximativ gefunden werden.

Bsp: $p(x) = x-1$, $q(x) = x^3 - 9x^2 + 26x - 24 = (x-2)(x-3)(x-4)$

1) $\alpha = \frac{p(4)}{\tilde q(4)} = \frac{3}{2} \Rightarrow r = \frac{3/2}{x-4} + \underbrace{\frac{-3x/2 + 5/2}{(x-2)(x-3)}}_{\hat r = \frac{\hat p}{\hat q}}$; ersetze nun $p\leftarrow\hat p$, $q\leftarrow\hat q$, $r\leftarrow\hat r$

2) $\alpha = \frac{p(3)}{\tilde q(3)} = \frac{-2}{1} \Rightarrow r = \frac{-2}{x-3} + \frac{1/2}{x-2}$

$\Rightarrow$ gesamt: $\frac{p}{q} = \frac{3/2}{x-4} - \frac{2}{x-3} + \frac{1/2}{x-2}$

alternativ: löse $\frac{p}{q} = \frac{\alpha}{x-4} + \frac{\beta}{x-3} + \frac{\gamma}{x-2}$ nach α,β,γ

Thm: Hat $r = \frac{p}{q}$ nur einfache Pole ($=$ Nullstellen von q) z_i, so gilt die Partialbruchzerlegung

$$r = \sum_{i=1}^{n} \frac{p(z_i)}{q'(z_i)} \frac{1}{x - z_i}.$$

Bew: Wir wissen bereits $r = \sum_{i=1}^{n} \frac{\alpha_i}{x - z_i}$ für $\alpha_1, \dots, \alpha_n \in \mathbb{C}$

$$\Rightarrow \alpha_i = \lim_{x \to z_i} (x - z_i) r = \lim_{x \to z_i} \frac{p(x)}{q(x)/(x - z_i)} = \lim_{x \to z_i} \frac{p(x)}{(q(x) - q(z_i))/(x - z_i)} = \frac{p(z_i)}{q'(z_i)} \quad \square$$

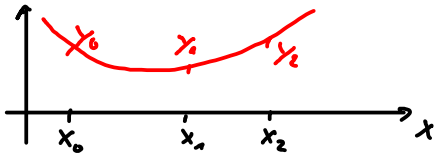
Interpolation & Approximation

Interpolation ist nützlich, um Funktionen wie sin, cos, exp, ... anhand von vorberechneten Werten an bestimmten Stellen anzunähern (früher mit Hilfe von Werttabellen; heute macht dies der Rechner intern) oder um eine unbekannte Funktion, von der nur einige Messwerte gegeben sind, zu approximieren.

Grundlagen Polynominterpolation

Def: (Polynom-Interpolationsaufgabe) Sei $\mathbb{K} = \mathbb{R}$ oder $\mathbb{K} = \mathbb{C}$, $n \in \mathbb{N}$. Gegeben Stützstellen $x_0, \dots, x_n \in \mathbb{K}$ und Stützwerte $y_0, \dots, y_n \in \mathbb{K}$ ist das Interpolationsproblem:

Finde $p \in P_n$ mit $p(x_k) = y_k$, $k = 0, \dots, n$. $(*)$



Thm: (Wohlgestelltheit) Sind die Stützstellen paarweise verschieden, besitzt $(*)$ eine eindeutige Lösung.

Bew:
- Existenz folgt aus Lagrange-Darstellung weiter unten
- Eindeutigkeit: Seien $p, q \in P_n$ Lösungen von $(*)$ $\Rightarrow r := p - q \in P_n$ mit $r(x_0) = \dots = r(x_n) = 0$

$\Rightarrow r \in P_n$ hat mehr als n Nullstellen $\Rightarrow r = 0$. $\square$

Thm: (Neville-Schema) Sei p_{ik} das Interpolationspolynom zu (x_j, y_j), $j = i, \dots, k$. Dann ist

$$p_{i,k+1}(x) = \frac{1}{x_{k+1} - x_i} \left[(x_{k+1} - x)\, p_{ik}(x) + (x - x_i)\, p_{i+1,k+1}(x) \right], \quad i = 0, \dots, n-1, \quad k = i, \dots, n-1,$$

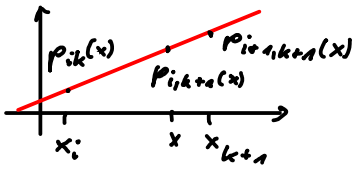
d.h. die Lösung $p_{0,n}$ von $(*)$ kann rekursiv berechnet werden.

Bew: $p_{i,k+1} \in P_{k+1-i}$ & $p_{i,k+1}(x_j) = y_j$ für $j = i, \dots, k+1$ (durch Einsetzen). $\square$

Die Berechnung nach Neville (oder Neville-Aitken) erfolgt nach dem Dreiecksschema

$$\begin{array}{lll} x_0 & y_0 = p_0 & \\ x_1 & y_1 = p_1 & p_{01} \\ x_2 & y_2 = p_2 & p_{12} \quad p_{02} \\ \vdots & \vdots & \vdots \quad \ddots \quad \ddots \\ x_n & y_n = p_n & p_{n-1,n} \quad \cdots \quad p_{0n} \end{array}$$

Bem: Man kann das gleiche Schema nutzen, um p_{0n} an einer Stelle x auszuwerten (ohne das Polynom explizit hinzuschreiben). Hierzu muss man in jedem Schritt nur eine Linear-Kombination mit Gewichten $w = \frac{x_{k+1} - x}{x_{k+1} - x_i}$ und $\tilde{w} = \frac{x - x_i}{x_{k+1} - x_i}$ bilden. Da $w + \tilde{w} = 1$, ist dies nichts anderes als eine lineare Inter-/Extrapolation:



$\Rightarrow$ das Schema funktioniert nicht nur in $\mathbb{K}$, sondern in allen Räumen, in denen „lineare Inter-/Extrapolation" definiert werden kann.

Bsp: $x = 3$,

$$\begin{array}{lll} x_0 = 1 & y_0 = 0 & p_{01}(x) = \frac{-1}{1}\,0 + \frac{2}{1}\,0 = 0 \\ x_1 = 2 & y_1 = 0 & p_{12}(x) = \frac{1}{2}\,0 + \frac{1}{2}\,5 = \frac{5}{2} \\ x_2 = 4 & y_2 = 5 & \end{array} \qquad p_{02}(x) = \frac{1}{3}\,0 + \frac{2}{3}\,\frac{5}{2} = \frac{5}{3}$$

Sei $v_0, \dots, v_n \in P_n$ eine Basis von P_n. Die Lösung $p = \sum_{i=0}^{n} a_i v_i$ von (*) kann dann berechnet werden durch Lösen des linearen Gleichungssystems

$$\underbrace{\begin{pmatrix} v_0(x_0) & \cdots & v_n(x_0) \\ \vdots & & \vdots \\ v_0(x_n) & \cdots & v_n(x_n) \end{pmatrix}}_{=:V} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix}. \quad (**)$$

Aus der Wohlgestelltheit von (*) folgt, dass (**) für alle rechten Seiten eine eindeutige Lösung hat $\Rightarrow V \in \mathbb{R}^{(n+1)\times(n+1)}$ ist regulär, unabhängig von der gewählten Basis.

Def: (Darstellungsformen)

(1) wähle $v_i(x) = l_i(x) := \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \Rightarrow V = I$

$p = \sum_{i=0}^{n} y_i l_i$ heißt Lagrange-Darstellung des Interpolationspolynoms

(2) wähle $v_i(x) = u_i(x) := \prod_{j=1}^{i} (x - x_{j-1})$

(d.h. rekursiv $v_0(x) = 1$, $v_i(x) = (x - x_{i-1}) v_{i-1}(x)$) $\Rightarrow V = \begin{pmatrix} 1 & & & \\ 1 & x_1 - x_0 & & \\ \vdots & x_2 - x_0 & \ddots & \\ 1 & x_n - x_0 & \cdots & \prod_{i=0}^{n-1}(x_n - x_i) \end{pmatrix}$

$p = \sum_{i=0}^{n} a_i u_i$ mit $\begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = V^{-1} \begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix}$ heißt Newton-Darstellung des Interpolationspolynoms

(3) wähle $v_i(x) = m_i(x) = x^i \Rightarrow V = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix}$ = Vandermonde-Matrix

$p = \sum_{i=1}^{n} a_i m_i$ mit $\begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = V^{-1} \begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix}$ heißt Monom-Darstellung des Interpolationspolynoms

heute: Interpolationsfehler

Die Lagrange-Darstellung eignet sich gut für theoretische Zwecke, allerdings ändern sich alle Basispolynome, sobald eine neue Stützstelle hinzukommt. Die Newton-Darstellung ist so gewählt, dass nur neue Basispolynome hinzukommen und gleichzeitig V eine untere Dreiecksmatrix ist — somit ändern neue Stützstellen die alten Koeffizienten nicht. Um die Koeffizienten näher zu untersuchen, führen wir Folgendes ein.

Def: (Dividierte Differenzen) Die dividierten Differenzen zu Stützstellen $x_0,\dots,x_n$ und -werten $y_0,\dots,y_n$ sind rekursiv definiert durch

$$[y_i] = y_i \quad , i = 0,\dots,n$$

$$[y_i,\dots,y_{i+j}] = \frac{[y_{i+1},\dots,y_{i+j}] - [y_i,\dots,y_{i+j-1}]}{x_{i+j} - x_i} \quad , j = 1,\dots,n \quad , i = 0,\dots,n-j$$

Sind $y_i = f(x_i)$ für ein $f: \mathbb{K} \to \mathbb{K}$, schreiben wir auch $f[x_i,\dots,x_{i+j}] := [y_i,\dots,y_{i+j}]$.

Die Berechnung geschieht im Dreiecksschema:

$$\begin{array}{llll} x_0 & y_0 = [y_0] & [y_0,y_1] & \\ x_1 & y_1 = [y_1] & [y_1,y_2] & [y_0,y_1,y_2] \\ x_2 & y_2 = [y_2] & \vdots & \\ \vdots & \vdots & [y_{n-1},y_n] & \\ x_n & y_n = [y_n] & & \end{array} \quad \cdots \quad [y_0,y_1,\dots,y_n]$$

Bei Hinzunahme neuer Stützstellen wird das Schema unten fortgesetzt.

Thm: (Eigenschaften der Newton-Darstellung) Seien a_i die Koeffizienten der Newton-Darstellung.

(1) $a_i = a_i(y_0,\dots,y_i)$ hängt nur von $y_0,\dots,y_i$ ab und ist der höchste Koeffizient des Interpolationspolynoms zu den ersten $i+1$ Stützstellen $x_0,\dots,x_i$ (sowohl in Monom- als auch Newton-Darstellung)

(2) $a_i = (w_0,\dots,w_i)\begin{pmatrix} y_0 \\ \vdots \\ y_i \end{pmatrix}$ für die Knoten-Gewichte $w_j = \prod_{\substack{k=0 \\ k\neq j}}^{i} \frac{1}{x_j - x_k}$

(3) $a_i = [y_0,\dots,y_i]$

Beweis: (1) $a_0,\dots,a_i$ lösen die ersten $i+1$ Zeilen von $V\begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix}$, also $\begin{pmatrix} 1 & & & \\ \vdots & x_1-x_0 & & \\ 1 & x_2-x_0 & \ddots & \prod_{j=0}^{i-1}(x_i-x_j) \end{pmatrix}\begin{pmatrix} a_0 \\ \vdots \\ a_i \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_i \end{pmatrix}$

$\Rightarrow$ dies sind die Gleichungen der Koeffizienten bei Interpolation mit nur $x_0,\dots,x_i$ & $y_0,\dots,y_i$.

(2) Sei p_i das Interpolationspolynom zu $x_0,\dots,x_i$ & $y_0,\dots,y_i$. Dessen höchster Monom-Koeffizient ist wegen

$$p_i = \sum_{j=0}^{i} y_j \ell_j = \sum_{j=0}^{i} y_j \prod_{i=0,i\neq j}^{i} \frac{x - x_i}{x_j - x_i} = \sum_{j=0}^{i} y_j w_j \prod_{i=0,i\neq j}^{i} (x - x_i)$$

gleich $\sum_{j=0}^{i} y_j w_j$; der höchste Monom-Koeffizient ist jedoch auch der höchste Newton-Koeffizient a_i.

(3) Seien $r, q \in P_{\ell-1}$ die Interpolationspolynome durch die Punkte (x_j,y_j), $j = i_1,\dots,i+\ell-1$ bzw. $j = i+1,\dots,i+\ell$. Der höchste Newton-Koeffizient von r bzw. q ist $a_{\ell-1}(y_{i_1},\dots,y_{i+\ell-1})$ bzw. $a_{\ell-1}(y_{i+1},\dots,y_{i+\ell})$.

Das Polynom $o(x) \overset{\text{Neville}}{=} \frac{1}{x_{i+\ell} - x_i}\left[(x - x_i)\,q(x) + (x_{i+\ell} - x)\,r(x)\right]$ interpoliert (x_j,y_j), $j = i_1,\dots,\ell$.

Der höchste Newton-Koeffizient von o ist $a_\ell(y_{i_1},\dots,y_{i+\ell})$ und ist gleichzeitig höchster Monom-Koeffizient, also $a_\ell(y_{i_1},\dots,y_{i+\ell}) = \frac{1}{x_{i+\ell} - x_i}\left[a_{\ell-1}(y_{i+1},\dots,y_{i+\ell}) - a_{\ell-1}(y_{i_1},\dots,y_{i+\ell-1})\right]$.

$\Rightarrow$ $a_\ell(y_{i_1},\dots,y_{i+\ell})$ erfüllt gleiche Rekursion wie dividierte Differenzen

$\Rightarrow$ $a_\ell(y_{i_1},\dots,y_{i+\ell}) = [y_{i_1},\dots,y_{i+\ell}]$ $\forall\, i,\ell$, insbesondere $a_i(y_0,\dots,y_i) = [y_0,\dots,y_i]$ $\square$

Bsp: $(x_0,x_1,x_2)=(-1,0,2)$, $(y_0,y_1,y_2)=(1,2,3)$

· Neville:
$$\begin{array}{cc|l} -1 & 1 & -x\cdot 1+(x+1)2 = x+2 \\ 0 & 2 & \frac{2-x}{2}\cdot 2+\frac{x}{2}\cdot 3=\frac{x}{2}+2 \\ 2 & 3 & \end{array} \qquad \frac{2-x}{3}(x+2)+\frac{x+1}{3}\left(\frac{x}{2}+2\right) = -\frac{x^2}{6}+\frac{5}{6}x+2 = p(x)$$

· Lagrange: $p(x) = 1\cdot\frac{(x-0)(x-2)}{(-1-0)(-1-2)} + 2\cdot\frac{(x+1)(x-2)}{(0+1)(0-2)} + 3\cdot\frac{(x+1)(x-0)}{(2+1)(2-0)} = -\frac{x^2}{6}+\frac{5}{6}x+2$

· Newton:
$$\begin{array}{cc|ll} -1 & 1 & \frac{2-1}{0-(-1)}=1 & \frac{\frac{1}{2}-1}{2-(-1)}=\frac{-1}{6} \\ 0 & 2 & \frac{3-2}{2-0}=\frac{1}{2} & \end{array}$$
← die sind die Höchstkoeffizienten der jeweiligen Einträge im Neville-Schema!

$\Rightarrow p(x) = 1+1(x-(-1))-\frac{1}{6}(x-(-1))(x-0) = -\frac{x^2}{6}+\frac{5}{6}x+2$

· Monom: $V=\begin{pmatrix}1&-1&1\\1&0&0\\1&2&4\end{pmatrix} \Rightarrow \begin{pmatrix}a_0\\a_1\\a_2\end{pmatrix} = V^{-1}\begin{pmatrix}y_0\\y_1\\y_2\end{pmatrix} = \begin{pmatrix}-1/6\\5/6\\2\end{pmatrix} \Rightarrow p(x) = -\frac{x^2}{6}+\frac{5}{6}x+2$

Interpolationsfehler

Wenn eine Funktion f an Stützstellen $x_0,\dots,x_n$ ausgewertet und $f(x_0),\dots,f(x_n)$ durch ein Polynom p_n interpoliert werden, wie groß ist der Fehler $p_n - f$?

Def: (B-Spline) (1) Die abgeschnittene Potenz ist definiert durch $x_+^m := \begin{cases} x^m & \text{falls } x\geq 0 \\ 0 & \text{sonst} \end{cases}$

(2) Der B-Spline $(m-1)$-ten Grades zu Stützstellen $x_0,\dots,x_m$ ist $B_{m-1}(t) := m\sum_{i=0}^{m} w_i\,(x_i-t)_+^{m-1}$, w_i: Knoten-Gewichte

Bem: · B-Spline steht für basic spline und kann auch zur Interpolation genutzt werden (siehe später)

· B_{m-1} ist $(m-2)$ mal differenzierbar

· Sind $x_0<\dots<x_n$, kann man leicht zeigen $B_{m-1}\geq 0$ & $B_{m-1}=0$ auf $\mathbb{R}\setminus[x_0,x_m]$ (siehe später)

Thm: (Peano-Darstellung dividierter Differenzen) Ist $f\in C^m([x_0,x_m])$, d.h. m mal stetig differenzierbar, gilt
$$\begin{aligned} f[x_0,\dots,x_m] &= \frac{1}{m!}\int_{x_0}^{x_m} f^{(m)}(t)\,B_{m-1}(t)\,dt \\ &= \frac{f^{(m)}(\xi)}{m!}\int_{x_0}^{x_m} B_{m-1}(t)\,dt \qquad \text{für ein } \xi\in[x_0,x_m] \\ &= f^{(m)}(\xi)/m! \end{aligned}$$

Bew: · Der Taylorsche Satz liefert $f(x) = \underbrace{\sum_{i=0}^{m-1}\frac{f^{(i)}(a)}{i!}(x-a)^i}_{=:p(x)} + \underbrace{\int_a^x \frac{(x-t)^{m-1}}{(m-1)!}f^{(m)}(t)\,dt}_{=:r(x)}$

Wir haben bereits gezeigt $f[x_0,\dots,x_m] = \sum_{i=0}^{m} w_i f(x_i)$ (w_i: Knoten-Gewichte), also $f[x_0,\dots,x_m] = \sum_{i=0}^{m} w_i p(x_i) + \sum_{i=0}^{m} w_i r(x_i)$

Da $p\in P_{m-1}$ ist $\sum_{i=0}^{m} w_i p(x_i) = p[x_0,\dots,x_m] =$ (m-ter Koeffizient von p = 0, siehe (3) früher).

$$\Rightarrow f[x_0,\dots,x_m] = \sum_{i=0}^{m} w_i r(x_i) = \sum_{i=0}^{m} w_i \int_{x_0}^{x_i}\frac{(x_i-t)^{m-1}}{(m-1)!}f^{(m)}(t)\,dt = \int_{x_0}^{x_m}\underbrace{\sum_{i=0}^{m} w_i\frac{(x_i-t)_+^{m-1}}{(m-1)!}}_{=B_{m-1}(t)/m!}f^{(m)}(t)\,dt$$

· Der erweiterte Mittelwertsatz liefert $\frac{1}{m!}\int_{x_0}^{x_m} B_{m-1}(t)\,f^{(m)}(t)\,dt = \frac{f^{(m)}(\xi)}{m!}\int_{x_0}^{x_m} B_{m-1}(t)\,dt$ für ein $\xi\in[x_0,x_m]$

· Für $q(x)=\prod_{i=0}^{m-1}(x-x_i)$ gilt $q[x_0,\dots,x_m] = \frac{1}{m!}\int_{x_0}^{x_m} B_{m-1}(t)\,q^{(m)}(t)\,dt = \int_{x_0}^{x_m} B_{m-1}(t)\,dt$.

Außerdem ist die Newton-Darstellung der Interpolation von $q(x_0),\dots,q(x_m)$ gleich

$$q(x) = \underbrace{q[x_0]}_{=0} + \underbrace{q[x_0,x_1]}_{=0}(x-x_0)+\dots+q[x_0,\dots,x_m]\underbrace{(x-x_0)\cdots(x-x_{m-1})}_{=q(x)} \Rightarrow q[x_0,\dots,x_m]=1 \qquad \square$$

Thm. (Interpolationsfehler) Für paarweise verschiedene Stützstellen $x_0, \dots, x_n \in \mathbb{R}$ sei I das kleinste Intervall mit $x_0, \dots, x_n, x \in I$ und das Knotenpolynom definiert durch $\omega_{n+1}(x) = \prod_{k=0}^{n} (x - x_k)$.

Sei p_n das Interpolationspolynom von $f(x_0), \dots, f(x_n)$. Es gilt

$$f(x) - p_n(x) = f[x_0, \dots, x_n, x]\, \omega_{n+1}(x)$$
$$= \frac{f^{(n+1)}(\xi)}{(n+1)!}\, \omega_{n+1}(x) \quad \text{für ein } \xi \in I \quad \text{falls } f \in C^{n+1}(I)$$

Bew: Sei $x_{n+1} := x$ und $p_i(t)$ das Interpolationspolynom zu $f(x_0), \dots, f(x_i)$, d.h. in Newton-Darstellung

$$p_i(t) = f[x_0] + f[x_0, x_1](t - x_0) + \dots + f[x_0, \dots, x_i](t - x_0) \cdots (t - x_{i-1})$$

Dann ist $f(x) - p_n(x) = p_{n+1}(x) - p_n(x) = f[x_0, \dots, x_n, x] \underbrace{(x - x_0) \cdots (x - x_n)}_{\omega_{n+1}(x)}$. □

Bsp: · Für $f(x) = \cos x$ ist $|f^{(i)}(x)| \le 1 \Rightarrow |f(x) - p_n(x)| \le \frac{|\omega_{n+1}(x)|}{(n+1)!}$.

Zwischen den Stützstellen (dort wo ω_{n+1} klein ist) ist dies sehr klein!

Außerhalb der Stützstellen wächst der Fehler wie die höchste Potenz von ω_{n+1}, also x^{n+1}.

· Runge fand das Beispiel $f(x) = \frac{1}{1+25x^2}$ mit $f^{(i)}$ exponentiell anwachsend (z.B. $f^{(2n)}(0) = (-25)^n (2n)!$)

⇒ hier steigt der Fehler bei äquidistanten Stützstellen mit wachsendem n an;

am Rand ergeben sich starke Oszillationen (Runge-Phänomen)!

```
% Interpolation analytischer Funktion
n = 3;
x = linspace(-1,1,n+1); % Stuetzstellen
t = linspace(-2,2,100); % fuer Graph

p = polyfit(x,cos(x),n);
plot(t,cos(t),'k','Linewidth',3,t,polyval(p,t),'b','Linewidth',3);

% Runges Beispiel
n = 21;
x = linspace(-1,1,n+1); % Stuetzstellen

p = polyfit(x,1./(1+25*x.^2),n);
plot(t,1./(1+25*t.^2),'k','Linewidth',3,t,polyval(p,t),'b','Linewidth',3);
axis([-1.5 1.5 -5 5]);
```

Nur für „nette" Funktionen kann man also erwarten, dass die Polynominterpolation konvergiert für $n \to \infty$!

heute: · Čebyšev-Interpolation
· Stabilität

Der Interpolationsfehler hat die Form $|f(x) - p_n(x)| \le \frac{|f^{(n+1)}(\xi)|}{(n+1)!} |\omega_{n+1}(x)|$, wobei ω_{n+1} nur von den Stützstellen abhängt. Wenn man die Stützstellen wählen kann, wie macht man $|\omega_{n+1}|$ am kleinsten?

Thm: (1) Sei $\|\cdot\|_\infty$ die Supremumsnorm auf $[-1,1]$. $\left\|\frac{T_{n+1}}{2^n}\right\|_\infty = \frac{1}{2^n} \le \|p\|_\infty \; \forall p \in P_{n+1}$ mit Höchstkoeffizient 1

(2) Wählt man als Stützstellen die Nullstellen von T_{n+1}, so ist $\omega_{n+1} = \frac{T_{n+1}}{2^n}$ ⇒ dies ist optimal!

Bew: (1) Sei $q = \frac{T_{n+1}}{2^n}$ und $p \in P_{n+1}$ mit Höchstkoeffizient 1 und $\|p\|_\infty < \frac{1}{2^n} = \|q\|_\infty$. $\Rightarrow r := q - p \in P_n$

Seien $z_k = \cos\left(\frac{k\pi}{n+1}\right)$, $k = 0, \dots, n+1$, die Extrema von q.

Aus $|p(z_k)| \le \|p\|_\infty < \|q\|_\infty = |q(z_k)|$ folgt $\operatorname{sign}(r(z_k)) = \operatorname{sign}(q(z_k) - p(z_k)) = \operatorname{sign}(q(z_k))$.

Da $q(z_k)$ wechselnde Vorzeichen hat wechselt auch r das Vorzeichen $(n+1)$-mal

$\Rightarrow r$ hat mindestens $n+1$ Nullstellen $\Rightarrow r = 0$.

(2) Ein Polynom vom Grad $n+1$ ist durch seine $n+1$ Nullstellen und den Höchstkoeffizienten eindeutig bestimmt. □

Bem: · Polynominterpolation auf $[-1,1]$ mit Nullstellen z_k von T_n als Stützstellen heißt Čebyšev-Interpolation

· Für Interpolation auf $[a,b]$ transformiert man einfach affin auf $[-1,1]$, d. h. man benutzt die Stützstellen at $\frac{z_k+1}{2}(b-a)$.

· $\|\omega_{n+1}\|_\infty$ wird auch für leichte Perturbationen der z_k klein. Im Grunde reicht es, die Stützstellen auf $[-1,1]$ ungefähr mit der Dichte $\left\|\left(\frac{1}{(\sqrt{1-x^2})'}\right)\right\| = \frac{1}{\sqrt{1-x^2}}$, $x \in [-1,1]$ zu verteilen (bzw. auf dem Einheitskreis ungefähr gleichmäßig Punkte zu verteilen und diese auf die x-Achse zu projizieren). Eine Interpretation (von Nick Trefethen): Äquidistante Stützstellen haben am Rand im Vergleich zur Mitte zu wenig Information (in der Mitte gibt es zu beiden Seiten Stützstellen, nicht jedoch am Rand). Die Information möglichst gleichmäßig zu verteilen ist ein ähnliches Problem wie negative Ladungen (Elektronen), die sich abstoßen, optimal auf $[-1,1]$ zu verteilen, und dies führt zur Dichte $\sqrt{1-x^2}$.

```
% omega fuer aequidistante Stuetzstellen
n = 21;
x = linspace(-1,1,n+1); % Stuetzstellen
t = linspace(-1,1,100); % fuer Graph
omega = 1;
for i = 1:length(x)
  omega = omega .* (t-x(i));
end
plot(t,abs(omega),'Linewidth',3);

% Runges Beispiel mit Cebysev Interpolation
x = cos((2*(1:n)-1)*pi/2/n); % Cebysev Stuetzstellen
p = polyfit(x,1./(1+25*x.^2),n);
plot(t,1./(1+25*t.^2),'k','Linewidth',3,t,polyval(p,t),'b','Linewidth',3);
```

Gegebenenfalls machen wir einen zusätzlichen Fehler bei der Messung/Auswertung der Stützwerte $f(x_0), \dots, f(x_n)$

Thm: Sei $y_k = f(x_k) + \varepsilon_k$, $k = 0, \dots, n$, mit $|\varepsilon_k| \le \varepsilon$. Seien $p, q \in P_n$ die Interpolationspolynome von $f(x_0), \dots, f(x_n)$ bzw. $y_0, \dots, y_n$. Es ist $|p(x) - q(x)| \le \varepsilon L_n(x)$ & $\max_{x \in [a,b]} |p(x) - q(x)| \le \varepsilon \max_{x \in [a,b]} L_n(x)$

für $L_n(x) = \sum_{k=0}^{n} |l_k(x)|$. ↖ Lagrange-Basis

Bew: $p-q$ ist das Interpolationspolynom zu $(x_0,\varepsilon_0),\dots,(x_n,\varepsilon_n)$, also

$$p(x)-q(x) = \sum_{i=0}^{n} \varepsilon_i \, l_i(x) \le \varepsilon \sum_{i=0}^{n} |l_i(x)| .$$ □

```
% Interpolation mit Stuetzwert-Fehler
x = linspace(-1,1,n+1); % Stuetzstellen
t = linspace(-1,1,100); % fuer Graph
y = cos(x) + .01 * randn(size(x));
p = polyfit(x,y,n);
plot(t,cos(t),'k','Linewidth',3,t,polyval(p,t),'b','Linewidth',3);
```

```
% Ln-Funktion
n = 10;
Ln = 0;
for i = 0:n
  y = zeros(1,n+1);
  y(i+1) = 1;
  p = polyfit(x,y,n);
  Ln = Ln + abs(polyval(p,t));
end
plot(t,Ln,'Linewidth',3);
```

Interpolationsversionen

Zusätzlich zu Funktionswerten kann man auch Ableitungen spezifizieren. Dies heißt Hermite-Interpolation

Thm: (Hermite-Interpolation) Gegeben Stützstellen $x_0,\dots,x_n$ und Stützwerte y_i^k, $i=0,\dots,n$, $k=0,\dots,n_i$,

$N := \sum_{i=0}^{n} n_i + 1$, gibt es ein eindeutiges Polynom $p \in P_{N-1}$ mit $p^{(k)}(x_i) = y_i^k \quad \forall i,k$.

Bew: Hausaufgabe

Bsp: $(x_0,x_1) = (0,1)$, $y_0^0 = 0$, $y_0^1 = 1$, $y_1^0 = 0$

Sei $p(x) = a_0 + a_1 x + a_2 x^2 \Rightarrow \begin{pmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 0 & 1 & 2x_0 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \Rightarrow a_0 = 0,\ a_1 = 1,\ a_2 = -1$

Bem: Ähnlich zur normalen Interpolation findet man den Fehler $f(x) - p_N(x) = \frac{f^{(N+1)}(\xi)}{(N+1)!} \omega_{N+1}(x)$

mit $\omega_{N+1}(x) = \prod_{i=0}^{n} (x-x_i)^{n_i}$.

Man kann Polynominter-/-extrapolation auch nutzen, um die Genauigkeit numerischer Simulationen zu erhöhen. Wollen wir z.B. eine gDgl numerisch lösen, so müssen wir einen numerischen Parameter h wählen (die Schrittweite, siehe später). Wir können die Lösung $f(h)$ nur für $h>0$ berechnen, die exakte Lösung wäre $\lim_{h\to 0} f(h)$. Angenommen, $f(h)$ hat die Taylorentwicklung $f(h) = f(0) + a_1 h + O(h^2)$, dann (unbekannt!) kann man den $O(h)$-Term wie folgt eliminieren:

$$2f\left(\tfrac{h}{2}\right) - f(h) = f(0) + O(h^2)$$

$\Rightarrow$ man erhält eine bessere Fehlerordnung. Analog kann man höhere Ordnungen eliminieren. Dieses Verfahren heißt Richardson-Extrapolation.

Alg: (Richardson-Extrapolation) f habe in 0 eine Taylorentwicklung $f(x) = f(0) + \sum_{i=1}^{n} a_i x^i + o(x^n)$

- wähle Stützstellen $x_i = z^i h$ für ein $z \in (0,1)$, $h > 0$, $i = 0, \dots, n$. (z.B. $x = h, \frac{h}{2}, \frac{h}{4}, \dots$)
- berechne Polynominterpolation $p(x)$ zu $(x_i, f(x_i))$ bzw. werte diese an Stelle $x = 0$ aus (mit Neville-Aitken)

Thm: Die Richardson-Extrapolation $p(0)$ erfüllt $f(0) - p(0) = o(h^n)$.

Bew: Die Peano-Darstellung des Fehlers ist $f(0) - p(0) = f[x_0, \dots, x_n, 0]\, \omega_{n+1}(0)$

- $\omega_{n+1}(0) = x_0 \cdots x_n = z^{n\frac{n+1}{2}} h^{n+1}$
- Wir wissen $f[x_0, \dots, x_n, x] = \dfrac{f[x_1, \dots, x_n, x] - f[x_0, \dots, x_{n-1}, x_n]}{x - x_0} = \dfrac{f^{(n)}(\xi_1)/n! - f^{(n)}(\xi_2)/n!}{-h}$ für $\xi_1, \xi_2 \in [0,h]$

$$= \frac{1}{n!\,h}\left(a_n + o(1) - a_n - o(1)\right) = \frac{o(1)}{n!\,h}$$

$$\Rightarrow f(0) - p(0) = \frac{z^{n\frac{n+1}{2}}}{n!}\, o(h^n) \qquad \square$$

Bsp: $g(z) = g(0) + a_1 z^2 + a_2 z^4 + O(z^6)$

Setze $x = z^2$,
$f(x) = g(\sqrt{x}) = g(0) + a_1 x + a_2 x^2 + O(x^3)$

h	$f(h)$	$\frac{h/2}{-h/2} f(h) - \frac{h}{-h/2} f(h/2)$	$\frac{h/4}{-3h/4}\left(-f(h) + 2f(h/2)\right) - \frac{h}{-3h/4}\left(-f(h/2) + 2f(h/4)\right)$
$h/2$	$f(h/2)$	$\frac{h/4}{-h/4} f(h/2) - \frac{h/2}{-h/4} f(h/4)$	$= \frac{1}{3} f(h) - 2 f(h/2) + \frac{8}{3} f(h/4) \approx g(0)$
$h/4$	$f(h/4)$		

Man kann auch mit rationalen Funktionen interpolieren (etwa sog. NURBS). Üblicherweise sucht man zu Stützstellen $x_0, \dots, x_n$ und -werten $y_0, \dots, y_n$ zwei Polynome $p \in P_m$, $q \in P_{n-m}$ und löst dann das Gleichungssystem $y_i\, q(x_i) = p(x_i)$, $i = 0, \dots, n$, $p(x_0) = 1$ für die $n+2$ Koeffizienten von p und q.

Schließlich kann man auch mit beliebigen Ansatzfunktionen $g_0, \dots, g_n : \mathbb{K} \to \mathbb{K}$ interpolieren. Um eine Interpolierende $g(x) = \sum_{i=0}^{n} a_i g_i(x)$ zu erhalten, müssen die $n+1$ Gleichungen $a_0 g_0(x_i) + \dots + a_n g_n(x_i) = y_i$, $i = 0, \dots, n$, nach $a_0, \dots, a_n$ gelöst werden.

Heute: DFT

Diskrete Fouriertransformation

Die diskrete Fouriertransformation ist ein wichtiges mathematisches Werkzeug. Eine ihrer Interpretationen kommt aus der Interpolation. Wir brauchen ein bisschen Vorbereitung.

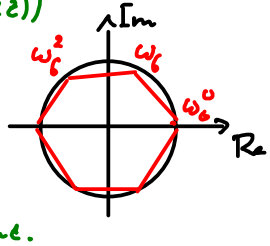
Thm: (Eulersche Identitäten) Für $z \in \mathbb{C}$ gilt

(1) $\exp(iz) = \cos z + i \sin z$

(2) $\cos(z) = \frac{1}{2}\left(\exp(iz) + \exp(-iz)\right)$

(3) $\sin(z) = \frac{1}{2i}\left(\exp(iz) - \exp(-iz)\right)$

Def: (Einheitswurzel)
- $\omega_n = \exp\left(\frac{2\pi i}{n}\right)$ heißt primitive n-te Einheitswurzel.
- $z_{j,n} = \omega_n^j$ heißen n-te Einheitswurzeln.



Bem: Die $z_{j,n}$ sind genau diejenigen Zahlen, die $z^n = 1$ lösen, daher der Name.

Herleitung A) Sei $f : \mathbb{C} \to \mathbb{C}$, wähle Stützstellen und -werte $x_j = \omega_{n+1}^j$, $y_j = f(x_j)$, $j = 0, \dots, n$.

Die Koeffizienten des Interpolationspolynoms $p_n(x) = \sum_{j=0}^{n} a_j x^j$ erfüllen

$V \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix}$ für die Vandermonde-Matrix $V_{kl} = \omega_{n+1}^{kl}$.

Herleitung B) Sei $f:\mathbb{R}\to\mathbb{C}$ periodisch mit Periode 2π, wähle periodische Ansatzfunktionen $g_k(t)=\exp(ikt)$, äquidistante Stützstellen $t_j=\frac{2\pi}{n+1}j$, $y_j=f(t_j)$, $j=0,\dots,n$.
Die Koeffizienten der Interpolation $g(t)=\sum_{j=0}^{n} a_j g_j(t)$ erfüllen auch $V\begin{pmatrix}a_0\\ \vdots\\ a_n\end{pmatrix}=\begin{pmatrix}y_0\\ \vdots\\ y_n\end{pmatrix}$.

Herleitung C) f, t_j, y_j wie in B). Der Nachteil von B) ist, dass mit diesen Ansatzfunktionen keine reelle oder imaginäre Funktion erzeugt werden kann. Aus der Analysis wissen wir jedoch, dass sich jede periodische Funktion darstellen lässt als Linearkombination $\sum_{j=-\infty}^{\infty} a_j \exp(ijt)$ $\Rightarrow$ wähle Ansatzfunktionen

$$g_k(t)=\begin{cases}\exp(ikt), & k\le \frac{n}{2}\\ \exp(i(k-n-1)t), & k>\frac{n}{2}\end{cases}$$

$\Rightarrow$ das zu lösende Gleichungssystem ist $y_j=\sum_{k=0}^{n} a_k g_k\left(\frac{2\pi j}{n+1}\right)$

wegen $\exp\left(i(k-n-1)t\frac{2\pi j}{n+1}\right)=\exp\left(ikt\frac{2\pi j}{n+1}\right)$ ist dies wieder das gleiche System.

Thm: $W=\frac{V}{\sqrt{n+1}}$ für $V=\left(\omega_{n+1}^{kl}\right)_{k,l=0,\dots,n}$ ist unitär, d.h. $W^{-1}=W^*=\overline{W}^T$

Bew: $(V^*V)_{kl}=V_{k:}^*\,V_{:l}=\overline{V}_{:k}\cdot V_{:l}=\sum_{j=0}^{n}\omega_{n+1}^{-kj}\,\omega_{n+1}^{jl}=\sum_{j=0}^{n}\omega_{n+1}^{j(l-k)}=\begin{cases}n+1, & k=l\\ \frac{(\omega_{n+1}^{l-k})^{n+1}-1}{\omega_{n+1}^{l-k}-1}=0 & \text{sonst}\end{cases}$

(geometrische Reihe)

$\Rightarrow W^*W=I$ □

Die obigen Koeffizienten erfüllen also

$$a_j=(V^{-1})_{j:}\begin{pmatrix}y_0\\ \vdots\\ y_n\end{pmatrix}=\sum_{k=0}^{n} y_k\frac{\omega_{n+1}^{-jk}}{n+1},\quad\text{und}\quad y_j=V_{j:}\begin{pmatrix}a_0\\ \vdots\\ a_n\end{pmatrix}=\sum_{k=0}^{n} a_k\,\omega_{n+1}^{jk}$$

Def: (Diskrete Fouriertransformation, DFT) Sei $y_0,\dots,y_n\in\mathbb{C}$.

(1) $(\hat{y}_0,\dots,\hat{y}_n)$ mit $\hat{y}_k=\sum_{j=0}^{n} y_j\exp\left(-\frac{2\pi ijk}{n+1}\right)$ heißt diskrete Fouriertransformation von $(y_0,\dots,y_n)$

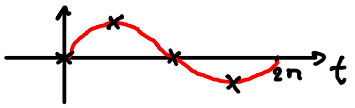
(2) $(\check{y}_0,\dots,\check{y}_n)$ mit $\check{y}_k=\frac{1}{n+1}\sum_{j=0}^{n} y_j\exp\left(\frac{2\pi ijk}{n+1}\right)$ heißt inverse diskrete Fouriertransformation von $(y_0,\dots,y_n)$

Bem: · $(\hat{y})^{\vee}=(\check{y})^{\wedge}=y$

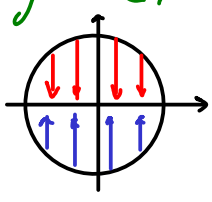
· Für obige Interpolationskoeffizienten gilt $a=\frac{\hat{y}}{n+1}$, $y=(n+1)\check{a}$.

Bsp: $n=3$, interpoliere nach Ansatz C) $t_j=\frac{2\pi j}{4}$, $j=0,\dots,3$, $y=(0\ \ 1\ \ 0\ \ -1)$

$$\left.\begin{aligned}\hat{y}_0&=y_0+y_1+y_2+y_3=0\\ \hat{y}_1&=y_0-iy_1-y_2+iy_3=-2i\\ \hat{y}_2&=y_0-y_1+y_2-y_3=0\\ \hat{y}_3&=y_0+iy_1-y_2-iy_3=2i\end{aligned}\right\}\Rightarrow g(t)=\frac{-2i}{4}\exp(it)+\frac{2i}{4}\exp(-it)=\sin t$$



Bem: Es gibt einen engen Zusammenhang zu Čebyšev-Interpolation: Fassen wir das periodische Gebiet $[0,2\pi]$ als den Einheitskreis auf und projizieren ihn auf die x-Achse, so wird



aus der Funktion $f:[0,2\pi]\to\mathbb{C}$ eine Funktion $\tilde{f}:[-1,1]\to\mathbb{C}$. Die äquidistanten Stützstellen $t_k=\frac{2\pi k}{n+1}$ auf $[0,2\pi]$ werden zu Stützstellen $x_k=\cos\frac{2\pi k}{n+1}$

auf $[-1,1]$, die nach der Dichte $\frac{1}{\sqrt{1-x^2}}$ verteilt sind – genau wie bei Čebyšev-Interpolation.

Technisches Problem hierbei: Man erhält ein $\tilde{f}$ durch runterprojizieren (rote Pfeile) und ein anderes durch hochprojizieren (blaue Pfeile) – damit beides gleich ist, muss f gerade sein, d.h. $f(t) = f(2\pi - t)$.

Technische Details zur Umgehung des Problems: Teile f in geraden und ungeraden Anteil auf

$$f_1(t) = \frac{1}{2}(f(t) + f(2\pi - t)) \quad , \quad f_2(t) = \frac{1}{2}(f(t) - f(2\pi - t))$$

und mache f_2 gerade: $f_3(t) = \sin t \, f_2(t)$

Nun projiziere f_1 & $f_3 \to \tilde{f}_1, \tilde{f}_3$ und interpoliere $\tilde{f}_1$ & $\tilde{f}_3$ durch Čebyšev-Polynome 1. & 2. Art und Stützstellen x_k (dies ist fast wie Čebyšev-Interpolation)

$$\Rightarrow \tilde{f}_1(x) \approx \sum_{k=0}^{n/2} a_k T_k(x) \quad , \quad \tilde{f}_3(x) \approx \sum_{k=1}^{n/2} b_k U_k(x)$$

$$\text{wobei } \tilde{f}_1(x_j) = \sum_{k=0}^{n/2} a_k T_k(x_j) \quad , \quad \tilde{f}_3(x_j) = \sum_{k=1}^{n/2} b_k U_k(x_j)$$

Setzen wir $x = \cos t$, ist dies äquivalent zu

$$f_1(t) \approx \sum_{k=0}^{n/2} a_k \cos(kt) \quad , \quad f_2(t) = \frac{f_3(t)}{\sin t} \approx \sum_{k=1}^{n/2} b_k \sin(kt)$$

$$\text{wobei } f_1(t_j) = \sum_{k=0}^{n/2} a_k \cos(kt_j) \quad , \quad f_2(t_j) = \sum_{k=1}^{n/2} b_k \sin(kt_j),$$

d.h. $$f(t) \approx \sum_{k=-n/2}^{n/2} c_k \exp(2\pi i k t) \quad \text{mit} \quad c_k = \begin{cases} a_0 & , k=0 \\ \frac{1}{2}(a_k - i b_k) & , k>0 \\ \frac{1}{2}(a_k + i b_k) & , k<0 \end{cases}$$

$$\text{wobei } f(t_j) = \sum_{k=-n/2}^{n/2} c_k \exp(2\pi i t_j),$$

d.h. die a_k & b_k sind die Komponenten der diskreten Fouriertransformation.

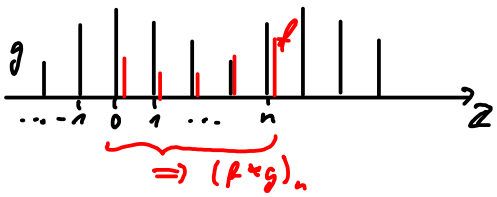
Somit können die Interpolationsfehler-Ergebnisse von der Čebyšev-Interpolation auf die trigonometrische Interpolation übertragen werden.

Eine weitere wichtige Anwendung der DFT hängt mit Faltung zusammen. Hierzu fassen wir (wie eigentlich auch schon oben) $(y_0, \dots, y_n)$ als Repräsentant einer periodischen Folge $\dots | y_0 y_1 \dots y_n | y_0 \dots$ auf.

Sei $V_n = \{ f \mid (f_k)_{k \in \mathbb{Z}} \text{ ist } (n+1)\text{-periodisch} \}$.

Def: (Hadamard- & Faltungsprodukt) Seien $f, g \in V_n$.

- <u>Hadamard- (bzw. punktweise) Produkt</u>: $f \cdot g = (f_k g_k)_{k \in \mathbb{Z}} \in V_n$
- <u>Faltungsprodukt</u>: $f * g = \left(\sum_{j=0}^{n} f_j g_{k-j} \right)_{k \in \mathbb{Z}} \in V_n$



Thm: (Faltungssatz) Seien $f, g \in V_n$.

(1) $(f * g)^\wedge = \hat{f} \cdot \hat{g}$

(2) $(f \cdot g)^\wedge = \frac{1}{n+1} \hat{f} * \hat{g}$

Beweis: Hausaufgabe

Bem: Entsprechendes gilt für $\check{}$. (Hausaufgabe)

Der Faltungssatz ist wichtig, weil Faltung oft benötigt wird, $\hat{}$ bzw. $\check{}$ aber erheblich weniger Rechenoperationen brauchen als $*$ ($\sim n \log n$ statt n^2, siehe später).

heute: · DFT-Anwendung
· FFT

Anwendungsbeispiele für DFT

Bem: · Die DFT hat (wie übrigens auch Interpolation allgemein) auch höherdimensionale Versionen. Z.B. in 2D wird y_{kl}, $k = 0, \dots, n_1$, $l = 0, \dots, n_2$ aufgefasst als (n_2+1)-periodische Folge im ersten und (n_2+1)-periodische Folge im zweiten Index. Die DFT wird in beide Richtungen separat gemacht:

$$(\hat{y})_{ab} = \left((y_{k0}, \dots, y_{kn_2})\hat{}_l \right)\hat{}_a = \left[\left(\sum_{j=0}^{n_2} y_{kj} \exp\left(\frac{2\pi i jb}{n_2+1}\right) \right)\hat{} \right]_a$$

$$= \sum_{k=0}^{n_1} \left(\sum_{j=0}^{n_2} y_{kj} \exp\left(\frac{2\pi i jb}{n_2+1}\right) \right) \exp\left(\frac{2\pi i ka}{n_1+1}\right)$$

$$= \sum_{k=0}^{n_1} \sum_{j=0}^{n_2} y_{kj} \exp\left[-2\pi i \left(\frac{ka}{n_1+1} + \frac{jb}{n_2+1}\right)\right]$$

$$(\check{y})_{ab} = \frac{1}{n_1+1}\frac{1}{n_2+1} \sum_{k=0}^{n_1} \sum_{j=0}^{n_2} y_{kj} \exp\left[2\pi i \left(\frac{ka}{n_1+1} + \frac{jb}{n_2+1}\right)\right]$$

· Der Faltungssatz gilt weiterhin mit $(f \cdot g)_{kl} = f_{kl} \cdot g_{kl}$, $(f * g)_{kl} = \sum_{j=0}^{n_1} \sum_{m=0}^{n_2} f_{jm} g_{k-j,l-m}$

1) In der Bildverarbeitung werden $n_1 \times n_2$ Pixelbilder aufgefasst als $(n_1 \times n_2)$-Matrizen u mit u_{kl} dem Grauwert des (k,l)-Pixel. Es können „features" durch Faltung gefunden werden, z.B. Bildkanten durch Faltung mit $\begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ (mit 0 erweitert auf $n_1 \times n_2$-Format)

Faltung liefert Differenzenquotienten, also Approximation an Ableitung!

```
img = double(imread('Schloss.png'));
imagesc(img); colormap(gray);

edgeDet = zeros(size(img));
edgeDet(1:3,1:3) = [0 -1 0;-1 0 1;0 1 0];
edges = real( ifft2( fft2(img) .* fft2(edgeDet) ) );
imagesc(edges);
```

2) Bilder können geglättet werden durch Faltung mit $\frac{1}{2}(1\ 2\ 1)$ oder einem Gauß-Kern

```
smoother = zeros(size(img));
n = 20;
gauss = exp(-(-n:n).^2/n^2);
smoother(1:2*n+1,1:2*n+1) = gauss'*gauss;
smoothed = real( ifft2( fft2(img) .* fft2(smoother) ) );
imagesc(smoothed);
```

3) Bilder können durch Weglassen der hohen Fourierkomponenten (die hohe, nicht relevante Oszillationen enthalten) komprimiert werden (dies ist eine Grundidee von jpeg)

```
imgHat = fft2(img);
imgHat = imgHat(round([1:end/10,9*end/10:end]),round([1:end/10,9*end/10:end]));
compressed = real( ifft2( imgHat ) );
imagesc(compressed);
numel(img) / numel(imgHat)
```

Obiges funktioniert natürlich nicht nur mit Bildern, sondern mit allen möglichen Signalen (auch Ton etc.)

4) pDgln wie z.B. die Wärmeleitung $\frac{\partial y}{\partial t} = \Delta y$, $t \in [0,\infty)$, $x \in [0,1]$ periodisch, können durch Differenzenquotienten approximiert werden: Sei $y_j^n = y(t_n, x_j)$ mit $t_n = n\tau$, $x_j = jh$, dann ist

im Prinzip auch n möglich, dies macht jedoch Probleme...

$$\frac{y_j^{n+1} - y_j^n}{\tau} \approx \frac{y_{j+1}^{n+1} - 2y_j^{n+1} + y_{j-1}^{n+1}}{h^2}$$

Die rechte Seite ist Faltung von $(y_j^{n+1})_{j\in\mathbb{Z}}$ mit $\frac{1}{h^2}(-2 \quad 1 \quad 0 \ \dots \ 0 \quad 1) =: w$

Nach DFT in j bekommen wir

$$\frac{\hat{y}_j^{n+1} - \hat{y}_j^n}{\tau} = \hat{w}_j \hat{y}_j^{n+1} \quad \Rightarrow \hat{y}_j^{n+1} = \hat{y}_j^n / (1 - \tau \hat{w}_j)$$

$\Rightarrow$ sehr einfaches Verfahren!

```
n = 200;
x = linspace(0,1,n); % alle xj
h = x(2)-x(1); tau = 1e-5;
w = zeros(1,n); w([1 2 n]) = [-2 1 1]/h^2;
factor = 1./(1-tau*fft2(w));
y = rand(1,n);
for j = 1:3000
  plot(x,y,'Linewidth',3);
  axis([0 1 0 1]);
  y = real(ifft2(fft2(y).*factor));
  pause(.1);
end
```

5) Sei $p(x) = \sum_{i=0}^{n} a_i x^i$, $q(x) = \sum_{i=0}^{n} b_i x^i \Rightarrow (pq)(x) = \sum_{i=0}^{2n} c_i x^i$ mit $c_i = \sum_{j=\max(0,i-n)}^{i} a_j b_{i-j}$.

Die naive Berechnung der c_i erfordert $1+2+3+\dots+n+n-1+\dots+2+1 = n^2$ Multiplikationen & Additionen. Fassen wir $(a_0, a_1, \dots, a_n, 0, \dots, 0)$ und $(b_0, \dots, b_n, 0, \dots, 0)$ als $(2n+2)$-periodische Folgen auf, so ist $c_i = (a * b)_i = (\hat{a} \cdot \hat{b})^\vee_i$ mit nur $n \log n$ Operationen!

Eine Anwendung ist die schnelle Multiplikation großer Zahlen, wobei $a_n a_{n-1} \dots a_0$, $b_n \dots b_0$ die Darstellung in einem B-adischen System sind (z.B. Dezimalsystem).

Schnelle Fouriertransformation (fast Fourier transform, FFT)

Die naive Implementierung der DFT $\hat{y}_k = \sum_{j=0}^{n} y_j \bar{\omega}_{n+1}^{kj}$ benötigt für jedes k n Additionen und Multiplikationen, also gesamt $2n^2$ Operationen. Die FFT hingegen braucht nur $\sim n \log_2$ Operationen

$\Rightarrow$ für $n \sim 1000$ ist dies bereits ~ 200 mal schneller, für $n \sim 10^6$ ca. 10^5 mal schneller

$\Rightarrow$ statt 3h dauert eine Rechnung nun 0,1s!

Für die Grundidee betrachten wir das Beispiel $n = 7$:

$$\begin{aligned}\hat{y}_k &= y_0 \bar{\omega}_8^0 + y_1 \bar{\omega}_8^k + y_2 \bar{\omega}_8^{2k} + y_3 \bar{\omega}_8^{3k} + \dots + y_7 \bar{\omega}_8^{7k} \\ &= \left(y_0 + y_2 \bar{\omega}_8^{2k} + y_4 \bar{\omega}_8^{4k} + y_6 \bar{\omega}_8^{6k}\right) + \bar{\omega}_8^k \left(y_1 + y_3 \bar{\omega}_8^{2k} + y_5 \bar{\omega}_8^{4k} + y_7 \bar{\omega}_8^{6k}\right) \\ &= \left[\left(y_0 + y_4 \bar{\omega}_4^{2k}\right) + \bar{\omega}_4^k \left(y_2 + y_6 \bar{\omega}_4^{2k}\right)\right] + \bar{\omega}_8^k \left[\left(y_1 + y_5 \bar{\omega}_4^{2k}\right) + \bar{\omega}_4^k \left(y_3 + y_7 \bar{\omega}_4^{2k}\right)\right]\end{aligned}$$

($\bar{\omega}_8^{2k} = \bar{\omega}_4^k$, $\bar{\omega}_8^{4k} = \bar{\omega}_4^{2k}$, $\bar{\omega}_4^{2k} = (-1)^k$)

Offenbar werden für verschiedene k immer wieder die gleichen Terme gebraucht!

Alg. (FFT nach Cooley & Tucker)

Sei $n+1 = (p+1)(q+1)$.

(1) Setze $z_{rs} = y_{(p+1)s+r}$ für $r = 0, \dots, p$, $s = 0, \dots, q$

(2) Für $r = 0, \dots, p$ berechne die DFT $(z_r)\hat{}$ von $(z_{rs})_{s=0,\dots,q}$

(3) Für $r = 0, \dots, p$ und $t = 0, \dots, q$ berechne $u_{rt} = \bar{\omega}_{n+1}^{rt} (z_r)\hat{}_t$

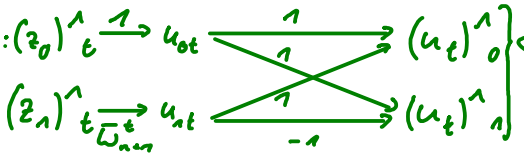
(4) Für $t = 0, \dots, q$ berechne die DFT $(u_t)\hat{}$ von $(u_{rt})_{r=0,\dots,p}$

(5) Setze $\hat{y}_{\ell(q+1)+t} = (u_t)\hat{}_\ell$ für $\ell = 0, \dots, p$, $t = 0, \dots, q$.

Bew:
$$\begin{aligned}\hat{y}_{\ell(q+1)+t} &= \sum_{j=0}^{n} \bar{\omega}_{n+1}^{j[\ell(q+1)+t]} y_j = \sum_{r=0}^{p} \sum_{s=0}^{q} \bar{\omega}_{n+1}^{[s(p+1)+r][\ell(q+1)+t]} z_{rs} \\ &= \sum_{r=0}^{p} \bar{\omega}_{n+1}^{r\ell(q+1)} \bar{\omega}_{n+1}^{rt} \sum_{s=0}^{q} \bar{\omega}_{n+1}^{s(p+1)[\ell(q+1)+t]} z_{rs} = \sum_{r=0}^{p} \bar{\omega}_{p+1}^{r\ell} \underbrace{\bar{\omega}_{n+1}^{rt} \underbrace{\sum_{s=0}^{q} \bar{\omega}_{q+1}^{st} z_{rs}}_{(z_r)\hat{}_t}}_{u_{rt}} = (u_t)\hat{}_\ell \quad \square\end{aligned}$$

($\bar{\omega}_{ab}^{a} = \bar{\omega}_b$)

Bem:
- Die DFT der Länge $n+1$ kann also durchgeführt werden als $(p+1)$ DFT der Länge $q+1$ (2), $(n+1)$ Multiplikationen (3), $(q+1)$ DFT der Länge $p+1$ (4)
- Für $p = 1$ sind Schritt (3) & (4): $(z_0)\hat{}_t \xrightarrow{1} u_{0t}$, $(z_1)\hat{}_t \xrightarrow{\bar{\omega}_{n+1}^t} u_{1t}$, $u_{0t} \pm u_{1t} \to (u_t)\hat{}_0, (u_t)\hat{}_1$ $\Leftrightarrow$ $(z_0)\hat{}_t, (z_1)\hat{}_t \to (u_t)\hat{}_0, (u_t)\hat{}_1$ mit Faktoren $1, 1, 1, -1$ bzw. $1, 1, \bar{\omega}_{n+1}^t, -\bar{\omega}_{n+1}^t$



Aufgrund des $\boxtimes$-Schemas heißen solche Algorithmen auch „Schmetterlings"-Schemata

Thm: (Aufwand FFT) Sei R_k die Anzahl an Rechenoperationen der FFT der Länge $n+1 = 2^k$. Es gilt

$$R_k = 2R_{k-1} + 2^k = k2^k$$

Bew: Hausaufgabe

Splineinterpolation

Die Computerhardware kann auch Verzweigungen („if") auswerten, daher ist auch eine Interpolation/Approximation mit stückweise definierten Polynomen/rationalen Funktionen effizient möglich. Hierdurch kann das Runge-Phänomen vermieden werden (siehe später). Außerdem haben bestimmte stückweise Polynome sehr schöne mathematische Eigenschaften (siehe später Minimalkrümmung).

Def: (Spline) Gegeben Stützstellen $x_0 < \dots < x_n \in \mathbb{R}$ ist ein **polynomialer Spline vom Grad d** eine Funktion $S: \mathbb{R} \to \mathbb{R}$ mit (1) $S|_{(x_i, x_{i+1})} \in \mathcal{P}_d$, $i = -1, \dots, n$, (2) $S \in C^{d-1}(\mathbb{R})$.

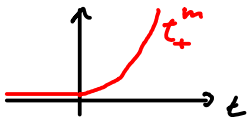
mit $x_{-1} = -\infty$, $x_{n+1} = \infty$

Die Restriktion $S|_{[x_0, x_n]}$ heißt Spline auf $[x_0, x_n]$.

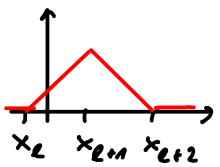
Wichtige Fälle sind $d = 1, 2, 3$ (linearer, quadratischer, kubischer Spline).

Bsp.:

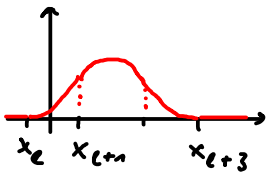
- Die abgeschnittene Potenz $t \mapsto t_+^d$ ist ein Spline vom Grad d (mit Stützstelle $x_0 = 0$)



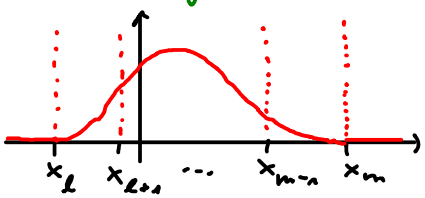
- Der B-Spline $B_{\ell,m}(t) = (m-\ell) \sum_{i=\ell}^{m} w_i (x_i - t)_+^{m-\ell-1}$ mit $w_j = \prod_{\substack{k=\ell \\ k \neq j}}^{m} \frac{1}{x_j - x_k}$ ist ein Spline vom Grad $m-\ell-1$



$m = \ell + 2$, linearer B-Spline



$m = \ell + 3$, quadratischer B-Spline



Für Interpolation mit Splines benötigen wir eine Basis des Vektorraums der Splines. B-Splines eignen sich hier besonders aufgrund ihrer schönen Eigenschaften.

Thm: (Satz von Rolle rückwärts) Ist $f \in C^k([a,b])$ und hat $f^{(k)}$ auf $[a,b]$ Nullstellen mit Gesamtvielfachheit N, so hat f in $[a,b]$ Nullstellen mit Gesamtvielfachheit höchstens $N+k$. (Intervalle, auf denen f verschwindet, werden als k-fache Nullstelle gezählt.)

Bew: Hausaufgabe

Thm:

(1) $B_{\ell,m}(t) = 0 \quad \forall\, t \in \mathbb{R} \setminus (x_\ell, x_m)$

(2) $B_{\ell,m}(t) > 0 \quad \forall\, t \in (x_\ell, x_m)$

(3) $\int_{\mathbb{R}} B_{\ell,m}(t)\, dt = 1$

Bew: (1) OBdA Sei $\ell = 0$. Wegen $t_+^k = t^k + (-1)^{k+1}(-t)_+^k$ folgt für $t \le x_0$

$$B_{0,m}(t) = m \sum_{k=0}^{m} w_k (x_k - t)^{m-1} + m(-1)^m \sum_{k=0}^{m} w_k (t - x_k)_+^{m-1} = m \sum_{k=0}^{m} w_k (x_k - t)^{m-1}$$

Dies ist die m-te dividierte Differenz der Funktion $f(x) = m(x-t)^{m-1}$

(siehe „Eigenschaften der Newtondarstellung"), also $B_{0,m}(t) = f[x_0, \dots, x_m] = \frac{f^{(m)}(\xi)}{m!} = 0$ $\xi \in [x_0, x_m]$

(2) $B_{0,m}(t)$ hat in x_0 und x_m je eine $(m-1)$-fache Nullstelle.

Es kann keine weiteren Nullstellen haben, da $B_{0,m}^{(m-2)}(t)$ ein Polygonzug mit m Segmenten (auf $[x_0, x_1], [x_1, x_2], \dots, [x_{m-1}, x_m]$) und somit $\le m$ Nullstellen ist

$\Rightarrow$ nach dem Satz von Rolle rückwärts hat $B_{0,m}$ $\le 2m-2$ Nullstellen.

(3) bereits früher gezeigt □

Thm: (Spline-Basis) Gegeben Stützstellen $x_0 < \dots < x_n$, wähle beliebige $x_{-d} < \dots < x_{-1} < x_0$, $x_n < x_{n+1} < \dots < x_{n+d}$.

Die B-Splines $B_{-d,1}, B_{-d+1,2}, \dots, B_{n-1,n+d}$ bilden eine Basis des Vektorraums aller Splines vom Grad d auf $[x_0, x_n]$.

Bew: · Die B-Splines sind $n+d$ linear unabhängige Splines vom Grad d, denn sei

$$0 = \sum_{i=1}^{n+d} a_i B_{i-d-1,i}(t) =: G(t) \text{ auf } [x_0, x_n].$$

Alle Terme für $i > d$ und ihre $d-1$ Ableitungen verschwinden an der Stelle $t = x_0$

$\Rightarrow F(t) := \sum_{i=1}^{d} a_i B_{i-d-1,i}(t)$ und $F', \dots, F^{(d-1)}$ verschwinden ebenfalls in $t = x_0$

$\Rightarrow$ F hat in x_{-d} und x_0 je eine d-fache Nullstelle, gleichzeitig folgt wie oben mit dem Satz von Rolle rückwärts, dass $F \le 2d-1$ Nullstellen auf $[x_{-d}, x_0]$ hat

$\Rightarrow F = 0$

$\Rightarrow G(t) = a_{d+1} B_{0,d+1}(t)$ auf $[x_0, x_1]$ $\Rightarrow a_{d+1} = 0$

$\Rightarrow G(t) = a_{d+2} B_{1,d+2}(t)$ auf $[x_1, x_2]$ $\Rightarrow a_{d+2} = 0$

$\vdots$

$\Rightarrow a_{d+1} = \dots = a_{d+n} = 0$

$\Rightarrow G(t) = a_d B_{-1,d}(t)$ auf $[x_{d-1}, x_d]$ $\Rightarrow a_d = 0$

$\Rightarrow G(t) = a_{d-1} B_{-2,d-1}(t)$ auf $[x_{d-2}, x_{d-1}]$ $\Rightarrow a_{d-1} = 0$

$\vdots$

$\Rightarrow a_d = a_{d-1} = \dots = a_1 = 0$

· Der Raum aller Splines vom Grad m auf $[x_0, x_n]$ hat Dimension $d+n$:

Es gibt n Intervalle mit je einem Polynom vom Grad d $\Rightarrow n(d+1)$ Freiheitsgrade

Es gibt $n-1$ Intervallgrenzen mit je d linearen Bedingungen an die Ableitungen

$\Rightarrow$ Dimension $= n(d+1) - (n-1)d = n+d$ □

Thm: (Splineinterpolation) Gegeben Stützstellen $x_0, \dots, x_n$ und Stützwerte $y_0, \dots, y_n$, $y_0^{(1)}, \dots, y_0^{(d-1)}$ existiert ein eindeutiger Spline S von Grad d mit $S(x_i) = y_i$, $i = 0, \dots, n$, $S^{(j)}(x_0) = y_0^{(j)}$, $j = 1, \dots, d-1$.

Beweis: Per definitionem ist $p_i := S|_{[x_{i-1}, x_i]} \in P_d$. Wir zeigen per Induktion nach i, dass p_i existiert und eindeutig ist.

Induktionsanfang: p_1 erfüllt $p_1(x_0) = y_0$, $p_1(x_1) = y_1$, $p_1^{(j)}(x_0) = y_0^{(j)}$, $j = 1, \dots, d-1$

$\Rightarrow$ es existiert eine eindeutige Lösung dieser Hermite-Interpolation

Induktionsschritt: p_{i+1} erfüllt $p_{i+1}(x_i) = y_i$, $p_{i+1}(x_{i+1}) = y_{i+1}$, $p_{i+1}^{(j)}(x_i) = p_i^{(j)}(x_i)$, $j = 1, \dots, d-1$

$\Rightarrow$ es existiert eine eindeutige Lösung dieser Hermite-Interpolation $\square$

Ähnlich kann man auch Ableitungen an beiden Enden oder periodische Randbedingungen vorgeben. Der Spline $S(t) = \sum_{i=1}^{d+n} a_i B_{i-d-1,i}(t)$ kann gefunden werden durch Lösen des $d+n$ linearen Gleichungen $S(x_i) = y_i$, $S^{(j)}(x_0) = y_0^{(j)}$ nach den $a_1, \dots, a_{d+n}$: $V \begin{pmatrix} a_1 \\ \vdots \\ a_{d+n} \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_n \\ y_0^{(1)} \\ \vdots \\ y_0^{(d)} \end{pmatrix}$

Heute: · Minimalkrümmungseigenschaft
· Bézierdarstellung

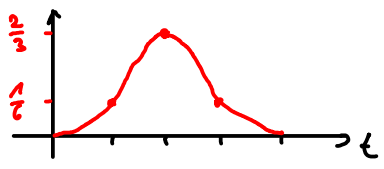
Bem: Die Nutzung der B-Splines als Basis hat Vorteile:

- **Lokalität:** Die Änderung eines Koeffizienten a_i in $\sum_{i=1}^{d+n} a_i B_{i-d-1,i}(t)$ hat nur Auswirkungen für $t \in [x_{i-d-1}, x_i]$.
- **Dünnbesetztheit:** Die Systemmatrix V hat in jeder Zeile nur d Einträge.

Bsp: · Für $x_k = k$ ist der kubische B-Spline

$$B_{0,4}(t) = 4\Big[\underbrace{\omega_0}_{\frac{1}{24}}(-t)_+^3 + \underbrace{\omega_1}_{-\frac{1}{6}}(1-t)_+^3 + \underbrace{\omega_2}_{\frac{1}{4}}(2-t)_+^3 + \underbrace{\omega_3}_{-\frac{1}{6}}(3-t)_+^3 + \underbrace{\omega_4}_{\frac{1}{24}}(4-t)_+^3\Big]$$

$$= \begin{cases} t^3/6 & t \in [0,1] \\ -t^3/2 + 2t^2 - 2t + 2/3 & t \in [1,2] \\ t^3/2 - 4t^2 + 10t - 22/3 & t \in [2,3] \\ (4-t)^3/6 & t \in [3,4] \\ 0 & \text{sonst} \end{cases}$$



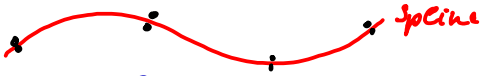
$$B_{i,i+4}(t) = B_{0,4}(t-i)$$

· Interpoliere mit einem kubischen Spline S die Stützstellen $(0, 1, 2, 3)$ mit $-$werten $(0, 1, 1, 0)$ und Ableitungen $S'(0) = 0$, $S'(3) = 0$

$$\begin{pmatrix} S'(0) \\ S(0) \\ S(1) \\ S(2) \\ S(3) \\ S'(3) \end{pmatrix} = \begin{bmatrix} B'_{-3,1}(0) & B'_{-2,2}(0) & \cdots & B'_{2,6}(0) \\ B_{-3,1}(0) & B_{-2,2}(0) & \cdots & B_{2,6}(0) \\ \vdots & & & \\ B'_{-3,1}(3) & B'_{-2,2}(3) & \cdots & B'_{2,6}(3) \end{bmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_6 \end{pmatrix} = \frac{1}{6} \begin{bmatrix} -3 & 0 & 3 & & & \\ 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & 1 & 4 & 1 & \\ & & & 1 & 4 & 1 \\ & & & -3 & 0 & 3 \end{bmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \Rightarrow \begin{pmatrix} a_1 \\ \vdots \\ a_6 \end{pmatrix} = \frac{2}{3} \begin{pmatrix} 2 \\ -1 \\ 2 \\ 2 \\ -1 \\ 2 \end{pmatrix}$$

```
% abgeschnittene Potenz
p = @(t,n) max(0,t).^n;
% kubischer B-spline
B = @(t) p(-t,3)/6-2/3*p(1-t,3)+p(2-t,3)-2/3*p(3-t,3)+p(4-t,3)/6;

t = linspace(-3,6,100);
for j = -3:2
  plot(t,B(t-j),'Linewidth',3); hold on;
end;
plot(t,4/3*B(t+3)-2/3*B(t+2)+4/3*B(t+1)+4/3*B(t)-2/3*B(t-1)+4/3*B(t-2),...
     'r','Linewidth',3);
```

Der Name Spline stammt aus dem Bootsbau: Um gekrümmte Linien zu zeichnen, wurden dünne Holzlatten (Splines) an bestimmten Punkten eingespannt $\Rightarrow$ durch Minimieren der elastischen Biegeenergie ergibt sich eine glatte Kurve.



Dieses Verhalten wird von kubischen Splines imitiert. Für $f \in C^2([a,b])$ ist der Krümmungsradius in x gegeben durch $\rho(x) = \frac{[1+(f'(x))^2]^{3/2}}{|f''(x)|}$.

Ist $f'(x)$ klein, kann die Krümmung in x durch $\kappa(x) = \frac{1}{\rho(x)} \approx |f''(x)|$ angenähert werden.

Die Biegeenergie einer dünnen Latte ist $\int_a^b \kappa(x)^2 dx \approx \int_a^b |f''(x)|^2 dx =: J_{a,b}[f]$

Thm: (Minimalkrümmungseigenschaft) Sei S der kubische Interpolationsspline mit $S(x_i) = y_i$, $i = 0,\dots,n$, $S'(x_0) = m_0$, $S'(x_n) = m_n$. S minimiert die Biegeenergie J_{x_0,x_n} unter allen Interpolationskurven.

Bew: Sei f eine andere Interpolation. Wegen $f = S + [f-S]$ gilt

$$J_{x_0,x_n}[f] = \int_{x_0}^{x_n} S''(x)^2 dx + 2\underbrace{\int_{x_0}^{x_n} S''(x)(f''(x)-S''(x))dx}_{=:A} + \int_{x_0}^{x_n} (f''(x)-S''(x))^2 dx .$$

$$A = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} S''(f''-S'')dx = \sum_{i=1}^{n} \Big[S''(x)(f'(x)-S'(x))\Big]_{x_{i-1}}^{x_i} - \int_{x_{i-1}}^{x_i} S'''(x)(f'(x)-S'(x))dx$$

$$= -S''(x_0)\underbrace{[f'(x_0)-S'(x_0)]}_{=0} + S''(x_n)\underbrace{[f'(x_n)-S'(x_n)]}_{=0} - \sum_{i=1}^{n} S'''\Big(\frac{x_i+x_{i-1}}{2}\Big)\int_{x_{i-1}}^{x_i}(f'(x)-S'(x))dx$$

$$= -\sum_{i=1}^{n} S'''\Big(\frac{x_i+x_{i-1}}{2}\Big)\Big[f(x)-S(x)\Big]_{x_{i-1}}^{x_i} = 0$$

$$\Rightarrow J_{x_0,x_n}[f] = \int_{x_0}^{x_n} S''(x)^2 dx + \int_{x_0}^{x_n} (f''(x)-S''(x))^2 dx \geq J_{x_0,x_n}[S] \qquad \square$$

Ein weiterer Vorteil von Spline-Interpolation: Das Runge-Phänomen bleibt aus! Der Einfachheit halber betrachten wir hier nur äquidistante Stützstellen und lineare Splines:

Thm: (Konvergenz) Sei $f \in C([a,b])$ und S_n der lineare Interpolationsspline durch $(x_i, f(x_i))$, $i = 0,\dots,n$, $x_i = a + \frac{b-a}{n} i$. S_n konvergiert gleichmäßig gegen f für $n \to \infty$.

Bew: Hausaufgabe

Bézier-Darstellung von Polynomen & Splines: Graphikmethoden

heute: Béziersplines

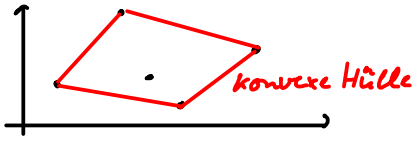
Splines werden zum Zeichnen und für CAD benutzt (z.B. Kurven in Microsoft Office oder OpenOffice sind kubische Splines; auch Schriftarten werden mit Splines definiert). Dabei wird typischerweise die Bézier-Darstellung gewählt, da sie für praktische Anwendungen viele schöne Eigenschaften hat.

Def: (Konvexität) $U \subset \mathbb{R}^n$ heißt **konvex**, wenn für alle $x, y \in U$ gilt $(1-t)x + ty \in U \quad \forall t \in [0,1]$.

Die **konvexe Hülle** von $x_0, \dots, x_m \in \mathbb{R}^n$ ist $\operatorname{conv}\{x_0, \dots, x_m\} = \left\{ x \in \mathbb{R}^n \mid x = \sum_{i=0}^m \lambda_i x_i, 0 \le \lambda_i \le 1, \sum_{i=0}^m \lambda_i = 1 \right\}$

Thm: (Konvexe Hülle) $\operatorname{conv}\{x_0, \dots, x_m\}$ ist die kleinste konvexe Menge, die $x_0, \dots, x_m$ enthält.

Bsp:



Thm: (Bézier-Kurve) (1) $t = \sum_{i=0}^n \frac{i}{n} b_{in}(t)$

(2) Für $t \in [0,1]$ liegt der Graph eines Polynoms $p(t) = \sum_{i=0}^n \beta_i b_{in}(t)$ in Bézierdarstellung in der konvexen Hülle der Bézier-Punkte $\begin{pmatrix} 0/n \\ \beta_0 \end{pmatrix}, \dots, \begin{pmatrix} n/n \\ \beta_n \end{pmatrix}$

Beweis: (1) Es ist $\frac{i}{n}\binom{n}{i} = \frac{i}{n} \frac{n!}{i!(n-i)!} = \frac{(n-1)!}{(i-1)!(n-i)!} = \binom{n-1}{i-1}$

$$\Rightarrow \sum_{i=0}^n \frac{i}{n} b_{in}(t) = \sum_{i=1}^n \frac{i}{n}\binom{n}{i} t^i (1-t)^{n-i} = t \sum_{i=1}^n \binom{n-1}{i-1} t^{i-1}(1-t)^{n-1-(i-1)}$$

$$= t \sum_{i=0}^{n-1} \binom{n-1}{i} t^i (1-t)^{n-1-i} = t\,(t+(1-t))^{n-1} = t$$

(2) $\begin{pmatrix} t \\ p(t) \end{pmatrix} = \sum_{i=0}^n b_{in}(t) \begin{pmatrix} i/n \\ \beta_i \end{pmatrix}$ mit $\sum_{i=0}^n b_{in}(t) = 1$ und $b_{in}(t) \in [0,1] \ \forall i$ □

Die Bézierpunkte geben somit den Kurvenverlauf bereits grob vor, was gut für interaktive graphische Anwendungen ist. Um mehrere Bézier-Kurven zu einem Spline zu verbinden, müssen an den Schnittstellen die Ableitungen übereinstimmen.

Thm: (Ableitungsformel) Für $p(t) = \sum_{i=0}^n \beta_i b_{in}(t)$ gilt $p^{(k)}(t) = n(n-1)\cdots(n-k+1) \sum_{i=0}^{n-k} \Delta^k \beta_i \, b_{i,n-k}(t)$

mit $\Delta^k \beta_i = \Delta^{k-1}\beta_{i+1} - \Delta^{k-1}\beta_i$, $\Delta^0 \beta_i = \beta_i$.

Insbesondere ist $p^{(k)}(0) = \frac{n!}{(n-k)!}\Delta^k \beta_0$, $p^{(k)}(1) = \frac{n!}{(n-k)!}\Delta^k \beta_{n-k}$.

Bem: Die rekursiv definierten Differenzen können wieder mit einem Dreiecksschema berechnet werden:

$$\begin{array}{llll} \beta_0 & \Delta\beta_0 & \Delta^2\beta_0 & \cdots \\ \beta_1 & \Delta\beta_1 & \vdots & \ddots \\ \beta_2 & \vdots & \Delta^2\beta_{n-2} & \\ \vdots & \Delta\beta_{n-1} & & \\ \beta_n & & & \end{array} \qquad \Delta^n \beta_0$$

Beweis: Dies folgt mit vollständiger Induktion aus

$$\left[\sum_{i=0}^m \alpha_i b_{im}(t)\right]' = m \sum_{i=0}^m \alpha_i \left(b_{i-1,m-1}(t) - b_{i,m-1}(t)\right) = m \sum_{i=0}^{m-1} (\alpha_{i+1} - \alpha_i) b_{i,m-1}(t) = m \sum_{i=0}^{m-1} \Delta\alpha_i \, b_{i,m-1}(t)$$ □

Kleben wir zwei Polynome $p(t) = \sum_{i=0}^n \beta_i b_{in}(t)$ und $q(t) = \sum_{i=0}^n \alpha_i b_{in}(t)$ zusammen zum Stückweisen

Polynom $S(t) = \begin{cases} p(t) & , t < 1 \\ q(t-1) & , t \geq 1 \end{cases}$, so ist S k mal differenzierbar, wenn

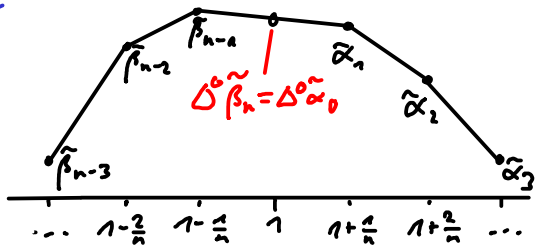
$\Delta^i \beta_{n-i} = \Delta^i \alpha_0$ für $i = 0, \dots, k$. Gilt dies für $k = n-1$, ist S ein Spline.

Für eine graphische Interpretation führen wir $\gamma_i = \frac{i}{n}$, $\tilde{\gamma}_i = 1 + \frac{i}{n}$ ein. Wegen $\Delta^i \gamma_{n-i} = \Delta^i \tilde{\gamma}_0 \ \forall i$ ist k-malige Differenzierbarkeit äquivalent zu

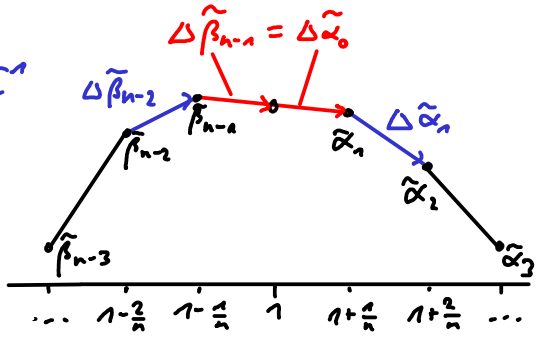
$$\Delta^i \tilde{\beta}_{n-i} = \Delta^i \tilde{\alpha}_0 \quad \text{für } i = 0, \dots, k \qquad \text{mit } \tilde{\beta}_i = \begin{pmatrix} \gamma_i \\ \beta_i \end{pmatrix}, \ \tilde{\alpha}_i = \begin{pmatrix} \tilde{\gamma}_i \\ \alpha_i \end{pmatrix}.$$

Graphische Interpretation der Differenzierbarkeit:

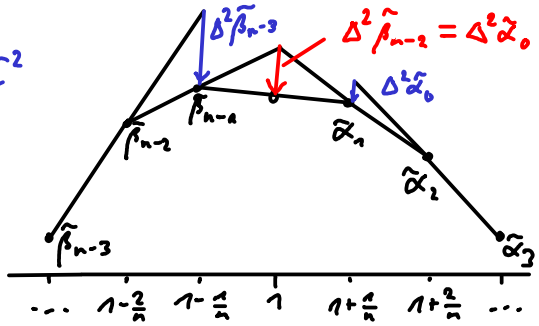
C^0



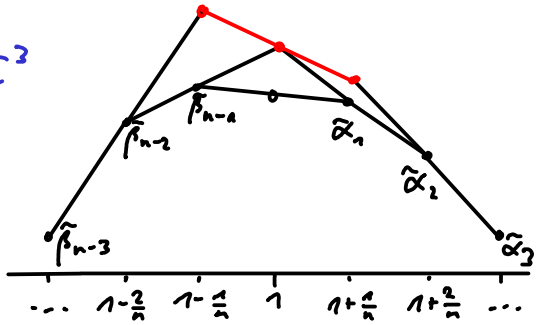
C^1



C^2



C^3

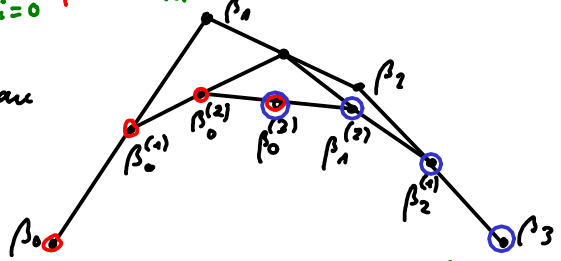


Entsprechend höhere Differenzierbarkeit erhält man, wenn sich das Gebäck weiter fortsetzen läßt.

Bem: Hieraus kann man folgern, dass ein Bézier-Polynom $p(t) = \sum_{i=0}^{n} \beta_i b_{in}(t)$ im Intervall $[0, \frac{1}{2}]$ und $[\frac{1}{2}, 1]$ in Bezier-Form geschrieben werden kann als

$$p(t) = \sum_{i=0}^{n} \beta_0^{(i)} b_{in}(2t) \ , \ t \in [0, \tfrac{1}{2}] \qquad p(t) = \sum_{i=0}^{n} \beta_i^{(n-i)} b_{in}(2t-1) \ , \ t \in [\tfrac{1}{2}, 1]$$

de Casteljau



Durch erneutes Anwenden von de Casteljau links und rechts bekommt man noch feinere Punkte, die schließlich gegen den Graphen von $p(t)$ konvergieren. Solche Verfeinerungsalgorithmen nennt man Subdivision Schemes.

Bem:
- Raumkurven $t \mapsto \vec{p}(t) \in \mathbb{R}^d$ für $d = 2$ oder $d = 3$ erzeugt man typischerweise, indem man für jede Raumkoordinate einen Spline wählt.
- Spline-Flächen erhält man durch Tensorierung, d.h. für Stützstellen $x_{ij} = (i, j) \in \mathbb{Z}^2$ wählt man die bivariaten Funktionen $B_{ij}(s,t) = B_i(s) B_j(t)$, wobei die B_i eine Basis der 1D-Splines bilden, und stellt Flächen dar als Linearkombinationen $\sum \alpha_{ij} B_{ij}(s,t)$.

heute: · Newton-Cotes Quadratur
· Gauß-Quadratur

Numerische Integration / Differentiation

Wie kann man näherungsweise das Volumen von Körpern oder die Geschwindigkeit von Objekten anhand weniger Messwerte bestimmen? Allgemeiner, wie kann man Integral & Ableitung einer Funktion anhand weniger Funktionswerte annähern? Dies bildet u.a. die Grundlage der numerischen Lösung von Differentialgleichungen.

Quadraturen (= Integralapproximation) mittels Polynominterpolation

Def: (Quadratur) Sei $f:[a,b]\to\mathbb{R}$, $a\le x_0<\dots<x_n\le b$ Stützstellen. Eine Approximation

$$I_n(f)=\sum_{i=0}^{n} a_i f(x_i)$$

an $I(f)=\int_a^b f(x)\,dx$ heißt **Quadraturformel** mit **Gewichten** $a_0,\dots,a_n$. $|I_n(f)-I(f)|$ heißt **Quadraturfehler**. I_n heißt **exakt von Ordnung/Grad m**, falls $I_n(p)=I(p)\ \forall p\in P_m$.

Quadraturformeln erhält man z.B. mittels Polynominterpolation:

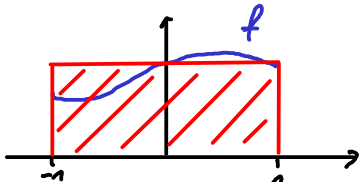
Def: (Newton-Cotes-Formel) Sei $p_n^f(x)=\sum_{i=0}^{n} f(x_i)\ell_i(x)$ das Interpolationspolynom zu f in Lagrange-Darstellung. $I_n(f):=\int_a^b p_n^f(x)\,dx=\sum_{i=0}^{n} a_i f(x_i)$ mit $a_i=\int_a^b \ell_i(x)\,dx$ heißt **Newton-Cotes-Formel**. Ist $x_0=a$, $x_n=b$ heißt sie **geschlossen**, im Fall $x_0>a$, $x_n<b$ **offen**.

Bem:
- Typischerweise verwendet man in der Newton-Cotes-Formel äquidistante Stützstellen.
- Offene Formeln eignen sich, wenn f einen Pol in a oder b hat.
- Die Newton-Cotes-Formel ist mindestens exakt vom Grad n, da $f\in P_n \Rightarrow p_n^f=f$.

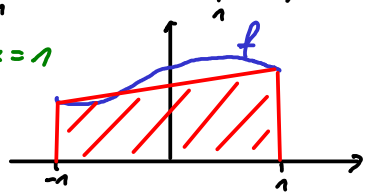
Bem: Wegen $\int_a^b f(x)\,dx=\frac{b-a}{2}\int_{-1}^{1} f(T(x))\,dx$ mit $T(x)=a+(b-a)\frac{x+1}{2}$ reicht es/ist es üblich, die Stützstellen und Gewichte einer Quadraturformel nur für $[-1,1]$ anzugeben.

Bsp: Sei $[a,b]=[-1,1]$.

- $x_0=0$, $a_0=\int_{-1}^{1}\ell_0(x)\,dx=\int_{-1}^{1}1\,dx=2$
  $\Rightarrow I_0(f)=2f(0)$ heißt **Mittelpunktregel/Rechteckregel**



- $x_0=-1$, $x_1=1$, $a_0=\int_{-1}^{1}\ell_0(x)\,dx=\int_{-1}^{1}\frac{x-1}{-1-1}\,dx=1$, $a_1=\int_{-1}^{1}\frac{x+1}{1+1}\,dx=1$
  $\Rightarrow I_1(f)=f(-1)+f(1)$ heißt **Trapezregel**



- $-1=x_0<\dots<x_n=1$ gleichverteilt:

n	$(a_0\ \dots\ a_n)$	Name	Exaktheitsgrad	$I_n(\cos\frac{\pi}{2}x)$
1	$(1\ 1)$	Trapezregel	1	0
2	$(1\ 4\ 1)/3$	Simpson-/Keplersche Fassregel	3	4/3
3	$(1\ 3\ 3\ 1)/4$	Newtonsche $\frac{3}{8}$-Regel/Pulcherrima	3	1,2990...
5	$(19\ 75\ 50\ 50\ 75\ 19)/144$	Milne-Regel	5	1,2727...
				$\approx\frac{4}{\pi}=1{,}2732\dots$

für großes n können Gewichte negativ werden!

Offenbar kann der Exaktheitsgrad von I_n auch größer als n sein – was weiß man darüber?

Thm: Sei I_n eine Quadraturformel zu Stützstellen $x_0, \dots, x_n$ mit Exaktheitsgrad m.

(1) Ist $m \geq n$, so ist I_n die Newton-Cotes-Formel. (2) $m \leq 2n+1$.

Bew: (1) Das Lagrange-Polynom ℓ_i hat Grad n, also $\int_a^b \ell_i \, dx = I_n(\ell_i) = \sum_{j=0}^{n} a_j \ell_i(x_j) = a_i$.

(2) Das quadrierte Knotenpolynom $(\omega_{n+1}(x))^2 = \prod_{i=0}^{n} (x-x_i)^2$ hat Grad $2n+2$ und erfüllt $I(\omega_{n+1}^2) - I_n(\omega_{n+1}^2) = \int_a^b \omega_{n+1}(x)^2 dx - \sum_{i=0}^{n} a_i \omega_{n+1}(x_i)^2 = \int_a^b \omega_{n+1}(x)^2 dx > 0$. □

Können wir eine Newton-Cotes-Formel I_n (also insbesondere Stützstellen) von Exaktheitsgrad $2n+1$ finden?

Sei $p \in P_{2n+1}$ und $\omega_{n+1}(x) = \prod_{i=0}^{n} (x-x_i)$ das Knotenpolynom. Mittels Euklidscher Division gilt

$$p = q_n \omega_{n+1} + r_n \quad \text{mit } q_n, r_n \in P_n.$$

Schreiben wir $R_n(f) = I_n(f) - I(f)$ für den Quadraturfehler, so erhalten wir

$$R_n(p) = R_n(q_n \omega_{n+1}) + R_n(r_n) = R_n(q_n \omega_{n+1}) = \sum_{i=0}^{n} a_i q_n(x_i) \omega_{n+1}(x_i) - \int_a^b q_n \omega_{n+1} dx = -\int_a^b q_n \omega_{n+1} dx$$

Thm: (L^2-Skalarprodukt) Die Abbildung $p, q \mapsto \langle p, q \rangle = \int_a^b p(x) q(x) dx$ definiert ein Skalarprodukt auf dem Vektorraum $C^0([a,b])$ der stetigen Funktionen auf $[a,b]$.

Wenn das Knotenpolynom ω_{n+1} orthogonal zu allen $q_n \in P_n$ bzgl. des obigen Skalarprodukts ist, ist $R_n(p) = 0 \; \forall p \in P_{2n+1}$ und somit I_n exakt von Ordnung $2n+1$. Ein solches Polynom ω_{n+1} erhält man durch Gram-Schmidt-Orthogonalisierung der $m_i(x) = x^i$, $i = 0, 1, 2, \dots$,

d.h. wir setzen $\tilde{\omega}_i = m_i - \sum_{j=0}^{i-1} \langle m_i, \omega_j \rangle \omega_j$, $\omega_i = \frac{\tilde{\omega}_i}{\langle \tilde{\omega}_i, \tilde{\omega}_i \rangle}$.

Thm: Sei $[a,b] = [-1,1]$. Das Gram-Schmidt-Verfahren auf $m_0, m_1, m_2, \dots$ liefert

(1) für $\langle \cdot, \cdot \rangle$ die Legendre-Polynome $\omega_i(x) = \frac{i!}{(2i)!} \frac{d^i}{dx^i} [(x^2-1)^i]$,

(2) für $\langle p, q \rangle_T := \int_{-1}^{1} \frac{p(x) q(x)}{\sqrt{1-x^2}} dx$ die Čebyšev-Polynome $\omega_i(x) = T_i(x)$.

Bew: Offensichtlich ist in beiden Fällen $\omega_i \in \operatorname{span}(m_0, \dots, m_i) = P_i$, somit ist nur noch zu zeigen, dass $\langle \omega_i, p \rangle = 0$ bzw. $\langle \omega_i, p \rangle_T = 0 \; \forall p \in P_{i-1}$.

(1) Sei $\Omega_i^0 = \omega_i$, $\Omega_i^j(x) = \int_{-1}^{x} \Omega_i^{j-1}(t) dt = \frac{i!}{(2i)!} \frac{d^{i-j}}{dx^{i-j}} [(x^2-1)^i]$ für $j = 1, \dots, i$,

dann hat Ω_i^j in 1 und -1 eine j-fache Nullstelle.

$$\langle \omega_i, p \rangle = \int_{-1}^{1} \omega_i(x) p(x) dx = \left[\Omega_i^1(x) p(x) \right]_{-1}^{1} - \int_{-1}^{1} \Omega_i^1(x) p^{(1)}(x) dx = -\int_{-1}^{1} \Omega_i^1(x) p^{(1)}(x) dx$$
$$= \dots = (-1)^i \int_{-1}^{1} \Omega_i^i(x) p^{(i)}(x) dx = 0$$

(2) war bereits Hausaufgabe □

Wenn wir also Stützstellen $x_0, \dots, x_n$ wählen, sodass das Knotenpolynom ω_{n+1} gleich dem $(n+1)$-ten Legendre-Polynom ist, dann erhalten wir die gesuchte Quadraturformel mit Exaktheitsgrad $2n+1$. Allerdings ist das Legendre-Polynom nur ein zulässiges Knotenpolynom, wenn es $n+1$ paarweise verschiedene Nullstellen $x_0, \dots, x_n$ hat. Dies ist der Fall:

Thm: (Nullstellen orthogonaler Polynome) Sei $\omega: (-1,1) \to (0,\infty)$ ein Gewicht und $\langle p, q \rangle_\omega = \int_{-1}^{1} \omega(x) p(x) q(x) dx$ ein Skalarprodukt. Sei $\omega_i \in P_i$ mit $\langle \omega_i, p \rangle_\omega = 0 \ \forall p \in P_{i-1}$, dann hat ω_i genau i verschiedene Nullstellen in $(-1,1)$.

Bew: Angenommen, ω_i habe weniger Nullstellen, d.h. insbesondere ändert es sein Vorzeichen nur an $m < i$ Stellen $x_0, \dots, x_{m-1}$. Dann ist $q(x) = \prod_{j=0}^{m-1} (x - x_j) \in P_m$, und $q\omega_i$ hat konstantes Vorzeichen.

$$\Rightarrow \langle q, \omega_i \rangle_\omega = \int_{-1}^{1} \omega q \omega_i \, dx > 0 \quad \text{außer für } \omega_i = 0 \quad \Rightarrow \text{Widerspruch} \qquad \square$$

Def: (Gauß-Quadratur) Die Quadraturformel von Exaktheitsgrad $2n+1$, deren Stützstellen die Nullstellen des $(n+1)$-ten Legendre-Polynoms sind, heißt Gauß-Quadratur.

Bem: Analog kann man Gauß-Quadraturformeln für gewichtete Integrale $I_\omega(f) = \int_a^b \omega(x) f(x) dx$ finden (Hausaufgabe).

Die Gauß-Quadratur hat außerdem positive Gewichte, was wichtig für numerische Stabilität ist.

Thm: (Gauß-Gewichte) Die Gewichte a_i der Gauß-Quadratur auf $[-1,1]$ erfüllen

$$a_i = \int_{-1}^{1} (l_i(x))^2 dx > 0 \qquad \text{für das Lagrangepolynom } l_i(x) = \prod_{j=0}^{n} \frac{x - x_j}{x_i - x_j}.$$

Bew: $l_i^2 \in P_{2n}$, daher ist die Gauß-Quadratur exakt mit

$$\int_{-1}^{1} (l_i(x))^2 dx = \sum_{j=0}^{n} a_j \, l_i(x_j)^2 = a_i. \qquad \square$$

Bsp: Gauß-Quadratur-Formeln:

n	$(x_0, \dots, x_n)$
0	0
1	$-\sqrt{3}/3 \quad \sqrt{3}/3$
2	$-\sqrt{3/5} \quad 0 \quad \sqrt{3/5}$

Schließlich möchten wir wissen, wie groß der Quadraturfehler ist.

Thm: (Quadraturfehler) Die Newton-Cotes-Formel mit Stützstellen $x_0, \dots, x_n \in [a,b]$ erfüllt mit $\|q\|_\infty := \sup_{x \in [a,b]} |q(x)|$

$$|I_n(f) - I(f)| \le \int_a^b f[x_0, \dots, x_n, x] \, \omega_{n+1}(x) dx \le \|f[x_0, \dots, x_n, x]\|_\infty \int_a^b |\omega_{n+1}(x)| dx$$

$$\le \frac{\|f^{(n+1)}\|_\infty}{(n+1)!} \int_a^b |\omega_{n+1}(x)| dx \qquad \text{falls } f \in C^{n+1}([a,b]).$$

Bew: Folgt aus $I_n(f) - I(f) = \int_a^b p_n^f - f \, dx$ für das Interpolationspolynom p_n^f und der Darstellung des Interpolationsfehlers. $\square$

Für Formeln mit Exaktheitsgrad $> n$ erwartet man noch kleinere Fehler – um dies zu sehen, reicht obige Darstellung nicht. Stattdessen benötigen wir eine explizitere Fehlerdarstellung (wie oben als ein Integral), aus der wir f mit dem erweiterten Mittelwertsatz herausziehen können. Wir gehen das Verfahren für Gauß-Quadraturen durch. Es ist ähnlich zu der Peano-Darstellung dividierter Differenzen.

Def: (Monospline) Seien $a_0, \dots, a_n$ die Gewichte zu Stützstellen $x_0, \dots, x_n \in [a,b]$ einer Quadratur mit Exaktheitsgrad $\geq k$. Der k-te **Monospline** ist $M_{k,n}(t) = (b-t)_+^{k+1} - (a-t)_+^{k+1} - (k+1)\sum_{i=0}^{n} a_i (x_i - t)_+^k$.

Thm: (1) $M_{k,n} = 0$ auf $\mathbb{R} \setminus [a,b]$

(2) Ist $x_0 = a$, so hat $M_{k,n}$ in a eine k-fache Nullstelle, ist $x_0 > a$, eine $(k+1)$-fache Nullstelle

(3) Analoges gilt für b.

Bew: Hausaufgabe □

Thm: (Peanodarstellung des Quadraturfehlers) Sei I_n eine Quadraturformel mit Stützstellen $x_0, \dots x_n \in [a,b]$ und Exaktheitsgrad $\geq k$, so ist der Quadraturfehler für $f \in C^{k+1}([a,b])$ gegeben durch

$$I_n(f) - I(f) = -\int_a^b f^{(k+1)}(t) \frac{M_{k,n}(t)}{(k+1)!}\, dt.$$

Bew: Partielle Integration liefert für $x \in [a,b]$

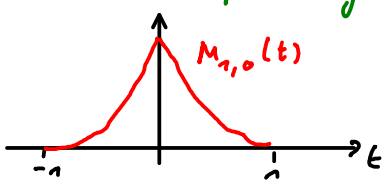
$$\int_a^b f^{(k+1)}(t)(x-t)_+^m\, dt = \Big[f^{(k)}(t)(x-t)^m \Big]_a^x + m\int_a^x f^{(k)}(t)(x-t)^{m-1}\, dt = f^{(k)}(a)(a-x) + m\int_a^b f^{(k)}(t)(x-t)_+^{m-1}\, dt,$$

somit

$$\begin{aligned}
\int_a^b f^{(k+1)}(t) \frac{M_{k,n}(t)}{(k+1)!}\, dt &= -f^{(k)}(a)\frac{M_{k,n}(a)}{(k+1)!} + \int_a^b f^{(k)}(t)\frac{M_{k-1,n}(t)}{k!}\, dt \\
&= \int_a^b f^{(k)}(t)\frac{M_{k-1,n}(t)}{k!}\, dt = \dots = \int_a^b f'(t)\frac{M_{0,n}(t)}{1!}\, dt \\
&= \int_a^b f'(t)(b-t) - \sum_{i=0}^{n} a_i f'(t)(x_i - t)_+^0\, dt \\
&= \Big[f(t)(b-t) \Big]_a^b + \int_a^b f(t)\, dt - \sum_{i=0}^{n} a_i (f(x_i) - f(a)) \\
&= f(a)\underbrace{\Big[a - b + \sum_{i=0}^{n} a_i \Big]}_{= I_n(1) - I(1) = 0} + I(f) - I_n(f)
\end{aligned}$$

□

Bsp: · Für die Mittelpunktregel auf $[-1,1]$ ist $n=0$, $k=1$, $M_{k,n}(t) = (1-t)_+^2 + (-1-t)_+^2 - 2\cdot 2\cdot (0-t)_+^1$



$\Rightarrow$ Ist $f \in C^2([-1,1])$, gilt für ein $\xi \in [-1,1]$:

$$I(f) - I_0(f) = \int_{-1}^{1} f'(t)\frac{M_{1,0}(t)}{2}\, dt = \frac{f''(\xi)}{2}\int_{-1}^{1} M_{1,0}(t)\, dt = \frac{f''(\xi)}{3}$$

· Für die Trapezregel auf $[-1,1]$ ist $k = n = 1$, $M_{1,1}(t) = (1-t)_+^2 + (-1-t)_+^2 - 2\big[(-1-t)_+ + (1-t)_+\big] = \min(0, t^2 - 1)$

$\Rightarrow$ für $f \in C^2([-1,1])$ gilt $I(f) - I_1(f) = \int_{-1}^{1} f''(t)\frac{t^2-1}{2}\, dt$

· Man kann zeigen, dass $M_{2n+1,n} \geq 0$ und $\int_{-1}^{1} M_{2n+1,n}(t)\, dt = \frac{2^{2n+3}}{2n+3}\frac{[(n+1)!]^4}{[(2n+2)!]^2}$

$$\Rightarrow I(f) - I_n(f) = \int_{-1}^{1} f^{(2n+2)}(t)\frac{M_{2n+1,n}(t)}{(2n+2)!}\, dt = \frac{2^{2n+3}}{2n+3}\frac{[(n+1)!]^4}{[(2n+2)!]^3} f^{(2n+2)}(\xi) \text{ für ein } \xi \in [-1,1].$$

Summierte Quadraturregeln

Wie schon bei der Interpolation ist es oft genauer (z.B. wenn die zu integrierende Funktion nicht oft differenzierbar ist oder die Ableitungen stark anwachsen, sodass Runges Phänomen auftritt), statt Polynomen stückweise Polynome für die Quadratur zu nutzen. Hierzu teilt man $[a,b]$ in N kleine Teilintervalle auf und approximiert auf jedem das Integral mit Quadratur.

Sei z.B. $I_n(f) = \sum_{i=0}^{n} a_i f(x_i)$ eine Quadraturformel auf $[-1,1]$, dann ist
$I_n^{cd}(f) = \frac{d-c}{2} \sum_{i=0}^{n} a_i f\left(c + \frac{x_i+1}{2}(d-c)\right)$ eine Quadratur auf $[c,d]$. Setzen wir nun

$$z_i = a + hi \quad \text{für } h = \frac{b-a}{N},\ i = 0,\dots,N, \quad x_i^j = z_i + \frac{h}{2}(x_j+1),\ j = 0,\dots,n,$$

so ist $I(f) = \int_a^b f(x)\,dx \approx \frac{h}{2} \sum_{i=0}^{N-1} \sum_{j=0}^{n} a_j f(x_i^j) =: I_n^N(f)$.

Bsp: Für die Trapezregel ist $x_i^0 = z_i = a+hi$, $x_i^1 = z_{i+1} = a+h(i+1)$, $a_0 = a_1 = 1$

$$\Rightarrow I_1^N(f) = \frac{h}{2}\left[f(x_0^0) + f(x_0^1) + f(x_1^0) + f(x_1^1) + \dots + f(x_{N-1}^0) + f(x_{N-1}^1)\right] = \frac{(b-a)}{N}\left[\frac{f(a)+f(b)}{2} + \sum_{i=1}^{N-1} f(z_i)\right]$$

Thm: (Fehler summierter Formeln) Sei I_n eine Quadraturformel auf $[-1,1]$ mit Exaktheitsgrad k, $f \in C^{k+1}([a,b])$. Für den Fehler der summierten Quadraturformel I_n^N gilt

$$|I_n^N(f) - I(f)| \le \max_{x\in[a,b]} |f^{(k+1)}(x)| \frac{(h/2)^{k+1}}{2(k+1)!} \int_{-1}^{1} |M_{k,n}(t)|\,dt$$

für $h = \frac{b-a}{N}$ und $M_{k,n}$ den Monosplines zu I_n.

Bew: Sei $I_n(f) = \sum_{i=0}^{n} a_i f(x_i)$ und sei $z_i = a + hi$ und $M_{k,n}^i$ der Monospline zur Quadraturformel auf $[z_i, z_{i+1}]$, also $M_{k,n}^i = (z_{i+1}-t)_+^{k+1} - (z_i - t)_+^{k+1} - (k-1)\sum_{i=0}^{n} a_i \frac{h}{2}\left(z_i + \frac{h}{2}(x_i+1)\right)_+^k$. Es ist

$$|I(f) - I_n^N(f)| = \left|\sum_{i=0}^{N} I(f|_{[z_i,z_{i+1}]}) - I_n(f|_{[z_i,z_{i+1}]})\right| = \left|\sum_{i=0}^{N} \int_{z_i}^{z_{i+1}} \frac{f^{(k+1)}(t)}{(k+1)!} M_{k,n}^i(t)\,dt\right|$$

$$\le \max_{x\in[a,b]} |f^{(k+1)}(x)| \frac{1}{(k+1)!} \sum_{i=0}^{N} \int_{z_i}^{z_{i+1}} |M_{k,n}^i(t)|\,dt$$

Mit der Variablentransformation $t = z_i + \frac{h}{2}(x+1)$ folgt $M_{k,n}^i(t) = \left(\frac{h}{2}\right)^{k+1} M_{k,n}(x)$, sodass

$$\sum_{i=0}^{N} \int_{z_i}^{z_{i+1}} |M_{k,n}^i(t)|\,dt = \sum_{i=0}^{N} \int_{-1}^{1} \left(\frac{h}{2}\right)^{k+1} |M_{k,n}(x)| \frac{h}{2}\,dx = \left(\frac{h}{2}\right)^{k+1} \frac{1}{2} \int_{-1}^{1} |M_{k,n}(x)|\,dx. \qquad \square$$

Der Fehler ist offenbar eine Potenz von h (bzw. hat vielleicht sogar eine höhere Taylorentwicklung in h), somit kann der Fehler mit Richardson-Extrapolation verkleinert werden. Dies ist die Idee in Folgendem.

Alg: (Romberg-Quadratur) Sei $f \in C^{2m+2}([a,b])$ und I_1^N für $N = \frac{b-a}{h}$ die summierte Trapezregel, d.h.

$$I_1^N(f) = h\left[\frac{f(a)+f(b)}{2} + \sum_{i=1}^{N-1} f(a+ih)\right] =: g(z) \quad \text{für } z = h^2$$

Berechne $g(z), g(\frac{z}{4}), \dots, g(\frac{z}{4^m})$ (also die Quadraturen zu $h, \frac{h}{2}, \dots, \frac{h}{2^m}$)

und approximiere $g(0) = I(f)$ mittels Richardson-Extrapolation $\tilde{g}(0)$.

Bem: Insgesamt muss f nur an $2^m \frac{b-a}{h}$ Stellen ausgewertet werden (den Stützstellen zu $\frac{h}{2^m}$), da die Stützstellen zu $h, \frac{h}{2}, \dots, \frac{h}{2^{m-1}}$ eine Teilmenge hiervon bilden.

Thm: (Romberg-Quadratur) Für die Romberg-Quadratur gilt

$$|\tilde{g}(0) - I(f)| = O(h^{2m+2})$$

Bew: Folgt aus Fehlerabschätzung für Richardson-Extrapolation, wenn wir zeigen können, dass $g(z)$ eine Taylorentwicklung $g(0) + \sum_{i=1}^{m} a_i z^i + O(z^{m+1})$ hat, also $I_1^{\frac{b-a}{h}}(f) = I(f) + \sum_{i=1}^{m} a_i h^{2i} + O(h^{2m+2})$. Dies trifft zu, da

$$I(f) - I_1^N(f) = \sum_{i=0}^{N-1} \int_{z_i}^{z_{i+1}} f''(x) \frac{M_1^i(x)}{2}\,dx = \sum_{i=0}^{N-1} \int_{z_i}^{z_{i+1}} f''(x) \frac{(x-z_i)(x-z_{i+1})}{2}\,dx$$

$$= \sum_{i=0}^{N-1} \int_{z_i}^{z_{i+1}} f''(x) \frac{(x-z_i)(x-z_{i+1}) + h^2/6}{2}\,dx - \frac{h^2}{12}\int_a^b f''(x)\,dx = \sum_{i=0}^{N-1} \int_{-h/2}^{h/2} f''\left(x + \frac{z_i + z_{i+1}}{2}\right) \frac{x^2 - h^2/12}{2}\,dx - \frac{h^2}{12}[f'(b) - f'(a)]$$

Sei nun $p_2(x) = \frac{x^2 - h^2/12}{2}$, $p_{i+1}'(x) = p_i(x)$ mit $\int_{-h/2}^{h/2} p_{i+1}(x)\,dx = 0$.

Offenbar ist p_{2i} gerade ($p_{2i}(x) = p_{2i}(-x)$) und p_{2i+1} ungerade ($p_{2i+1}(x) = -p_{2i+1}(-x)$) $\forall i$, außerdem $p_k(-\frac{h}{2}) - p_k(\frac{h}{2}) = \int_{-h/2}^{h/2} p_k'(x)\,dx = 0 \quad \forall k$.

Mittels $(2m)$facher partieller Integration folgt

$$I(f) - I_1^N(f) = \sum_{i=0}^{N-1} \Big[f''\left(x + \tfrac{z_i + z_{i+1}}{2}\right) p_3(x) - f'''\left(x + \tfrac{z_i + z_{i+1}}{2}\right) p_4(x) + \dots - f^{(2m+1)}\left(x + \tfrac{z_i + z_{i+1}}{2}\right) p_{2m+1}(x) \Big]_{x=-h/2}^{h/2}$$

$$- \int_{-h/2}^{h/2} f^{(2m+2)}\left(x + \tfrac{z_i + z_{i+1}}{2}\right) p_{2m+2}(x)\,dx - \frac{h^2}{12}[f'(b) - f'(a)]$$

$$= \sum_{j=1}^{m} \left[f^{(2j+1)}(a)\, p_{2j+2}(-\tfrac{h}{2}) - f^{(2j+1)}(b)\, p_{2j+2}(\tfrac{h}{2}) \right] - \sum_{i=0}^{N-1} \int_{-h/2}^{h/2} f^{(2m+2)}\left(x + \tfrac{z_i + z_{i+1}}{2}\right) p_{2m+2}(x)\,dx$$

$$= c_1 h^2 + c_2 h^4 + \dots + c_m h^{2m} + O(h^{2m+2})$$ ☐

Bem: Ist f periodisch, d.h. $f^{(k)}(a) = f^{(k)}(b)$, $k = 0, \dots, 2m+2$, dann zeigt obige Rechnung wegen $p_{2j}(-h) = p_{2j}(h)$ sogar $I(f) - I_1^N(f) = O(h^{2m+2})$. Ist f analytisch, konvergiert $I_1^N(f)$ also sogar schneller gegen $I(f)$ als jede Potenz von h!

Stabilität der Quadratur

Erinnerung: Bei numerischen Verfahren machen wir immer Diskretisierungsfehler (z.B. durch Wahl endlich vieler Stützstellen) und einen Rundungsfehler (durch gerundete Eingabedaten oder Maschinenoperationen).

Dass der Diskretisierungsfehler durch geeignete Parameter beliebig klein gemacht werden kann, nennt man **Konsistenz** (z.B. ist Polynominterpolation analytischer Funktionen mit äquidistanten Stützstellen konsistent, nicht jedoch für alle Funktionen wegen des Runge-Phänomens).

Dass die akkumulierten Rundungsfehler nicht größer werden als der unvermeidbare Fehler,

den man bei exakter, nicht diskretisierter Rechnung macht, wenn die Eingabedaten in gleichem Maße gestört werden, nennt sich Stabilität. Bisher haben wir nur Stabilität von Verfahren betrachtet, die keine Diskretisierungsfehler hatten. In diesen Untersuchungen haben wir vor allem die Auswirkungen von Maschinenoperationen betrachtet. Bei allen komplexeren numerischen Verfahren wollen wir nun annehmen, dass alle Rechnungen stabil implementiert sind, sodass wir für die Stabilität des ganzen Verfahrens nur die Verstärkung von Eingangsdatenfehlern betrachten müssen (also als wenn keine Maschinenoperationen, sondern exakt Operationen durchgeführt würden).

Thm: (Kondition) Ist $\tilde{f}$ eine Näherung an $f: [a,b] \to \mathbb{R}$ mit $\frac{|f-\tilde{f}|}{|f|} \leq \varepsilon$ auf $[a,b]$, dann ist $\frac{|I(f)-I(\tilde{f})|}{I(|f|)} \leq \varepsilon$.

Bew: $|I(\tilde{f}) - I(f)| \leq \int_a^b |\tilde{f}-f|\,dx \leq \varepsilon \int_a^b |f|\,dx = \varepsilon I(|f|)$. □

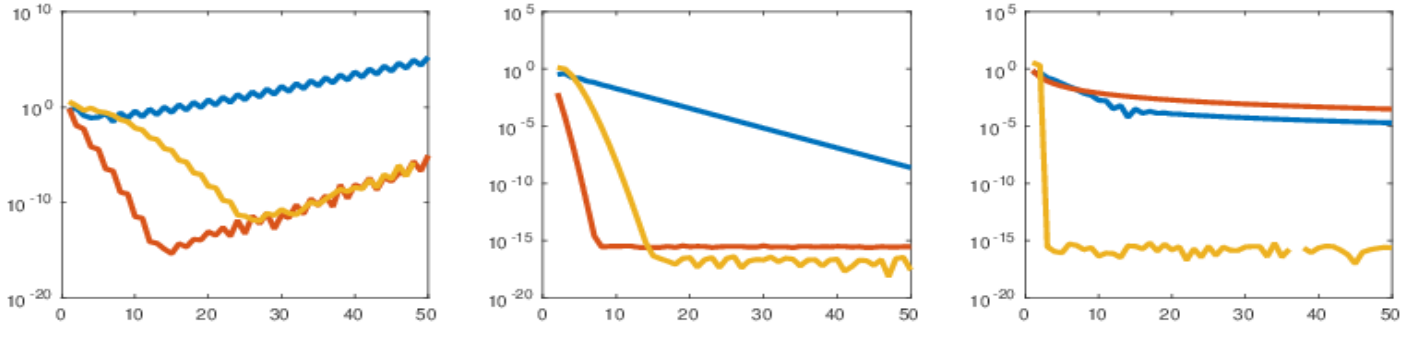
Thm: (Stabilität) Konsistente Quadraturformeln mit positiven Gewichten sind stabil.

Bew: Sei $\tilde{f}$ eine Näherung an f mit $\frac{|f-\tilde{f}|}{|f|} \leq \varepsilon$, die Quadraturformel I_n habe Stützstellen $x_0, \dots, x_n$ und Gewichte $a_0, \dots, a_n > 0$. Wir müssen zeigen, dass $|I_n(\tilde{f}) - I_n(f)|$ nicht größer als der unvermeidbare Fehler $I(|f|)\varepsilon$ ist:

$$|I_n(\tilde{f}) - I_n(f)| = \left| \sum_{i=0}^{n} a_i (\tilde{f}(x_i) - f(x_i)) \right| \leq \varepsilon \sum_{i=0}^{n} |a_i|\,|f(x_i)| = \varepsilon\, I_n(|f|) \overset{\text{Konsistenz}}{\approx} \varepsilon I(|f|). \quad \square$$

```
function exampleQuadrature
  % Integranden
  f = cell(3);
  f{1} = @(x) 1./(1+25*x.^2);
  f{2} = @(x) exp(x);
  f{3} = @(x) sin(x*pi+1)-cos(2*x*pi);
  % Integral auf [-1,1]
  val = [2*atan(5)/5 exp(1)-exp(-1) 0];

  % geschlossene Newton-Cotes-Quadratur
  n = 50;
  error = zeros(3,n);
  for i = 1:n
    % Stuetzstellen
    x = linspace(-1,1,i+1);
    % Gewichte
    weights = computeQuadratureWeights( x, -1, 1 );
    % Quadratur
    for j = 1:3
      error(j,i) = abs(weights*f{j}(x')-val(j));
    end
  end
  subplot(1,3,1);
  semilogy(1:n,error,'Linewidth',3);
```

```
  % Gauss-Quadratur
  error = zeros(3,n);
  for j = 2:n
    % Legendre-Polynom-Nullstellen
    syms x;
    roots = vpasolve( legendreP(j,x) == 0 );
    % Gewichte
    weights = computeQuadratureWeights( roots, -1, 1 );
    % Quadratur
    for i = 1:3
      error(i,j) = abs(weights*f{i}(roots)-val(i));
    end
  end
  subplot(1,3,2);
  semilogy(1:n,error,'Linewidth',3);

  % summierte Trapezregel
  error = zeros(3,n);
  for j = 1:n
    % Stuetzstellen
    x = linspace(-1,1,j+1);
    % Quadratur
    for i = 1:3
      q = (sum(f{i}(x(2:end-1)))+f{i}(x(1))/2+f{i}(x(end))/2)*2/j;
      error(i,j) = abs(q-val(i));
    end
  end
  subplot(1,3,3);
  semilogy(1:n,error,'Linewidth',3);
end

% Quadratur-Gewicht-Berechnung
function weights = computeQuadratureWeights( knots, a, b )
  n = numel(knots)-1;
  weights = zeros(1,n+1);
  syms t lagrangeP;
  for j = 1:n+1
    % Lagrange Polynom
    lagrangeP = 1;
    for l = 1:n+1
      if ( l ~= j )
        lagrangeP = lagrangeP * (t-knots(l)) / (knots(j)-knots(l));
      end
    end
    % Integral des Lagrange Polynoms
    weights(j) = int(lagrangeP,'t',a,b);
```



heute: · Numerische Differentiation
· gewöhnliche Differentialgleichungen

Numerische Differentiation

Nun möchten wir die Ableitung $f^{(k)}(x)$ einer Funktion anhand weniger Messwerte bestimmen. Hierzu können wir wieder das Interpolationspolynom p zu Stützstellen $x_0, \dots, x_n$ berechnen und dann $f^{(k)}(x) \approx p^{(k)}(x)$ approximieren.

Thm: (Differentiation Fehlerordnung) Sei $f \in C^{n+1}(\mathbb{R})$, $p \in P_n$ das Interpolationspolynom zu f an den paarweise verschiedenen Stützstellen $x_0, \dots, x_n \in I \subset \mathbb{R}$, $x \in I$. Für $k = 0, \dots, n$ gilt

$$\left| f^{(k)}(x) - p^{(k)}(x) \right| = \left\| \left(V^{-1}\right)_{k+1,:} \right\| O\left(\operatorname{diam}(I)^{n+1}\right)$$

mit V der Vandermonde-Matrix zu $x_0 - x, \dots, x_n - x$.

Bew: OBdA sei $x = 0$ (sonst führe die Variablentransformation $t \mapsto t - x$ durch).

Sei $p(t) = \sum_{i=0}^{n} a_i t^i$ mit $\begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = V^{-1} \begin{pmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{pmatrix}$.

Es ist $p^{(k)}(x) = k!\, a_k = k! \left(V^{-1}\right)_{k+1,:} \begin{pmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{pmatrix}$.

Setzen wir die Taylorentwicklung $f(x_\ell) = \sum_{j=0}^{n} \frac{f^{(j)}(x)}{j!} x_\ell^j + O\left(x_\ell^{n+1}\right)$ ein, erhalten wir

$$\begin{aligned} p^{(k)}(x) &= k! \sum_{\ell=0}^{n} \left(V^{-1}\right)_{k+1,\ell+1} \left(\sum_{j=0}^{n} \frac{f^{(j)}(x)}{j!} x_\ell^j + O\left(x_\ell^{n+1}\right) \right) \\ &= k! \sum_{j=0}^{n} \frac{f^{(j)}(x)}{j!} \sum_{\ell=0}^{n} \left(V^{-1}\right)_{k+1,\ell+1} x_\ell^j + \left\| \left(V^{-1}\right)_{k+1,:} \right\| O\left(\operatorname{diam}(I)^{n+1}\right) \\ &= k! \frac{f^{(k)}(x)}{k!} + \left\| \left(V^{-1}\right)_{k+1,:} \right\| O\left(\operatorname{diam}(I)^{n+1}\right) \end{aligned}$$

□

Bem: Wählen wir Stützstellen $x_k = x + h z_k$ für $h > 0$, $z_0, \dots, z_n \in [-a, a]$, so ist $\operatorname{diam} I = 2ah$ und $V = \left(\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \middle| h \begin{pmatrix} z_0 \\ \vdots \\ z_n \end{pmatrix} \middle| h^2 \begin{pmatrix} z_0^2 \\ \vdots \\ z_n^2 \end{pmatrix} \middle| \dots \middle| h^n \begin{pmatrix} z_0^n \\ \vdots \\ z_n^n \end{pmatrix} \right)$, $V^{-1} = \begin{pmatrix} — w_0 — \\ — w_1/h — \\ \vdots \\ — w_n/h^n — \end{pmatrix}$ für Vektoren $w_0, \dots, w_n \in \mathbb{R}^{n+1}$,

sodass $\left| f^{(k)}(x) - p^{(k)}(x) \right| = \|w_k\| O\left(h^{n+1-k}\right)$ $\quad p(z) = \left(z - x, (z-x)^2, \dots, (z-x)^n\right) V^{-1} \begin{pmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{pmatrix}$

bzw. $f^{(k)}(x) = p^{(k)}(x) + \|w_k\| O\left(h^{n+1-k}\right) = \frac{k!}{h^k} \sum_{i=0}^{n} w_{ki} f(x_i) + O\left(h^{n+1-k}\right)$.

Die w_{ki} kann man auch erhalten durch Taylorentwicklung der $f(x_i)$ um x und Koeffizientenvergleich auf beiden Seiten.

Bsp: · berechne $f'(x)$ mittels $f(x)$, $f(x+h)$

$$\Rightarrow f'(x) \approx \frac{w_{10} f(x) + w_{11} f(x+h)}{h} \overset{\text{Taylor}}{=} \frac{w_{10} f(x) + w_{11} f(x) + w_{11} h f'(x)}{h} + w_{11} O(h)$$

$$\Rightarrow 0 = w_{10} + w_{11}, \quad 1 = w_{11}$$

$$\Rightarrow f'(x) = \frac{f(x+h) - f(x)}{h} + O(h) \quad \text{für } f \in C^2(\mathbb{R}) \qquad \text{„Vorwärtsdifferenzenquotient“}$$

· analog: $f'(x) = \frac{f(x) - f(x-h)}{h} + O(h)$ „Rückwärtsdifferenzenquotient“

· $f'(x) = \frac{w_{10} f(x-h) + w_{11} f(x+h)}{h} = \frac{w_{10}\left(f(x) - f'(x)h + f''(x)/2\, h^2\right) + w_{11}\left(f(x) + f'(x)h + f''(x)/2\, h^2\right)}{h} + O(h^2)$

$\Rightarrow w_{11} = -w_{10} = \frac{1}{2} \Rightarrow f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2)$ für $f \in C^3(\mathbb{R})$ (sogar besser als h!) „zentraler Differenzenquotient“

· analog: $f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2)$ für $f \in C^4(\mathbb{R})$

Bem: · Ist $\hat{f}$ eine leichte Störung der Eingangsfunktion f bzgl. der Norm $\|f\|_{C^1} = \max_{x \in [a,b]} \left(|f(x)| + |f'(x)| \right)$

d.h. $\|\hat{f} - f\|_{C^1} \leq \varepsilon$, so ist auch $|f'(x) - \hat{f}'(x)| \leq \varepsilon$, die Berechnung von $f'(x)$ ist also gut konditioniert.

· Ist $\hat{f}$ jedoch eine Störung bzgl. $\|f\|_{C^0}$, d.h. $\|\hat{f} - f\|_{C^0} \leq \varepsilon$, so ist $\hat{f}'(x)$ ggfs. noch nicht einmal definiert, die Kondition also unendlich schlecht.

(die Wahl der Normen spielt bei Konditions- und Stabilitätsuntersuchungen immer eine Rolle!)

⇒ wird in einem Algorithmus irgendwo die Ableitung einer Funktion berechnet/approximiert und können die Funktionswerte störungsbehaftet sein, ist der Algorithmus höchstwahrscheinlich instabil.

Numerik gewöhnlicher Differentialgleichungen

Gewöhnliche & partielle Differentialgleichungen gehören zu den Hauptwerkzeugen der mathematischen Modellierung von physikalischen, biologischen, ..., ökonomischen Prozessen, können aber selten/st analytisch gelöst werden => numerische Verfahren sind notwendig.

Gewöhnliche Differentialgleichungen

Def: (Gewöhnliche Differentialgleichung). Sei $I \subset \mathbb{R}$ ein Intervall, $F: I \times (\mathbb{R}^n)^{m+1} \to \mathbb{R}^n$, $f: I \times (\mathbb{R}^n)^m \to \mathbb{R}^n$.

Eine <u>implizite gewöhnliche Differentialgleichung der Ordnung m</u> ist eine Gleichung der Form

$$0 = F(t, y(t), y'(t), \dots, y^{(m)}(t)),$$

zu lösen für die Funktion $y: I \to \mathbb{R}^n$, $y \in C^m(I)$. Eine <u>explizite gDgl</u> ist von der Form

$$y^{(m)}(t) = f(t, y(t), y'(t), \dots, y^{(m-1)}(t)).$$

Hängen F oder f nicht von t ab, heißt die gDgl. <u>autonom</u>, hängen f, F nicht von $y(t), \dots, y^{(m-1)}(t)$ ab, heißt die gDgl. <u>einfach</u>.

Bem: Eine Differentialgleichung $y^{(m)} = f(t, y, y', \dots, y^{(m-1)})$ m-ter Ordnung kann in eine äquivalente Differentialgleichung erster Ordnung umgewandelt werden mittels $Y_i = y^{(i)}$, $i = 0, \dots, m-1$,

$$Y = \begin{pmatrix} Y_0 \\ \vdots \\ Y_{m-1} \end{pmatrix}: \qquad Y'(t) = \begin{pmatrix} Y_1(t) \\ \vdots \\ Y_{m-1}(t) \\ f(t, Y_0(t), \dots, Y_{m-1}(t)) \end{pmatrix}$$

Eine Differentialgleichung $y'(t) = f(t, y(t))$ mit Anfangsbedingung $y(t_0) = y_0$ kann in ein äquivalentes autonomes System umgewandelt werden: $Y(t) = \begin{pmatrix} t \\ y(t) \end{pmatrix}$, $Y'(t) = \begin{pmatrix} 1 \\ f(Y_1, Y_2) \end{pmatrix}$, $Y(t_0) = \begin{pmatrix} t_0 \\ y_0 \end{pmatrix}$

Def: (Anfangswertproblem) Eine Differentialgleichung (o.B.d.A. erster Ordnung) mit Anfangsbedingungen,

$$y'(t) = f(t, y), \quad y(t_0) = y_0 \qquad \text{(AWP)}$$

heißt <u>Anfangswertproblem</u>.

Bsp: (Mehrkörperproblem) Seien $y_S(t), y_E(t), y_M(t) \in \mathbb{R}^3$ die Position von Sonne, Erde, Mond mit Massen $m_S, m_E, m_M > 0$ zur Zeit t. Die Gravitationskraft zwischen zwei Körpern der Masse m_a, m_e ist $G \frac{m_a m_e}{\text{Abstand}^2}$ für die Gravitationskonstante $G = 6{,}674 \cdot 10^{-11} \frac{m^3}{kg\, s^2}$, und die Beschleunigung $y''(t)$ eines Körpers ist gleich der Kraft F pro Masse m, $y''(t) = \frac{F}{m}$. Folglich gilt (unter Vernachlässigung anderer Himmelskörper)

$$\left.\begin{aligned} y_S'' &= \frac{G}{m_S}\left[m_S m_E \frac{y_E - y_S}{|y_E - y_S|^3} + m_S m_M \frac{y_M - y_S}{|y_M - y_S|^3}\right] \\ y_E'' &= \frac{G}{m_E}\left[m_E m_S \frac{y_S - y_E}{|y_S - y_E|^3} + m_E m_M \frac{y_M - y_E}{|y_M - y_E|^3}\right] \\ y_M'' &= \frac{G}{m_M}\left[m_M m_S \frac{y_S - y_M}{|y_S - y_M|^3} + m_M m_E \frac{y_E - y_M}{|y_E - y_M|^3}\right] \end{aligned}\right\} \Leftrightarrow \tilde{y}'' := \begin{pmatrix} y_S \\ y_E \\ y_M \end{pmatrix}'' = g(y_S, y_E, y_M) = g(\tilde{y}) \in \mathbb{R}^9$$

$$\Rightarrow Y' = \begin{pmatrix} \tilde{y} \\ \tilde{y}' \end{pmatrix}' = \begin{pmatrix} \tilde{y}' \\ g(\tilde{y}) \end{pmatrix} = \begin{pmatrix} Y_2 \\ g(Y_1) \end{pmatrix} \in \mathbb{R}^{18}$$

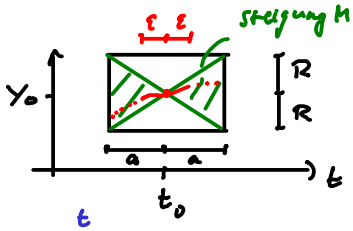
```
ms = 1.989e30; % Masse Sonne [kg]
me = 5.972e24; % Masse Erde [kg]
mm = 7.349e22; % Masse Mond [kg]
G = 6.674e-20; % Gravitationskonstante [kg km^3/s^2]

g = @(ys,ye,ym) G*[me*(ye-ys)/norm(ye-ys)^3+mm*(ym-ys)/norm(ym-ys)^3;
                   ms*(ys-ye)/norm(ys-ye)^3+mm*(ym-ye)/norm(ym-ye)^3;
                   ms*(ys-ym)/norm(ys-ym)^3+me*(ye-ym)/norm(ye-ym)^3];

% Anfangsposition [km] & -Geschwindigkeit [km/s]
y0 = [0 0 0,1.496e8 0 0,1.496e8+370300 0 0,0 0 0,0 29.78 0,0 29.78+1.06 0]';

f = @(y,t) [y(10:18);g(y(1:3),y(4:6),y(7:9))];
%  [t,y] = ode15s(@(t,y) f(y,t),[0 60^2*24*365],y0);
y = lsode(f,y0,linspace(0,60^2*24*365,365));
ye = y(:,4:5);
ym = ye+20*(y(:,7:8)-ye); % übertreibe Mondposition
plot(ye(:,1),ye(:,2),'b',ym(:,1),ym(:,2),'r','LineWidth',3); axis equ
```

Heute: Analysis gewöhnlicher Dgl.

Wann ist ein Anfangswertproblem wohlgestellt, d.h. es besitzt eine eindeutige Lösung $y(t)$?

Def: (Lipschitz) Seien X, Y normierte Räume. $f: X \to Y$ heißt Lipschitz-stetig mit Lipschitz-Konstante $L > 0$, wenn $\|f(x) - f(y)\| \le \|x - y\| \quad \forall x, y \in X$ gilt.

Thm: (Picard-Lindelöf) Sei $I = [t_0 - a, t_0 + a]$, $B = \{x \in \mathbb{R}^n \mid \|x - y_0\| \le R\}$, $f: I \times B \to \mathbb{R}^n$ stetig und Lipschitz-stetig im zweiten Argument, d.h. $\exists L > 0: \|f(t,x) - f(t,y)\| \le L\|x - y\| \ \forall t \in I, x, y \in B$.
Sei $M = \max_{(t,x) \in I \times B} \|f(t,x)\|$ und $a \le \frac{R}{M}$, dann existiert eine eindeutige Lösung $y: I \to B$ zu (AWP).



Bew: Zunächst zeige Existenz & Eindeutigkeit auf $\tilde{I} = [t_0 - \varepsilon, t_0 + \varepsilon]$ für $\varepsilon = \min\{a, \frac{1}{2L}\}$.

1) definiere die Picard-Iteration $T: C(\tilde{I}; B) \to C(\tilde{I}; B)$, $x(t) \mapsto y_0 + \int_{t_0}^{t} f(\tau, x(\tau))\, d\tau$;
dies ist wohldefiniert, da $T(x)(\cdot)$ stetig ist mit $\|T(x)(t) - y_0\| \le \int_{t_0}^{t} \|f(\tau, x(\tau))\|\, d\tau \le M\varepsilon \le R$.

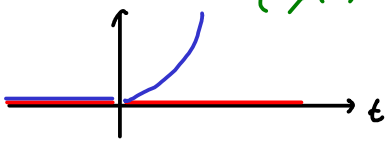
2) T ist eine Kontraktion: Mit der Supremumsnorm $\|g\|_\infty = \sup_{t \in \tilde{I}} \|g(t)\|$ gilt

$$\|T(x)(t) - T(y)(t)\| = \left\| \int_{t_0}^{t} f(\tau, x(\tau)) - f(\tau, y(\tau))\, d\tau \right\| \le \int_{t_0}^{t} \|f(\tau, x(\tau)) - f(\tau, y(\tau))\|\, d\tau$$
$$\le |t - t_0| L \|x - y\|_\infty \le \tfrac{1}{2} \|x - y\|_\infty \quad \forall t \in \tilde{I}$$
$$\Rightarrow \|T(x) - T(y)\|_\infty \le \tfrac{1}{2} \|x - y\|_\infty$$

3) $C(\tilde{I}; B)$ mit $\|\cdot\|_\infty$ ist vollständig, somit hat T nach dem Banachschen Fixpunktsatz einen eindeutigen Fixpunkt $y^* \in C(\tilde{I}; B)$ mit $y^* = T(y^*)$, d.h. eine eindeutige Lösung zu (AWP).

Wiederhole das Argument nun für $y'(t) = f(t, y(t))$, $y(t_0 + \varepsilon) = y^*(t_0 + \varepsilon)$, um eine eindeutige Lösung auf $[t_0, t_0 + 2\varepsilon]$ zu erhalten, dann analog für Anfangswerte bei $t_0 + 2\varepsilon$, $t_0 + 3\varepsilon$ usw., bis die eindeutige Lösung auf ganz I definiert ist. □

Bem: · Die Lipschitzbedingung ist nötig für Eindeutigkeit: z.B. ist $f(t,x)=\sqrt{x}$ nicht Lipschitzstetig um $x=0$, und $\begin{cases} y'(t)=f(t,y(t)) \\ y(0)=0 \end{cases}$ hat Lösungen $y(t)=0$ & $y(t)=\frac{(t)_+^2}{4}$.



· $a \le \frac{R}{M}$ ist nötig für Existenz einer Lösung auf dem ganzen Intervall I: $\begin{cases} y'(t)=y^2 \\ y(0)=1 \end{cases}$ hat Lösung $y(t)=\frac{1}{1-t}$ und ist somit nur für $t<1$ wohldefiniert.

· Die Picard-Iteration liefert auch eine Vorschrift zur Annäherung der Lösung: $y=\lim_{n\to\infty} T^n(y_0)$

Bsp: Bakterienanzahl in einer Nährlösung gehorcht $y'(t)=\alpha y(t)$ mit $\alpha>0$, $y(0)=y_0>0$.

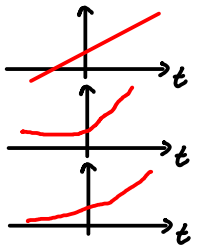
$f(t,x):=\alpha x$ ist stetig & Lipschitzstetig in x mit Lipschitzkonstante α.

$M=\max_{(t,x)\in I\times B} |f(t,x)| = \alpha(y_0+R)$, d.h. Lösung $y: I\to B$ existiert für $a \le \frac{R}{\alpha(y_0+R)}$.

$T(y_0)(t) = y_0 + \int_0^t \alpha y_0 \, d\tau = y_0(1+\alpha t)$

$T^2(y_0)(t) = y_0 + \int_0^t \alpha y_0(1+\alpha t)dt = y_0\left(1+\alpha t+\frac{(\alpha t)^2}{2}\right)$

$T^n(y_0)(t) = y_0 \sum_{j=0}^{n} \frac{(\alpha t)^j}{j!} \xrightarrow{n\to\infty} y_0 \exp(\alpha t)$



Für Stabilitätsbetrachtungen bei der Untersuchung numerischer Lösungsverfahren müssen wir den unvermeidbaren Fehler durch Rundung der Eingabedaten y_0 und f abschätzen. Ein Werkzeug hierfür ist Gronwalls Lemma.

Thm: (Gronwall) Sei $I=[a,b]$, $\alpha,\beta,u: I\to\mathbb{R}$ stetig.

1) $u'(t) \le \alpha(t)+\beta(t)u(t) \quad \forall t\in I \Rightarrow u(t) \le u(a)e^{\int_a^t \beta(s)ds} + \int_a^t \alpha(\tau) e^{\int_\tau^t \beta(s)ds} d\tau \quad \forall t\in I$

2) $u(t) \le \alpha(t)+\int_a^t \beta(s)u(s)ds \quad \forall t\in I \Rightarrow u(t) \le \alpha(t)+\int_a^t \alpha(\tau)\beta(\tau) e^{\int_\tau^t \beta(s)ds} d\tau \quad \forall t\in I.$

Bew: 0) Sei $u'(t)\le v(t) \ \forall t\in I$, dann gilt $u(t)\le w(t) \ \forall t\in I$ mit $w(t)=u(a)+\int_a^t v(s)ds$, da $(u-w)'\le 0$ & $(u-w)(0)=0$, also $u-w\le 0$ auf I.

1) Sei v die Lösung von $v'(t)=-\beta(t)v(t)$, $v(a)=1$, d.h. $v(t)=e^{-\int_a^t \beta(s)ds}$

$\Rightarrow (uv)' = u'v-\beta uv \le \alpha v+\beta uv-\beta uv = \alpha v \overset{0)}{\Rightarrow} u(t)v(t) \le u(a)+\int_a^t \alpha(s)v(s)ds$

Das Resultat folgt nun mit Division durch $v(t)$ auf beiden Seiten.

2) Sei $v(t)=\int_a^t \beta(s)u(s)ds \Rightarrow v'=\beta u \le \beta\alpha+\beta v$, $v(a)=0$

$u(t)-\alpha(t) \le v(t) \overset{1)}{\le} \int_a^t \beta(\tau)\alpha(\tau) e^{\int_\tau^t \beta(s)ds} d\tau$ □

Thm: (Kondition / Stetigkeit des Anfangswertproblems) $\begin{cases} y'(t) = f(t,y) \\ y(t_0) = y_0 \end{cases}$ erfülle die Voraussetzungen des Satzes von Picard und Lindelöf, $\tilde f$ und $\tilde y_0$ seien Näherungen mit $\|y_0 - \tilde y_0\| \le \varepsilon$ und $\|f - \tilde f\|_\infty = \sup_{t,x} \|f(t,x) - \tilde f(t,x)\| \le \delta$. Für die Lösung $\tilde y : [t_0, t_0+a] \to \mathbb{R}^n$ (wenn sie existiert) von $\begin{cases} \tilde y'(t) = \tilde f(t, \tilde y(t)) \\ \tilde y(t_0) = \tilde y_0 \end{cases}$ gilt $\|\tilde y(t) - y(t)\| \le (\varepsilon + \delta(t-t_0))\, e^{L(t-t_0)} \quad \forall\, t \in [t_0, t_0+a]$.

Bew: Sei $u(t) = \|\tilde y(t) - y(t)\|$.

$$u(t) = \Big\| \tilde y_0 - y_0 + \int_{t_0}^t \tilde f(s, \tilde y(s)) - f(s, y(s))\, ds \Big\| \le \varepsilon + \int_{t_0}^t \delta + \|f(s,\tilde y(s)) - f(s, y(s))\|\, ds \le \varepsilon + \delta(t-t_0) + L \int_{t_0}^t u(s)\, ds$$

$$\overset{\text{Gronwall}}{\Longrightarrow} u(t) \le \varepsilon + \delta(t-t_0) + \int_{t_0}^t (\varepsilon + \delta(r-t_0))\, L\, e^{L(t-r)}\, dr \le (\varepsilon + \delta(t-t_0)) \Big[1 + \underbrace{\int_{t_0}^t e^{L(t-r)}\, dr}_{e^{L(t-t_0)} - 1}\Big] \qquad \square$$

Diskrete Verfahren zur Lösung gewöhnlicher Differentialgleichungen

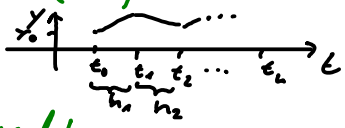
heute: · Diskretisierung
· Stabilität & Konvergenz

Sei $I = [a,b]$, $f : I \times \mathbb{R}^n \to \mathbb{R}^n$; $\begin{cases} y'(t) = f(t, y(t)) \\ y(a) = y_0 \end{cases}$ (AWP)

habe eine eindeutige Lösung auf I. Ein numerisches Verfahren liefert für Stützstellen / Knoten $t_0 = a < t_1 < \dots < t_n = b$ eine Approximation $y_i \approx y(t_i)$, $i = 0, \dots, n$.

Def: (Konvergenz)

- Die Schritt- oder Gitterweite zu Knoten $t_0, \dots, t_n$ ist $h = \max_{i=1,\dots,n} h_i$ mit $h_i = t_i - t_{i-1}$.
- Der (globale) Diskretisierungsfehler des Verfahrens ist $e_h = \max_{i=1,\dots,n} \|e_i\|$ mit $e_i = y_i - y(t_i)$.
- Das Verfahren heißt konvergent (von Ordnung p), falls $e_h \xrightarrow[h\to 0]{} 0$ ($e_h = O(h^p)$).



Def: (Verfahrenstypen) Ein Verfahren der Form

- $y_{k+1} = y_k + h_{k+1}\, \varphi(t_k, y_k, h_{k+1})$ heißt explizites Einschrittverfahren
- $y_{k+1} = y_k + h_{k+1}\, \varphi(t_k, y_k, y_{k+1}, h_{k+1})$ heißt implizites Einschrittverfahren
- $y_{k+1} = y_k + h_{k+1}\, \varphi(t_0, \dots, t_k, y_0, \dots, y_k, h_{k+1})$ heißt explizites Mehrschrittverfahren
- $y_{k+1} = y_k + h_{k+1}\, \varphi(t_0, \dots, t_k, y_0, \dots, y_k, y_{k+1}, h_{k+1})$ heißt implizites Mehrschrittverfahren

Bem: Man findet nacheinander $y_0, y_1, y_2, \dots$. In einem impliziten Verfahren muss hierzu in jedem Schritt eine Gleichung gelöst werden.

Thm: (Wohldefiniertheit) Sei $y_{k+1} = y_k + h_{k+1}\, \varphi((t_i, y_i, h_i)_i)$ ein implizites Verfahren für (AWP). Sei $B = \{y \in \mathbb{R}^n \mid \|y - y_0\| \le R\}$, φ Lipschitz-stetig in y_{k+1} mit Konstante L und $\|\varphi\| \le M$ für alle $t_0, \dots, t_k \in I$, $y_0, \dots, y_{k+1} \in B$. Schließlich sei $b - a \le \frac{R}{2M}$ (analog zu Picard-Lindelöf). Es existiert ein h, sodass bei Wahl von $h_k \le h$ für alle k das Verfahren in jedem Schritt k eine eindeutige Lösung $y_{k+1} = \psi((t_i, y_i, h_i)_{i=0,\dots,k}) \in B$ besitzt. Ist φ Lipschitz-stetig in $y_0, \dots, y_k$, so auch ψ.

Bem: Wegen des Theorems können wir ein implizites Verfahren auch interpretieren als ein explizites $y_{k+1} = y_k + h_{k+1}\,\varphi(t_0,\dots,t_k,y_0,\dots,y_k,\Psi(t_0,\dots,t_k,y_0,\dots,y_k,h_{k+1}),h_{k+1})$ mit Lipschitz-stetiger rechter Seite (wird für Stabilität benötigt).

Bew: Sei $h < \frac{1}{L}$ (entspricht ε im Satz von Picard-Lindelöf) und $h < \frac{R}{2M}$, $h_k \le h\ \forall k$. Wir zeigen mittels vollständiger Induktion $y_k \in B_k = \{y \in \mathbb{R}^n \mid \|y - y_0\| \le M(t_k - a)\}$ und die Aussagen des Satzes. Der Induktionsanfang $k=0$ ist trivial. Zum Induktionsschritt betrachte y_{k+1}. Setze $g(y) = y_k + h_{k+1}\varphi(t_0,\dots,t_k,y_0,\dots,y_k,y,h_{k+1})$. Für $y \in B$ ist $\|g(y) - y_k\| \le h\|\varphi(\dots)\| \le hM < \frac{R}{2}$ und somit $g: B \to B$. Außerdem ist g eine Kontraktion: $\|g(x) - g(z)\| = h_{k+1}\|\varphi(\dots,x,h_{k+1}) - \varphi(\dots,z,h_{k+1})\| \le \underbrace{hL}_{<1}\|x - z\|$. Nach dem Banachschen Fixpunktsatz existiert ein eindeutiger Fixpunkt von g in B, also eine eindeutige Lösung $y_{k+1} \in B$. Außerdem $\|y_{k+1} - y_k\| = \|g(y_{k+1}) - y_k\| \le hM$ und somit $y_{k+1} \in B_{k+1}$. Schließlich ist $\Psi(t_0,\dots,t_k,y_0,\dots,y_k,h_{k+1}) := y_{k+1}$ Lipschitz-stetig in $(y_0,\dots,y_k)$: Sei $x = \Psi(t_0,\dots,t_k,x_0,\dots,x_k,h_{k+1})$, $z = \Psi(t_0,\dots,t_k,z_0,\dots,z_k,h_{k+1})$, dann ist

$$\begin{aligned}\|x - z\| &= \|x_k + h_{k+1}\varphi(\dots,x_0,\dots,x_k,x,h_{k+1}) - z_k - h_{k+1}\varphi(\dots,z_0,\dots,z_k,z,h_{k+1})\| \\ &\le \|x_k - z_k\| + h\|\varphi(\dots,x_0,\dots,x_k,x,h_{k+1}) - \varphi(\dots,z_0,\dots,z_k,x,h_{k+1})\| \\ &\quad + h\|\varphi(\dots,z_0,\dots,z_k,x,h_{k+1}) - \varphi(\dots,z_0,\dots,z_k,z,h_{k+1})\| \\ &\le \|x_k - z_k\| + h\tilde{L}\|(x_0,\dots,x_k) - (z_0,\dots,z_k)\| + hL\|x - z\|\end{aligned}$$

(Lipschitz-Konstante $\tilde{L}$ in $(y_0,\dots,y_k)$)

$$\Rightarrow \|x - y\| \le \frac{1 + h\tilde{L}}{1 - hL}\|(x_0,\dots,x_k) - (z_0,\dots,z_k)\| \Rightarrow \Psi \text{ hat Lipschitz-Konstante } \frac{1 + h\tilde{L}}{1 - hL}$$ □

Bem: Die Fixpunkt-Iteration $y_{k+1}^{i+1} = g(y_{k+1}^i)$ aus dem Beweis kann mit $y_{k+1}^0 = y_k$ genutzt werden, um y_{k+1} im impliziten Verfahren zu berechnen.

Wie kann man entsprechende Verfahren herleiten? Mittels Quadratur!

$$y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_{k+1}} f(t, y(t))\,dt$$

$$\approx y(t_k) + h_{k+1}\begin{cases} f(t_k, y(t_k)) & \text{Newton-Cotes mit Stützstelle } t_k \\ f(t_{k+1}, y(t_{k+1})) & \text{Newton-Cotes mit Stützstelle } t_{k+1} \\ (f(t_k, y(t_k)) + f(t_{k+1}, y(t_{k+1})))/2 & \text{Trapezregel} \end{cases}$$

Def: (Euler-Verfahren) Zur Differentialgleichung $y'(t) = f(t, y(t))$ heißt

$y_{k+1} = y_k + h_{k+1} f(t_k, y_k)$ — explizites Eulerverfahren,

$y_{k+1} = y_k + h_{k+1} f(t_{k+1}, y_{k+1})$ — implizites Eulerverfahren,

$y_{k+1} = y_k + h_{k+1}\frac{f(t_k, y_k) + f(t_{k+1}, y_{k+1})}{2}$ — Trapezverfahren,

$y_{k+1} = y_k + h_{k+1}\left((1-\theta) f(t_k, y_k) + \theta f(t_{k+1}, y_{k+1})\right)$ — θ-Verfahren,

$y_{k+1} = y_k + h_{k+1}\frac{f(t_k, y_k) + f(t_{k+1}, y_k + h_{k+1} f(t_k, y_k))}{2}$ — verbessertes Eulerverfahren.

```
% löse y' = t - y^2, y(0) = 0
f = @(t,y) t-y^2;
h = .1;      % Gitterweite
t = 0:h:1;  % Gitter
n = length(t)-1;
y = zeros(3,n+1);
for k = 1:n
  % explizites Eulerverfahren
  y(1,k+1) = y(1,k) + h * f( t(k), y(1,k) );
  % implizites Eulerverfahren
  y(2,k+1) = ( -.5 + sqrt(.25+h*(h*t(k+1)+y(2,k))) ) / h;
  % verbessertes Eulerverfahren
  y(3,k+1) = y(3,k) + h * ( f( t(k), y(3,k) ) ...
                          + f( t(k+1), y(3,k)+h*f(t(k),y(3,k)) ) ) / 2;
end
plot(t',y','.-','Linewidth',3,'Markersize',20);
hold on;
t = linspace(0,1,100);
y = lsode(@(y,t)f(t,y),0,t);
plot(t,y,'k','Linewidth',3);
```

Das verbesserte Eulerverfahren ist viel genauer – woran liegt dies?

Konvergenz- & Stabilitätsanalyse von Einschrittverfahren

Thm: (diskretes Gronwall-Lemma) Seien $\alpha_k, \beta_k, u_k \geq 0$ reelle Folgen mit $u_{k+1} - u_k \leq \alpha_k + \beta_k u_k \ \forall k$.

Dann gilt $u_k \leq u_0 \exp\left(\sum_{j=0}^{k-1} \beta_j\right) + \sum_{i=0}^{k-1} \alpha_i \exp\left(\sum_{j=i+1}^{k-1} \beta_j\right)$.

Bem: Im Vergleich zum Gronwall-Lemma wurden Ableitungen & Integrale durch Differenzen & Summen ersetzt.

Bew: Induktionsanfang: $k=1$ trivial

Induktionsschritt:

$$\begin{aligned} u_{k+1} &\leq \alpha_k + (1+\beta_k) u_k \leq \alpha_k \exp(\beta_k) + \exp(\beta_k) u_k \\ &\leq \alpha_k \exp(\beta_k) + \exp(\beta_k)\left[u_0 \exp\left(\sum_{j=0}^{k-1} \beta_j\right) + \sum_{i=0}^{k-1} \alpha_i \exp\left(\sum_{j=i+1}^{k-1} \beta_j\right)\right] \\ &= u_0 \exp\left(\sum_{j=0}^{k} \beta_j\right) + \sum_{i=0}^{k} \alpha_i \exp\left(\sum_{j=i+1}^{k} \beta_j\right) \end{aligned}$$ □

Der Einfachheit halber sei im Folgenden $h_k = h$ für alle k (die Verallgemeinerung zu h_k ist einfach).

Thm: (Stabilität) Seien $\tilde{y}_0$, $\tilde{\varphi}$ Näherungen für y_0, φ mit $\|\tilde{y}_0 - y_0\| \leq \varepsilon$, $\|\tilde{\varphi} - \varphi\|_\infty \leq \delta$ für ein (o.B.d.A. explizites) Einschrittverfahren $y_{k+1} = y_k + h\varphi(t_k, y_k, h)$. φ sei Lipschitz-stetig mit Konstante L. Die Lösung $\tilde{y}_k$ des gestörten Verfahrens erfüllt

$$\|\tilde{y}_k - y_k\| \leq (\varepsilon + \delta t_k) \exp(L t_k).$$

Bew: Sei $u_k = \tilde{y}_k - y_k$.

$$\begin{aligned} \|u_{k+1}\| &= \|\tilde{y}_k + h\tilde{\varphi}(t_k, \tilde{y}_k, h) - y_k - h\varphi(t_k, y_k, h)\| \\ &\leq \|u_k\| + h\left[\|\tilde{\varphi}(t_k, \tilde{y}_k, h) - \varphi(t_k, \tilde{y}_k, h)\| + \|\varphi(t_k, \tilde{y}_k, h) - \varphi(t_k, y_k, h)\|\right] \\ &\leq \|u_k\| + h\delta + hL\|u_k\| = h\delta + (1+hL)\|u_k\|, \end{aligned}$$

mit dem diskreten Gronwall-Lemma folgt also

$$\|u_k\| \le \|u_0\| \exp(khL) + \sum_{i=0}^{k-1} h\delta \exp((k-i-1)hL) \le c\exp(Lt_k) + t_k\delta\exp(Lt_n) \quad \square$$

Def: (Abschneidefehler) Für ein Verfahren $y_{k+1} = y_k + h_{k+1}\varphi(t_i, y_i, h_i)_i$ und eine Lösung $y(t)$ von $y'(t) = f(t, y(t))$ heißt $\tau_k = \frac{1}{h_{k+1}}\left[y(t_{k+1}) - (y(t_k) + h_{k+1}\varphi((t_i, y(t_i), h_i)_i))\right]$ lokaler Diskretisierungsfehler oder Abschneidefehler.

Das Verfahren heißt konsistent (von Ordnung p), falls $\max_k \|\tau_k\| \xrightarrow{h\to 0} 0$ $\left(\max_k \|\tau_k\| = O(h^p)\right)$.

Bsp.: · Abschneidefehler explizites Eulerverfahren: Taylor: $y(t_{k+1}) = y(t_k) + h\,y'(t_k) + \frac{h^2}{2}y''(\xi_k)$, $\xi_k \in [t_k, t_{k+1}]$

$$\tau_k = \frac{y(t_{k+1}) - y(t_k)}{h} - f(t_k, y(t_k)) = y'(t_k) + \frac{h}{2}y''(\xi_k) - f(t_k, y(t_k)) = \frac{h}{2}y''(\xi_k)$$

$\Rightarrow$ Konsistent von Ordnung 1

· verbessertes Eulerverfahren: ($y'(t_k)$)

$$\tau_k = \frac{y(t_{k+1}) - y(t_k)}{h} - \frac{f(t_k, y(t_k)) + f(t_{k+1}, y(t_k) + h f(t_k, y(t_k)))}{2}$$

$$f(t_{k+1}, y(t_k) + h\,y'(t_k)) = f(t_k, y(t_k)) + \begin{pmatrix} h \\ h\,y'(t_k)\end{pmatrix} \cdot \begin{pmatrix} \partial f/\partial t \\ \partial f/\partial y\end{pmatrix}\Big|_{(t_k, y(t_k))} + \begin{pmatrix} h \\ h\,y'(t_k)\end{pmatrix} \begin{pmatrix} \frac{\partial^2 f}{\partial t^2} & \frac{\partial^2 f}{\partial t \partial y} \\ \frac{\partial^2 f}{\partial t \partial y} & \frac{\partial^2 f}{\partial y^2}\end{pmatrix}\Big|_{(\xi_k, \eta_k)} \begin{pmatrix} h \\ h\,y'(t_k)\end{pmatrix}$$

$$= y'(t_k) + h\frac{d}{dt}f(t, y(t))\Big|_{t_k} + O(h^2) = y'(t_k) + h\,y''(t_k) + O(h^2)$$

$$\tau_{k+1} = y'(t_k) + \frac{h}{2}y''(t_k) + \frac{h^2}{6}y'''(\xi_k) - y'(t_k) - \frac{h\,y''(t_k) + O(h^2)}{2} = O(h^2)$$

$\Rightarrow$ Konsistent von Ordnung 2

Thm: (Konvergenz) Das Einschrittverfahren $y_{k+1} = y_k + h\varphi(t_k, y_k, h)$ sei konsistent (von Ordnung p) und stabil. Dann ist das Verfahren konvergent (von Ordnung p).

Bem: · „Konsistenz + Stabilität = Konvergenz", d.h. werden Rundungsfehler nicht mehr als unvermeidbar verstärkt und ist das Verfahren konsistent, dann folgt hieraus, dass der globale Diskretisierungsfehler beliebig klein gemacht werden kann.

· Explizites / implizites Eulerverfahren konvergieren nur mit Ordnung 1, das verbesserte Eulerverfahren hingegen mit Ordnung 2.

Bew: Sei $\tilde\varphi(t, y_k, h)$ definiert als $\frac{\hat y(t+h) - \hat y(t)}{h}$ für die Lösung $\hat y$ von $\hat y'(t) = f(t, \hat y(t))$ mit $\hat y(t) = y_k$.

Es ist $\varphi(t_k, y_k, h) - \tilde\varphi(t, y_k, h) = \varphi(t_k, y_k, h) - \frac{\hat y(t_{k+1}) - \hat y(t_k)}{h} = \tau_k$.

Das Verfahren mit $\tilde\varphi$ liefert die exakte Lösung $y(t_k)$, das mit φ kann aufgefasst werden als Approximation. Der Stabilitätssatz liefert nun

$$\|e_k\| = \|y_k - y(t_k)\| \le \max_j \|\tau_j\|\, t_k \exp(Lt_k) \le C \max_j \|\tau_j\| \text{ für eine Konstante } C > 0. \quad \square$$

heute: Runge-Kutta-Verfahren

Einschrittverfahren

Explizite Einschrittverfahren haben die Form $y_{k+1} = y_k + h\,\varphi(t_k, y_k, h)$. Wie findet man gute φ?

Def (Taylorverfahren) Sei $f \in C^p(\mathbb{R}\times\mathbb{R}^n)$ und $\hat{y}$ die Lösung zu $\hat{y}'(t) = f(t, \hat{y}(t))$, $\hat{y}(t_k) = y_k$. Das Verfahren mit $h\,\varphi(t_k, y_k, h) = \sum_{j=1}^{p} \frac{h^j}{j!}\,\hat{y}^{(j)}(t_k)$, der Taylorentwicklung von $\hat{y}(t_{k+1}) - y_k$ um t_k von Ordnung p, heißt Taylorverfahren.

Bem: Die Ableitungen $\hat{y}^{(j)}(t_k)$ können nur mit Hilfe der Ableitungen von f und des Werts y_k bestimmt werden:

$$\hat{y}(t_k) = y_k,\quad \hat{y}'(t_k) = f(t_k, y_k),\quad \hat{y}''(t_k) = \frac{d}{dt} f(t, \hat{y}(t))\Big|_{t_k} = \left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}\hat{y}'\right)\Big|_{\substack{t=t_k\\ y=y_k}} = \left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial y} f\right)\Big|_{\substack{t=t_k\\ y=y_k}} \text{usw.}$$

Thm (Taylorverfahren) Das Taylorverfahren ist stabil und konsistent von Ordnung p (somit konvergent von Ordnung p).

Bew: Stabilität: φ ist Lipschitz-stetig in y_k (da $\hat{y}^{(j)}(t_k)$ eine Summe von Produkten von bis zu $(j-1)$-ten Ableitungen von f in y_k ist und $f \in C^p$)

Konsistenz: $\tau_k = \frac{\hat{y}(t_{k+1}) - \hat{y}(t_k)}{h} - \varphi(t_k, y_k, h) = \sum_{j=1}^{p} \frac{h^{j-1}}{j!}\,\hat{y}^{(j)}(t_k) + O(h^p) - \sum_{j=1}^{p} \frac{h^{j-1}}{j!}\,\hat{y}^{(j)}(t_k) = O(h^p)$ □

Im Taylorverfahren müssen hohe Ableitungen von f ausgewertet werden – oft sind die Formeln zu kompliziert oder gar nicht bekannt. Stattdessen kann man versuchen, höhere Ordnung nur mit Auswertungen von f zu erhalten. Hierzu sei $I_p(g) = \sum_{i=0}^{p} c_i\, g(a_i)$ eine Quadraturformel auf $[0,1]$

$$\Rightarrow\ y(t_{k+1}) - y(t_k) = \int_{t_k}^{t_{k+1}} f(t, y(t))\,dt \approx h \sum_{i=0}^{p} c_i\, f(t_k + a_i h,\ y(t_k + a_i h)).$$

Hier muss noch $y(t_k + a_i h)$ mittels $y(t_k)$ und f möglichst effizient approximiert werden.

Def (Runge-Kutta-Verfahren) Seien $c_i, a_i, b_{ij} \in \mathbb{R}$ für $i, j = 1, 2, \ldots$

- Ein Verfahren mit

$$\varphi(t, y, h) = \sum_{r=1}^{R} c_r k_r, \qquad k_r = f\left(t + a_r h,\ y + h \sum_{s=1}^{r-1} b_{rs} k_s\right),\ r = 1, \ldots, R$$

heißt R-stufiges explizites Runge-Kutta-Verfahren.

- Ein Verfahren mit

$$\varphi(t, y, h) = \sum_{r=1}^{R} c_r k_r, \qquad k_r = f\left(t + a_r h,\ y + h \sum_{s=1}^{R} b_{rs} k_s\right),\ r = 1, \ldots, R$$

heißt R-stufiges implizites Runge-Kutta-Verfahren.

Runge-Kutta-Verfahren sind der Standard unter den Einschrittverfahren.
Geeignete Parameter c_i, a_i, b_{ij} erhält man durch Ermitteln des Abschneidefehlers.

Bsp: Ein 2-stufiges Runge-Kutta-Verfahren hat die Form $y_{k+1} = y_k + h\,(c_1 k_1 + c_2 k_2)$ mit

$k_1 = f(t_k + a_1 h, y_k)$, $k_2 = f(t_k + a_2 h, y_k + b_{21} h k_1)$.

Abschneidefehler $\tau_k = \frac{y(t_{k+1}) - y(t_k)}{h} - c_1 f(t_k + a_1 h, y(t_k)) - c_2 f(t_k + a_2 h, y(t_k) + b_{21} h f(t_k, y(t_k)))$

$= y'(t_k) + \frac{h}{2} y''(t_k) + O(h^2) - c_1 y'(t_k) - c_1 h a_1 \partial_t f - c_2 \left[f(t_k, y(t_k)) + h\{a_2 \partial_t f + b_{21} \partial_y f\, f\} + O(h^2) \right]$

$y'' = \frac{d}{dt} f(t,y) = \partial_t f + f \partial_y f$

$= y'(t_k)(1 - c_1 - c_2) + \partial_t f(t_k, y(t_k))\, h\left(\frac{1}{2} - c_1 a_1 - c_2 a_2\right) + f(t_k, y(t_k))\, \partial_y f(t_k, y(t_k))\, h\left(\frac{1}{2} - c_2 b_{21}\right) + O(h^2)$

$\Rightarrow$ wähle $c_1 + c_2 = 1$, $c_2 b_{21} = \frac{1}{2}$, $c_1 a_1 + c_2 a_2 = \frac{1}{2}$ $\Rightarrow$ Verfahren ist konsistent von Ordnung 2.

- $a_2 = \frac{1}{2}$, $a_1 = 0$ $\Rightarrow$ $y_{k+1} = y_k + h\, f\left(t_k + \frac{h}{2}, y_k + \frac{h}{2} f(t_k, y_k)\right)$ „modifiziertes Eulerverfahren"
- $a_2 = 1$, $a_1 = 0$ $\Rightarrow$ $y_{k+1} = y_k + \frac{h}{2}\left[f(t_k, y_k) + f(t_k + h, y_k + h f(t_k, y_k))\right]$ „verbessertes Eulerverfahren"

Thm: (Runge-Kutta-Verfahren) Das Runge-Kutta-Verfahren ist stabil.

- Ist $f \in C^n$ und $\sum_{r=1}^{R} c_r = 1$, so ist das Verfahren konsistent von Ordnung ≥ 1.
- Ist zusätzlich $f \in C^2$, $\sum_{r=1}^{R} c_r \sum_{s=1}^{R} b_{rs} = \frac{1}{2}$, $\sum_{r=1}^{R} a_r c_r = \frac{1}{2}$, so ist das Verfahren konsistent von Ordnung ≥ 2.

  (Für explizite Verfahren setzen wir $b_{rs} = 0 \;\forall\, s \geq r$.)
- Es gibt kein zweistufiges explizites Verfahren der Konsistenzordnung ≥ 3.

Bew: $\varphi(t, y, h)$ ist Summe und Komposition von f, welches Lipschitzstetig in y ist $\Rightarrow$ auch φ ist dies

- $\tau_k = \frac{y(t_{k+1}) - y(t_k)}{h} - \sum_{r=1}^{R} c_r f\left(t_k + a_r h, y_k + h \sum_{s=1}^{R} b_{rs} k_s\right)$

$= y'(t_k) + \frac{h}{2} y''(t_k) + O(h^2) - \sum_{r=1}^{R} c_r \left[f(t_k, y_k) + h\left\{a_r \partial_t f + \partial_y f \sum_{s=1}^{R} b_{rs} k_s\right\}_{t_k, y_k} + O(h^2)\right]$

$\sum c_r = 1$ $\Rightarrow$ $= h\left[\frac{y''(t_k)}{2} - \sum_{r=1}^{R} c_r \left\{a_r \partial_t f + \partial_y f \sum_{s=1}^{R} b_{rs} k_s\right\}_{t_k, y_k}\right] + O(h^2)$

- $\tau_k = h\left[\frac{y''(t_k)}{2} - \sum_{r=1}^{R} c_r \left\{a_r \partial_t f + \partial_y f \sum_{s=1}^{R} b_{rs} \left(f + O(h)\right)\right\}_{t_k, y_k}\right] + O(h^2)$

$\sum c_r a_r = \frac{1}{2}$, $\sum c_r \sum b_{rs} = \frac{1}{2}$ $\Rightarrow$ $= h\left[\frac{y''(t_k)}{2} - \frac{1}{2}\left\{\partial_t f + f \partial_y f\right\}_{t_k, y_k} + O(h)\right] + O(h^2) = O(h^2)$

- Der Einfachheit halber sei (AWP) autonom, d.h. f hängt nur von y ab.

$\tau_k = \frac{y(t_{k+1}) - y(t_k)}{h} - c_1 f(y(t_k)) - c_2 f\left(y(t_k) + b_{21} h f(y(t_k))\right)$

$= y'(t_k) + \frac{h}{2} y''(t_k) + \frac{h^2}{6} y'''(t_k) + O(h^3) - c_1 y'(t_k) - c_2 \left[f + b_{21} h f f' + \frac{b_{21}^2 h^2}{2} f^2 f'' + O(h^3)\right]_{y = y(t_k)}$

$= h^2 \left[\frac{y'''}{6} - \frac{c_2}{2} b_{21}^2 f^2(y) f''(y)\right]_{t = t_k} + O(h^3)$,

jedoch ist $y'''(t) = \frac{d^2}{dt^2} f(y(t)) = \frac{d}{dt}\left[f'(y(t)) f(y(t))\right] = f'(y(t))^2 f(y(t)) + f(y(t))^2 f''(y(t))$ □

Bem:

- Die Konsistenzordnung eines expliziten R-stufigen Runge-Kutta-Verfahrens ist $\leq R$, z.B. ist die maximal mögliche Konsistenzordnung eines 5-stufigen Verfahrens 4.
- Sind die Stützstellen a_i und Gewichte c_i der Gauß-Quadratur gewählt, so gibt es b_{ij}, so dass das implizite R-stufige Verfahren Konsistenzordnung $2R$ hat.
- Runge-Kutta-Verfahren werden typischerweise per Butcher-Diagramm dargestellt:

a_1	b_{11}	...	b_{1R}
$\vdots$	$\vdots$		$\vdots$
a_R	b_{R1}	...	b_{RR}
	c_1	...	c_R

Bsp:

- explizites Euler-Verfahren
$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}$$
- implizites Euler-Verfahren
$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$
- modifiziertes Euler-Verfahren
$$\begin{array}{c|cc} 0 & & \\ 1/2 & 1/2 & \\ \hline & 0 & 1 \end{array}$$
- verbessertes Euler-Verfahren
$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 1/2 & 1/2 \end{array}$$
- 4. Ordnung Runge-Kutta-Verfahren
$$\begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & & 1/2 & & \\ 1 & & & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

Bem: Matlab & Octave implementieren mit ode45 das Dormand-Prince-Verfahren (siehe Wikipedia: Dormand-Prince-method). Dies sind zwei 7-stufige Runge-Kutta-Verfahren mit Konsistenzordnung 4 bzw. 5, die sich nur in den c_i unterscheiden und somit die gleichen Funktionsauswertungen benutzen. Die Differenz beider Methoden liefert eine Fehlerschätzung für das 4.-Ordnung-Verfahren und kann somit genutzt werden, um die Schrittweite h adaptiv (d.h. so, dass eine vorgegebene Fehlerschranke nicht überschritten wird) anzupassen.

Lineare Mehrschrittverfahren

Einschrittverfahren haben den Nachteil, dass sie alle vorherigen Funktionsauswertungen wieder vergessen und eine Approximation an y_{k+1} nur aus y_k erhalten. Für hohe Konsistenzordnungen muss f in jedem Schritt häufig ausgewertet werden. Mehrschritt-Verfahren hingegen benutzen die alten Auswertungen weiter. Die Motivation stammt aus Polynominterpolation und Quadratur.

Bsp:

- Herleitung eines Mehrschritt-Verfahrens aus Polynominterpolation: „Rückwärts-Differentiationsformel"

  Sei $\sum_{i=0}^{m} y(t_{k+i}) \ell_i(t)$ das Interpolationspolynom zu y an den Stützstellen $t_k, \dots, t_{k+m}$.

  Die Gleichung $f(t_{k+\ell}, y_{k+\ell}) = y'(t_{k+\ell}) \approx \sum_{i=0}^{m} y(t_{k+i}) \ell_i'(t_{k+\ell})$ für $\ell \in \{0, \dots, m\}$ liefert das Verfahren $\sum_{i=0}^{m} \alpha_i y_{k+i} = f(t_{k+\ell}, y_{k+\ell})$ mit $\alpha_i = \ell_i'(t_{k+\ell})$, zu lösen für y_{k+m}.

- Herleitung eines Mehrschritt-Verfahrens aus Quadratur:

  Sei $I_m(g) = \sum_{i=0}^{m} \beta_i g(t_{k+i})$ eine Quadraturformel zu Stützstellen $t_k, \dots, t_{k+m}$.

  Die Gleichung $y(t_{k+m}) - y(t_k) = \int_{t_k}^{t_{k+m}} f(t, y(t))\, dt \approx \sum_{i=0}^{m} \beta_i f(t_{k+i}, y(t_{k+i}))$ liefert das Verfahren $y_{k+m} - y_k = \sum_{i=0}^{m} \beta_i f(t_{k+i}, y_{k+i})$, zu lösen für y_{k+m}.

Bsp:

- Simpson-Regel: $y_{k+2} - y_k \approx \int_{t_k}^{t_{k+2}} y(t)\, dt \approx \frac{h}{3}[f_k + 4 f_{k+1} + f_{k+2}]$ $\quad (f_k := f(t_k, y_k))$
- Adams-Bashforth-Verfahren: $y_{k+4} = y_{k+3} + \frac{h}{24}[55 f_{k+3} - 59 f_{k+2} + 37 f_{k+1} - 9 f_k]$
- Adams-Moulton-Verfahren: $y_{k+4} = y_{k+3} + \frac{h}{24}[9 f_{k+4} + 19 f_{k+3} - 5 f_{k+2} - 9 f_{k+1}]$

Def: (Lineares Mehrschritt-Verfahren) Sei die Schrittweite konstant, $h_k = h \ \forall k$. Ein Verfahren der Form

$$\sum_{i=0}^{m} \alpha_i y_{k+i} = h \sum_{i=0}^{m} \beta_i f(t_{k+i}, y_{k+i}), \quad \alpha_m \neq 0,$$

zu lösen für y_{k+m}, heißt lineares **m-Schritt-Verfahren** (implizit für $\beta_m \neq 0$, explizit für $\beta_m = 0$).

Bem: Oft werden lineare Mehrschritt-Verfahren in Paaren als Prädiktor-Korrektor-Verfahren genutzt: Ein explizites Verfahren ist der Prädiktor und liefert einen guten, einfach zu berechnenden Startwert für eine Fixpunkt-Iteration für den Korrektor, ein implizites Verfahren.

· Das Verfahren kann geschrieben werden als $y_{k+m} = y_k + h\varphi(t_0, \dots, t_{k+1}, y_0, \dots, y_{k+m}, h)$ für $\varphi(\dots) = \sum_{i=0}^{m} \beta_i f(t_{k+i}, y_{k+i}) + \frac{1}{h}\left[y_{k+m} - y_k - \sum_{i=0}^{m} \alpha_i y_{k+i}\right]$, der **Abschneidefehler** wird daher oft eingeführt als

$$\tau_{k+m} = \frac{1}{h}\sum_{i=0}^{m} \alpha_i y(t_{k+i}) - \sum_{i=0}^{m} \beta_i \underbrace{f(t_{k+i}, y(t_{k+i}))}_{y'(t_{k+i})}.$$

Thm: (Konsistenz) Sei $C_0 = \sum_{j=0}^{m} \alpha_j$, $C_1 = \sum_{j=1}^{m} j\alpha_j - \sum_{j=0}^{m} \beta_j$, $C_q = \sum_{j=1}^{m} \frac{j^q}{q!}\alpha_j - \sum_{j=1}^{m} \frac{j^{q-1}}{(q-1)!}\beta_j \quad \forall q > 1$.

Das m-Schritt-Verfahren ist konsistent von Ordnung p, wenn $C_0 = \dots = C_p = 0$.

Bew: Sei $y \in C^{p+1}$, Taylor liefert $y(t_{k+i}) = \sum_{j=0}^{p} \frac{y^{(j)}(t_k)}{j!}(ih)^j + O(h^{p+1})$

$$\Rightarrow \tau_{k+m} = \sum_{i=0}^{m} \frac{\alpha_i y(t_{k+i}) - h\beta_i y'(t_{k+i})}{h} = \sum_{i=0}^{m} \frac{\alpha_i \sum_{j=0}^{p} \frac{y^{(j)}(t_k)}{j!}(ih)^j - h\beta_i \sum_{j=0}^{p} \frac{y^{(j+1)}(t_k)}{j!}(ih)^j}{h}$$

$$= y(t_k)\left[\sum_{i=0}^{m}\alpha_i\right]/h + y'(t_k)\left[\sum_{i=0}^{m} i\alpha_i - \sum_{i=0}^{m}\beta_i\right] + \sum_{j=2}^{p} y^{(j)}(t_k)\left[\sum_{i=0}^{m} \alpha_i \frac{i^j}{j!} - \sum_{i=0}^{m}\beta_i \frac{i^{j-1}}{(j-1)!}\right]h^{j-1} + O(h^p)$$

$$= \sum_{j=0}^{p} C_j y^{(j)}(t_k) h^{j-1} + O(h^p) \qquad \square$$

Stabilität linearer Mehrschrittverfahren

Im Unterschied zu Einschrittverfahren sind bei Mehrschritt-Verfahren die ersten m Werte $y_0, \dots, y_{m-1}$ Eingabedaten (auch wenn $y_0, \dots, y_{m-1}$ in Wirklichkeit berechnet werden, jedoch durch ein anderes Verfahren). Daher definiert man Stabilität für Mehrschritt-Verfahren wie folgt (der Begriff „Nullstabilität" kommt daher, dass es ausreichen wird, die Differentialgleichung $y' = 0$ zu betrachten).

Def: (Nullstabilität) Ein lineares m-Schritt-Verfahren $y_{k+m} = y_k + h\varphi(t_k, y_{k+m-1}, \dots, y_k, y_{k+1}, h)$ zur numerischen Lösung einer Differentialgleichung auf $[a,b]$ mit Schrittweite h heißt (asymptotisch) **stabil** oder **nullstabil** wenn $K, h_0 > 0$ existieren, so dass für jede Lösung u_k, $k = 0, \dots, N_h = \frac{b-a}{h}$, mit $h < h_0$ und jede durch Rundung und Eingabefehler gestörte Lösung v_k gilt

$$\max_{k=0,\dots,N_h} \|v_k - u_k\| \leq K \max_{k=0,\dots,N_h} \|F_k\|,$$

wobei F_k der Rundungsfehler ist: $F_k = \begin{cases} v_k - u_k & , k < m \\ \frac{v_{k+1} - v_k}{h} - \varphi(t_k, v_{k-m+1}, \dots, v_k, v_{k+1}, h) \end{cases}$

Bem: Wie immer hat dies die Interpretation, dass akkumulierte Rundungsfehler nicht größer als der unvermeidbare Fehler sein sollen; dieser ist const. · (Anfangswertfehler + Funktionswertfehler).

Bsp: Betrachte das 2-Schritt-Verfahren $y_{k+2} + 4y_{k+1} - 5y_k = h\left(4f(t_{k+1}, y_{k+1}) + 2f(t_k, y_k)\right)$;
es hat Konsistenzordnung 3 (Hausaufgabe).

```
f = @(t,y) 0;   oder 1
for h = logspace(-1,-1.5,3)
  t = 0:h:1;
  n = length(t);
  y = zeros(1,n);
  y(2) = eps;   oder t(2)
  for k = 3:n
    y(k) = 5*y(k-2) - 4*y(k-1) + h*( 2*f(t(k-2),y(k-2)) + 4*f(t(k-1),y(k-1)) );
  end;
  plot(t,y,'Linewidth',3); hold on; pause;
end;
```

Rundungsfehler in Anfangswerten oder Rechenschritten werden erheblich verstärkt (mit kleineren h immer mehr) $\Rightarrow$ das Verfahren ist weder stabil noch konvergent!

Es gibt ein einfaches Kriterium, Nullstabilität von Mehrschritt-Verfahren zu prüfen.

Def: (charakteristischen Polynom) Das <u>erste und zweite charakteristische Polynom</u> eines Mehrschritt-Verfahrens $\sum_{i=0}^{m} \alpha_i y_{k+i} = h \sum_{i=0}^{m} \beta_i f(t_{k+i}, y_{k+i})$ sind gegeben durch

$$\rho(x) = \sum_{j=0}^{m} \alpha_j x^j \quad \& \quad \sigma(x) = \sum_{j=0}^{m} \beta_j x^j .$$

Thm: (Wurzelbedingung von Dahlquist) Ein lineares Mehrschritt-Verfahren ist nullstabil für alle (AWP) mit Lipschitz-stetigem f genau dann, wenn alle Nullstellen des ersten charakteristischen Polynoms ρ im komplexen Einheitskreis liegen, wobei Nullstellen auf dem Rand einfach sein müssen.

Bsp: Obiges 2-Schritt-Verfahren hat $\rho(x) = x^2 + 4x - 5$ mit Nullstellen $\{-5, 1\}$ $\Rightarrow$ instabil

Der Beweis erfordert eine Einführung in Differenzengleichungen.

Def: (Differenzengleichung) Die <u>lineare Differenzengleichung</u> zu den Koeffizienten $\alpha_0, \dots, \alpha_m \in \mathbb{R}, \alpha_m \neq 0$ und der Folge b_k, $k = 0, 1, 2, \dots$, ist $\sum_{i=0}^{m} \alpha_i y_{k+i} = b_k$. (*)

Für $(b_k) \neq 0$ heißt sie <u>inhomogen</u>, für $(b_k) = 0$ <u>homogen</u>.

Thm: 1) Die Menge aller Lösungen (y_k) von (*) mit $(b_k) = 0$ ist ein m-dimensionaler Vektorraum

2) Alle Lösungen von (*) haben die Form $y = \sum_{k=0}^{\infty} b_k v^k + \sum_{k=0}^{m-1} y_k w^k$, wobei

- w^k die Lösung der homogenen Gleichung mit Anfangswerten $w_i^k = \delta_{ik}$, $i = 0, \dots, m-1$, ist,
- $v^k = (\underbrace{0, \dots, 0}_{(k+1)\text{-mal}}, \frac{1}{\alpha_m} w^{m-1})$.

3) y_k löst die homogene Gleichung $\Leftrightarrow$ $y_k = \sum_{j=0}^{r} q_j(k) \lambda_j^k$ für $\lambda_0, \dots, \lambda_r$ die Nullstellen des charakteristischen Polynoms ρ mit Vielfachheiten $m_0, \dots, m_r$ und $q_j \in P_{m_j - 1}$, $j = 0, \dots, r$.

Bew: 1) Vektorraumeigenschaften sind klar. Wegen $\alpha_m \neq 0$ ist eine Lösung eindeutig durch ihre Anfangswerte $y_0, \dots, y_{m-1}$ bestimmt. Der Raum der Anfangswerte $(y_0, \dots, y_{m-1})$ ist m-dimensional.

2) $w^0, \dots, w^{m-1}$ bilden eine Basis zu 1), d.h. jede Lösung der homogenen Gleichung lässt sich schreiben als $\sum_{k=0}^{m-1} a_k w^k$, $a_k \in \mathbb{R}$. Seien z_1, z_2 zwei Lösungen von (*), dann ist $z_1 - z_2$ Lösung der homogenen Gleichung $\Rightarrow$ jede Lösung von (*) ist von der Form $z_1 + \sum_{k=0}^{m-1} a_k w^k$ für eine spezielle Lösung z_1 und $a_0, \dots, a_{m-1} \in \mathbb{R}$. Es ist nur noch zu zeigen, dass $z_1 = \sum_{k=0}^{\infty} b_k v^k$ eine spezielle Lösung ist, dann folgt $a_i = y_i$ wegen $v_i^k = 0 \ \forall i < m$. z_1 ist wohldefiniert, da $(z_1)_j = \sum_{k=0}^{\infty} b_k v_j^k = \sum_{k=0}^{j-m} b_k v_j^k$ für alle j nur endlich viele Summanden enthält.
Außerdem ist $\sum_{j=0}^{m} \alpha_j (z_1)_{l+j} = \sum_{k=0}^{\infty} b_k \sum_{j=0}^{m} \alpha_j v_{l+j}^k = b_l$, da $\sum_{j=0}^{m} \alpha_j v_{l+j}^k = \begin{cases} 0, & l < k \\ \frac{1}{\alpha_m} \sum_{j=0}^{m} \alpha_j w_{l-k-1+j}^{m-1} & \text{sonst} \end{cases}$
und $\sum_{j=0}^{m} \alpha_j w_{l-k-1+j}^{m-1} = \begin{cases} 0, & l > k \\ \alpha_m, & l = k \end{cases}$.

3) Idee: Sei $y_k = \lambda_j^k$, dann ist $\sum_{i=0}^{m} \alpha_i y_{k+i} = \lambda_j^k \sum_{i=0}^{m} \alpha_i \lambda_j^i = 0$, also y_k eine Lösung.
Sei y_k eine Lösung und $Y_k := \begin{pmatrix} y_k \\ y_{k+1} \\ \vdots \\ y_{k+m-1} \end{pmatrix}$, dann erfüllt Y_k

$$Y_{k+1} = A Y_k \quad \text{für} \quad A = \begin{pmatrix} 0 & 1 & & \\ & 0 & \ddots & \\ & & \ddots & 1 \\ -\frac{\alpha_0}{\alpha_m} & -\frac{\alpha_1}{\alpha_m} & \cdots & -\frac{\alpha_{m-1}}{\alpha_m} \end{pmatrix}$$

siehe numerische Linear Algebra

Es ist $\rho(x) \overset{!}{=} \alpha_m \chi_A(x)$ für das charakteristische Polynom von A; insbesondere sind $\lambda_1, \dots, \lambda_r$ die Eigenwerte von A. Sei $J = X^{-1} A X$ die Jordan-Normalform von A, dann ist $Y_k = A^k Y_0 = X J^k X^{-1} Y_0$. Da zu jedem λ_i nur ein Eigenvektor (bis auf ein Vielfaches) existiert (Hausaufgabe), hat J die Form

$$J = \begin{pmatrix} J_1 & & \\ & \ddots & \\ & & J_r \end{pmatrix} \text{ mit Blöcken } J_i = \lambda_i I + \underbrace{N_i}_{N_i^{m_i} = 0} \in \mathbb{R}^{m_i \times m_i}, \quad N_i = \begin{pmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{pmatrix}.$$

Somit ist $J^k = \begin{pmatrix} J_1^k & & \\ & \ddots & \\ & & J_r^k \end{pmatrix}$ mit $J_i^k = (\lambda_i I + N_i)^k = \sum_{j=0}^{m_i - 1} \binom{k}{j} N_i^j \lambda_i^{k-j} = \sum_{j=0}^{m_i - 1} p_j^i(k) N_i^j \lambda_i^k$,
wobei $p_j^i(k) = \lambda_i^{-j} \binom{k}{j} = \lambda_i^{-j} \frac{k(k-1)\cdots(k-j+1)}{j!} \in P_j \subset P_{m_i - 1}$ ist. Folglich sind die Einträge von J^k Linearkombinationen aus Termen der Form $p^i(k) \lambda_i^k$, $i = 1, \dots, r$, $p^i \in P_{m_i - 1}$
$\Rightarrow$ auch $y_k = (Y_k)_0$ ist eine Linearkombination dieser Terme, d.h. $y_k = \sum_{i=1}^{r} q_i(k) \lambda_i^k$, $q_i \in P_{m_i - 1}$.
Dass alle Folgen der Form $y_k = \sum_{i=1}^{r} q_i(k) \lambda_i^k$ auch wirklich Lösungen sind, folgt daraus, dass die Dimension des Vektorraums aller Lösungen dieser Form genau $m_1 + \dots + m_r = m$ ist, also mit der Dimension des Lösungsraums übereinstimmt. □

Bew: (Wurzelbedingung) Notwendigkeit: Betrachte $y'(t)=0$, $y(0)=0$ auf $[0,1]$. Das m-Schritt-Verfahren lautet $\sum_{i=0}^{m} \alpha_i y_{k+i} = 0$. Sei λ eine Nullstelle von ρ mit $|\lambda|>1$. Betrachte die Lösung $u_k=0$ und die gestörte Lösung $v_k = \varepsilon \lambda^{k-m}$ mit Anfangswert-Rundungsfehler $|F_k| = |v_k - u_k| \le \varepsilon$, $k=0,\dots,m-1$. Für Schrittweite h sei $N_h = \lfloor \frac{1}{h} \rfloor$ und u_{N_h} bzw. v_{N_h} die Approximation an $y(1)$, dann gilt

$$|v_{N_h} - u_{N_h}| = \varepsilon \lambda^{N_h - m} \xrightarrow[h\to 0]{} \infty \quad \Rightarrow$$

das Verfahren ist instabil. Ist λ eine Nullstelle mit $|\lambda|=1$ und Vielfachheit ≥ 2, wiederhole das Argument für $v_k = \varepsilon k \lambda^k$.

Hinlänglichkeit: Das m-Schritt-Verfahren sei $\sum_{i=0}^{m} \alpha_i y_{k+i} = h \sum_{i=0}^{m} \beta_i f(t_{k+i}, y_{k+i}) =: h\Psi(t_k, y_k, \dots, y_{k+m}, h)$.

Sei u_k eine Lösung und v_k eine gestörte Lösung, setze $e_k = v_k - u_k$. Es gilt

$$e_k = F_k, \ k=0,\dots,m-1, \qquad \sum_{i=0}^{m} \alpha_i e_{k+i} = h\left[\Psi(t_k, v_k, \dots, v_{k+m}, h) + F_{k+m} - \Psi(t_k, u_k, \dots, u_{k+m}, h)\right] =: h\varepsilon_k \ \forall k.$$

Sei L die Lipschitz-Konstante von f, dann ist Ψ Lipschitz in $(y_k, \dots, y_{k+m})$ mit Konstante $\tilde{L} = L \sum_{i=0}^{m} |\beta_i|$.

$$\Rightarrow \|\varepsilon_k\| \le \|F_{k+m}\| + L \underbrace{\max_{j=0,\dots,m} \|e_{k+j}\|}_{=:\delta_k}$$

(e_k) kann geschrieben werden als $e = \sum_{k=0}^{\infty} h\varepsilon_k v^k + \sum_{k=0}^{m-1} e_k w^k$, wobei die w^k und somit auch v^k wegen der Wurzelbedingung beschränkt sind, $|v_j^k|, |w_j^k| \le M \ \forall j,k$ (in der Tat hat w^k die Form $w_\ell^k = \sum_{i=1}^{r} q_i(\ell) \lambda_i^\ell$ für q_i Polynome und λ_i Nullstellen von ρ; dies ist beschränkt).

$$\Rightarrow \|e_\ell\| = \left\| \sum_{k=0}^{\ell-m} h\varepsilon_k v_\ell^k + \sum_{k=0}^{m-1} F_k w^k \right\| \le hM \sum_{k=0}^{\ell-m} (\|F_{k+m}\| + L\delta_k) + M \sum_{k=0}^{m-1} \|F_k\|$$

$$\le M\|F\|_\infty [m + h(\ell - m + 1)] + hML \sum_{k=0}^{\ell-m} \delta_k \qquad \text{für } \|F\|_\infty = \max_{k=0,\dots,N_h} \|F_k\|$$

$$\Rightarrow \delta_\ell \le \max_{j=0,\dots,m} \left[M\|F\|_\infty [m + h(\ell + j - m + 1)] + hML \sum_{k=0}^{\ell-m+j} \delta_k \right] = M\|F\|_\infty [m + h(\ell+1)] + hML \sum_{k=0}^{\ell} \delta_k$$

$$\Leftrightarrow \delta_\ell (1 - hML) \le M\|F\|_\infty [m + h(\ell+1)] + hML \sum_{k=0}^{\ell-1} \delta_k$$

Wähle nun $h_0 = \frac{1}{2ML}$, somit $\delta_\ell \le 2M\|F\|_\infty [m + \underbrace{h(\ell+1)}_{\le b-a}] + h 2ML \sum_{k=0}^{\ell} \delta_k$. Hieraus folgt mittels vollständiger Induktion analog zum diskreten Gronwall-Lemma (Hausaufgabe)

$$\delta_\ell \le 2M[m + b - a] e^{\ell h 2ML} \|F\|_\infty$$

und somit $\|e_k\| \le \delta_k \le 2M[m+b-a] e^{2ML(b-a)} \|F\|_\infty \ \forall k$. □

Bem:
- Einschrittverfahren haben $\rho(x) = x - 1$ und sind somit stabil.
- Aus Quadraturen hergeleitete m-Schritt-Verfahren haben $\rho(x) = x^m - 1$ und sind somit stabil.

Konvergenz linearer Mehrschrittverfahren

Wieder gilt „Stabilität + Konsistenz = Konvergenz".

Thm: (Konvergenz) Ein stabiles und p. Ordnung konsistentes lineares Mehrschrittverfahren ist p. Ordnung konvergent für alle p. Ordnung konsistente Anfangswerte $y_0, \dots, y_{m-1}$ (d.h. Werte mit $\|y_i - y(t_i)\| = O(h^p)$, $i=0,\dots,m-1$).

Bew: Sei u_k die Mehrschrittlösung mit Anfangswerten $u_k = y_k$, $k = 0, \dots, m-1$, und $v_k = y(t_k)$. Fasse v_k als Störung von u_k auf, dann gilt

$$\|e_k\| = \|u_k - y(t_k)\| = \|u_k - v_k\| \le K \max_k \|F_k\| = K \max_k \|\tau_k\| = O(h^p). \qquad \square$$

Bem: Es gibt konsistente Mehrschritt-Verfahren, die unabhängig von der Differentialgleichung sind (Hausaufgabe). Diese sind natürlich nicht konvergent und somit nicht stabil.

Welche Konvergenzordnung (= Konsistenzordnung) kann ein lineares m-Schritt-Verfahren haben? Für p. Ordnung erhalten wir die $p+1$ Bedingungen $C_0 = \dots = C_p = 0$ für $C_0 = \sum_{i=0}^{m} \alpha_i = \rho(1)$, $C_1 = \sum_{i=0}^{m} (i\alpha_i - \beta_i) = \rho'(1) - \sigma(1)$, $C_q = \sum_{j=1}^{m} \frac{j^q}{q!}\alpha_j - \sum_{j=1}^{m} \frac{j^{q-1}}{(q-1)!}\beta_j$. Dies sind $p+1$ lineare Bedingungen für die $2m+1$ zu wählenden Koeffizienten $\alpha_0, \dots, \alpha_{m-1}, \beta_0, \dots, \beta_m$ (α_m kann zu 1 normiert werden) für implizite bzw. die $2m$ Koeffizienten $\alpha_0, \dots, \alpha_{m-1}, \beta_0, \dots, \beta_{m-1}$ für explizite Verfahren. Somit ist maximal erreichbare Konsistenzordnung $p = 2m$ für implizite und $p = 2m-1$ für explizite Verfahren – wir werden jedoch sehen, dass stabile Verfahren nur eine geringere Ordnung erreichen.

Thm: Ein lineares m-Schritt-Verfahren hat Konsistenzordnung p genau dann, wenn $\Phi: \mathbb{C} \to \mathbb{C}$,

$$\Phi(z) = \frac{\rho(z)}{\log(z)} - \sigma(z) \quad \text{in } z = 1 \text{ eine } p\text{-fache Nullstelle hat.}$$

$z = e^h$ ist um $h = 0$ eine bijektive holomorphe Abbildung

Bew: Φ hat p-fache Nullstelle in $z=1$ $\overset{!}{\iff}$ $\Phi(e^h) = \frac{1}{h}[\rho(e^h) - h\sigma(e^h)]$ hat in $h=0$ eine p-fache Nullstelle

$\iff$ $g(h) = h\Phi(e^h) = \sum_{i=0}^{m} (\alpha_i - h\beta_i) e^{ih}$ hat in $h=0$ $(p+1)$-fache Nullstelle

$\iff$ $g(0) = g'(0) = \dots = g^{(p)}(0) = 0$

($g(0) = C_0$, $g'(0) = C_1$, $g^{(p)}(0) = p! C_p$) $\qquad \square$

Das folgende Korollar zeigt, wie für vorgegebene $\alpha_0, \dots, \alpha_m$ (bzw. äquivalenterweise ρ) die bzgl. der Konsistenz optimalen β_i (bzw. σ) berechnet werden (natürlich können wir dies bereits mittels Taylorentwicklung).

Thm: Sei $\rho \in P_m$ mit $\rho(1) \overset{!}{=} 0$ (Konsistenz) und $0 \le n \le m$. Es gibt ein eindeutiges Polynom $\sigma \in P_n$, für das das zugehörige m-Schritt-Verfahren Konsistenzordnung $\ge n+1$ hat. Es sind $\sigma(z) = \sum_{i=0}^{n} c_i (z-1)^i$ die ersten $n+1$ Terme der Taylorentwicklung von $\frac{\rho(z)}{\log z}$.

Bew: $\frac{\rho(z)}{\log z}$ ist holomorph um $z=1$ und hat somit eine Taylorentwicklung $\sum_{i=0}^{n} c_i (z-1)^i + O((z-1)^{n+1})$

$\Rightarrow \Phi(z) = \frac{\rho(z)}{\log z} - \sigma(z) = \sum_{i=0}^{n} c_i (z-1)^i - \sigma(z) + O((z-1)^{n+1})$ hat genau dann eine $(n+1)$-fache Nullstelle in $z=1$, wenn $\sigma(z) = \sum_{i=0}^{n} c_i (z-1)^i$. $\qquad \square$

Bsp: Ein 2-Schritt-Verfahren mit $\rho(z) = (z-1)(z-\lambda)$ ist stabil für $\lambda \in [-1,1)$, und es ist

$$\frac{\rho(z)}{\log z} = \frac{(z-1)(1-\lambda+(z-1))}{\log(1+(z-1))} = 1-\lambda + \frac{3-\lambda}{2}(z-1) + \frac{5+\lambda}{12}(z-1)^2 - \frac{1+\lambda}{24}(z-1)^3 + O((z-1)^4)$$

$\Rightarrow$ maximale Konsistenzordnung wird erreicht für $\vartheta(z) = 1-\lambda + \frac{3-\lambda}{2}(z-1) + \frac{5+\lambda}{12}(z-1)^2 = -\frac{1+5\lambda}{12} + \frac{2-2\lambda}{3}z + \frac{5+\lambda}{12}z^2$

Für $\lambda \in (-1,1)$ ergibt sich 3. Ordnung, für $\lambda = -1$ das Simpson-Regel-Verfahren mit 4. Ordnung,

$$y_{k+2} - y_k = \frac{h}{3}(f_{k+2} + 4f_{k+1} + f_k).$$

Thm: (1. Dahlquistbarriere) Ein stabiles lineares m-Schritt-Verfahren hat höchstens Konsistenzordnung

- $m+2$, falls m gerade,
- $m+1$, falls m ungerade ist.

Bew: $T: \mathbb{C} \to \mathbb{C}$, $T(\zeta) = \frac{\zeta-1}{\zeta+1}$ bildet den komplexen Einheitsball $B = \{\zeta \in \mathbb{C} \mid |\zeta| \leq 1\}$ auf die linke Halbebene $H = \{z \in \mathbb{C} \mid \mathrm{Re}\, z \leq 0\}$ ab und hat dort Inverse $T^{-1}(z) = \frac{1+z}{1-z}$.

- $r(z) = \left(\frac{1-z}{2}\right)^m \rho\left(\frac{1+z}{1-z}\right)$, $s(z) = \left(\frac{1-z}{2}\right)^m \sigma\left(\frac{1+z}{1-z}\right)$ erfüllen $r, s \in P_m$
- Verfahren ist stabil $\Rightarrow \zeta = 1$ ist einfache Nullstelle von ρ $\Rightarrow z = 0$ ist einfache Nullstelle von r

  $\Rightarrow r(z) = a_1 z + a_2 z^2 + \dots + a_m z^m$, $a_1 \neq 0$; o.B.d.A nehmen wir $a_1 > 0$ an
- Verfahren ist stabil $\Rightarrow$ Nullstellen von ρ liegen in B $\Rightarrow$ Nullstellen z_i von r liegen in H

  $\Rightarrow r(z) = C z (z-z_1)\cdots(z-z_m) = a_1 z + a_2 z^2 + \dots + a_m z^m$ hat $a_1, a_2, \dots, a_m \geq 0$
- Verfahren ist konsistent von Ordnung $p \geq m$ $\Rightarrow \phi$ hat p-fache Nullstelle in $\zeta = 1$

  $\Rightarrow q(z) = \left(\frac{1-z}{2}\right)^m \phi\left(\frac{1+z}{1-z}\right) = \frac{1}{\log\frac{1+z}{1-z}} r(z) - s(z)$ hat p-fache Nullstelle in $z = 0$

  $\Rightarrow$ die ersten p Terme der Taylorentwicklung $\frac{z}{\log\frac{1+z}{1-z}} \frac{r(z)}{z} = b_0 + b_1 z + b_2 z^2 + \dots$ stimmen mit s überein

  $\Rightarrow$ da $p \geq m$ aber $s \in P_m$ muss gelten $b_i = 0$ für $i = m+1, \dots, p-1$
- Die Taylorentwicklung von $\frac{z}{\log\frac{1+z}{1-z}}$ ist von der Form $c_0 + c_2 z^2 + c_4 z^4 + \dots$ mit $c_2, c_4, c_6, \dots < 0$

  (die Funktion ist gerade, somit sind $c_1 = c_3 = c_5 = \dots = 0$) (Hausaufgabe)
- Wir setzen $a_i = 0$ für $i > m$, dann gilt

$$b_0 = c_0 a_1$$
$$b_1 = c_0 a_2$$
$$\vdots$$
$$b_{2i} = c_0 a_{2i+1} + c_2 a_{2i-1} + \dots + c_{2i} a_1$$
$$b_{2i+1} = c_0 a_{2i+2} + c_2 a_{2i} + \dots + c_{2i} a_2 \qquad , i = 1, 2, \dots$$

- Für $k \geq m$ gerade gilt $b_k = \underbrace{c_0}_{<0} a_{k+1} + c_2 a_{k-1} + \dots + c_k a_1 < 0$, insbesondere ist $b_k \neq 0$.

  $\Rightarrow \{m+1, \dots, p-1\}$ darf keine gerade Zahl enthalten $\Rightarrow p \leq \begin{cases} m+2, & m \text{ gerade} \\ m+1, & m \text{ ungerade} \end{cases}$ □

Bsp: Das Simpson-Regel-Verfahren hat höchstmögliche Konsistenzordnung.

Absolute Stabilität & steife Probleme

Auch wenn ein Verfahren stabil und konvergent ist, kann die numerische Lösung manchmal unbefriedigend sein, insbesondere wenn qualitative Eigenschaften der exakten Lösung erst für unverhältnismäßig kleine Schrittweiten h korrekt reproduziert werden. Ein besonders wichtiges Beispiel für eine solche Eigenschaft ist die Tatsache (A), dass die Lösung von $y'(t) = \lambda y(t)$, $\mathrm{Re}(\lambda) < 0$, betragsmäßig mit der Zeit abnimmt. Das Konzept der absoluten Stabilität untersucht, wann ein Verfahren diese Eigenschaft korrekt reproduziert.

```
% löse y'(t)=-20y(t) mittels Euler-Verfahren
for h = [.01 .05 .1 0.2]
  t = 0:h:1;
  y = ones(size(t));
  for k = 2:length(t)
    y(k) = y(k-1) - h*20*y(k-1);
  end
  plot(t,y,'Linewidth',3); hold on; pause;
end
```

Ein Verfahren $y_{k+1} = y_k + h\,\varphi(t_k, y_0, \dots, y_{k+1}, h)$ für $y'(t) = \lambda y(t)$ mit Schrittweite h kann auch aufgefasst werden als Verfahren für $y'(t) = \tilde{\lambda} y(t)$, $\tilde{\lambda} = \frac{\lambda}{\gamma}$, mit Schrittweite $\tilde{h} = h\gamma$; für das Eulerverfahren beispielsweise ist $y_{k+1} = y_k + h\lambda y_k$ äquivalent zu $y_{k+1} = y_k + \tilde{h}\tilde{\lambda} y_k$. Daher spielt bei der Analyse nur das Produkt $h\lambda$ eine Rolle.

Def: (absolute Stabilität) Das Gebiet der <u>absoluten Stabilität</u> eines numerischen Verfahrens ist

$$R_A = \{ h\lambda \in \mathbb{C} \mid \text{es gilt } |y_{k+1}| < |y_k| \ \forall k \text{ für die Lösung } y_k \text{ des Verfahrens zu } y'(t) = \lambda y(t) \text{ mit Schrittweite } h \}.$$

Das Verfahren heißt <u>A-stabil</u>, wenn $R_A \supset \{ z \in \mathbb{C} \mid \mathrm{Re}(z) < 0 \}$ (d.h. (A) gilt für alle h & $\mathrm{Re}\,\lambda < 0$)

und <u>A(α)-stabil</u>, wenn $R_A \supset \{ z \in \mathbb{C} \mid \mathrm{Re}(z) < 0,\ \frac{\mathrm{Im}(z)}{\mathrm{Re}(z)} < \tan\alpha \}$.

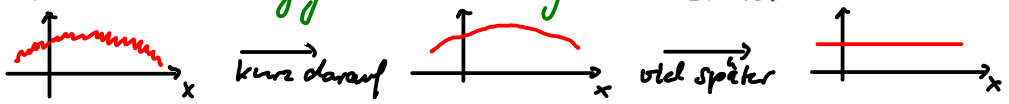
Warum ist A-Stabilität also im Grunde eine Art Stabilität für alle h, wichtig? Sollte man nicht einfach h klein genug wählen, dass das Verfahren stabil ist? Dies verursacht bei steifen Problemen allerdings unverhältnismäßig großen Aufwand.

Def: (steifes Problem) Eine vektorwertige Differentialgleichung $y'(t) = f(t, y(t))$ heißt <u>steif</u>, wenn die Jakobimatrix $J = Df$ an einer Stelle $(\hat{t}, \hat{y})$ diagonalisierbar ist, ihre Eigenwerte $\lambda_1, \dots, \lambda_r$ $\mathrm{Re}(\lambda_i) < 0$ erfüllen, und $\mathrm{Re}(\lambda_j) \ll \mathrm{Re}(\lambda_\ell)$ für mindestens zwei Indizes j, ℓ.

Bem: · In der Nähe von $(\hat{t}, \hat{y})$ verhält sich die Differentialgleichung etwa wie ihre Linearisierung $y'(t) = f(\hat{t}, \hat{y}) + J(y(t) - \hat{y})$, bzw. nach der Variablentransformation $\tilde{y}(t) = X^{-1}(y(\hat{t}+t) - \hat{y})$, wobei $J = X \Lambda X^{-1}$ die Diagonalisierung ist, wie $\tilde{y}'(t) = b + \Lambda \tilde{y}(t)$ für $b = X^{-1} f(\hat{t}, \hat{y})$. Daher reicht es, die Eigenschaften steifer Systeme an linearen diagonalen Differentialgleichungen zu untersuchen.

· Man betrachtet nur $\mathrm{Re}\,\lambda_i < 0$, da für $\mathrm{Re}\,\lambda_i > 0$ exponentielles Wachstum auftritt und die Schrittweite h für ausreichende Genauigkeit ohnehin sehr klein sein muss.

· $\mathrm{Re}(\lambda_j) \ll \mathrm{Re}(\lambda_k)$ bedeutet stark unterschiedliche Abklingzeiten. Dies taucht in vielen Systemen auf, z.B. werden bei der Wärmeleitung räumlich schnell oszillierende Inhomogenitäten schneller geglättet als langsam oszillierende.



Bsp: · $\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}'(t) = \begin{pmatrix} -100 & 0 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}(t)$, $\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}(0) = \begin{pmatrix} \varepsilon \\ 1 \end{pmatrix}$, hat die Lösung

$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}(t) = \begin{pmatrix} \varepsilon e^{-100t} \\ e^{-2t} + \frac{\varepsilon}{98}(e^{-2t} - e^{-100t}) \end{pmatrix}$, y_1 klingt daher sehr viel schneller ab als y_2.

Ist $\varepsilon = 10^{-6}$, ist y_1 praktisch vernachlässigbar, und die Differentialgleichung für y_2 wird $y_2' = -2y_2$.

```
for h = [1e-4 1e-1]    % für verschiedene Schrittweiten
  t = 0:h:1;
  n = length(t);
  y = zeros(2,n);      % Lösung des Systems
  y2 = zeros(1,n);     % Lösung für y2 unter Vernachlässigung von y1
  y(:,1) = [1e-6;1];
  y2(1) = 1;
  for k = 2:n          % Eulerverfahren
    y(:,k) = y(:,k-1) + h*[-100 0;1 -2]*y(:,k-1);
    y2(k)  = y2(k-1)  - h*2*y2(k-1);
  end
  plot(t,y2,'b','Linewidth',3); hold on; pause;
  plot(t,y,'r','Linewidth',3); hold off; pause;
end
```

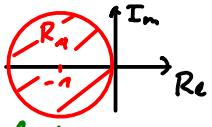
· Van der Pol-Oszillator $y'' + \mu(y^2 - 1)y' + y = 0$, $y(0) = 2$, $y'(0) = 1$

$\Leftrightarrow \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}' = \begin{pmatrix} y_2 \\ \mu(1 - y_1^2)y_2 - y_1 \end{pmatrix}$, $\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}(0) = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$

```
h = .0201021698315;
t = 0:h:60;
n = length(t);
y = zeros(2,n);
y(:,1) = [2;1];
for mu = [0 1 2 5 10 20 50]
  mu
  for k = 2:n
    y(:,k) = y(:,k-1) + h*[y(2,k-1);mu*(1-y(1,k-1)^2)*y(2,k-1)-y(1,k-1)];
  end
  plot(t,y,'Linewidth',3); pause;
  if mu == 20
    axis([50 52 2 3]); pause;
  end
end
```

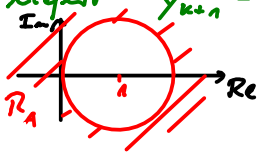
Die Beispiele zeigen: In Bereichen, wo eine Komponente fast verschwindet, könnte man auch mit sehr großem h eine gute Genauigkeit erzielen – die Schritte in der verschwindenden Komponente führen dann jedoch zu Instabilität! Bei A-stabilen Verfahren gibt es dieses Problem nicht – das Verfahren ist stabil für alle h!

Bsp: · Das explizite Euler-Verfahren $y_{k+1} = y_k + hf(t_k, y_k)$ angewandt auf $y' = \lambda y$ liefert $y_{k+1} = y_k(1+h\lambda) = y_0(1+h\lambda)^{k+1}$. Dies ist strikt monoton fallend genau dann, wenn $|1+\lambda h| < 1$

$\Rightarrow R_A = \{z \in \mathbb{C} \mid |z+1| < 1\}$ 

$\Rightarrow$ nicht A-stabil

· Das implizite Euler-Verfahren liefert $y_{k+1} = y_k + h\lambda y_{k+1}$, also $y_{k+1} = \frac{1}{1-h\lambda} y_k = \frac{1}{(1-h\lambda)^{k+1}} y_0$

$\Rightarrow R_A = \{z \in \mathbb{C} \mid |1-z| > 1\}$ 

$\Rightarrow$ A-stabil

Thm: 1) Ein m-stufiges Runge-Kutta-Verfahren angewandt auf $y' = \lambda y$ liefert $y_{k+1} = R(\lambda h) y_k$ für ein $R \in P_m$ (explizites Verfahren) bzw. eine rationale Funktion R vom Typ (m,m).

2) Ist das Verfahren konsistent der Ordnung p, gilt $|R(z) - e^z| = |R(z) - \sum_{i=0}^{\infty} \frac{z^i}{i!}| = O(z^{p+1})$.

3) $R_A = \{z \in \mathbb{C} \mid |R(z)| < 1\}$

4) Kein explizites Runge-Kutta-Verfahren ist A-stabil oder A(0)-stabil.

Bew: 1) Hausaufgabe

2) Betrachte $y' = y$, $y(0) = 1$. Das Verfahren ist konvergent von Ordnung p, somit ist

$$O(h^{p+1}) = h\tau_0 = y(h) - y(0) - h\varphi(t_0, y(0), y(h), h) - \underbrace{[y_n - y_0 - h\varphi(t_0, y_0, y_n, h)]}_{0}$$

$$= (y(h) - y_0)(1 + hL) = (e^h - R(h))(1+hL);$$

(L: Lipschitz-Konstante von φ)

Da R und $\exp$ um 0 herum holomorph sind, folgt die Behauptung für $z \in \mathbb{C}$.

3) trivial

4) Für jedes $R \in P_m$ mit $m > 0$ gilt $\lim_{z \to -\infty} |R(z)| = \infty \Rightarrow \{z \in \mathbb{C} \mid \mathrm{Re}(z) < 0, \mathrm{Im}(z) = 0\} \notin R_A$. □

Für Mehrschritt-Verfahren angewandt auf $y' = \lambda y$ hängt die Monotonie der y_k von allen vorgegebenen Anfangswerten ab – daher wird die Definition der absoluten Stabilität hier oft angepasst. Wir wollen hier die strikte Monotonie $|y_{k+1}| < |y_k|$ erst für $k \geq K$ mit einem $K > 0$ fordern.

Thm: (absolute Stabilität linearer Mehrschritt-Verfahren) $\gamma \in \mathbb{C}$ liegt genau dann im absoluten Stabilitätsgebiet R_A eines linearen Mehrschritt-Verfahrens, wenn sein Stabilitätspolynom $\pi(z;\gamma) = \rho(z) - \gamma\sigma(z)$ nur Nullstellen $z \in \mathbb{C}$ mit $|z| < 1$ besitzt.

Beweis: Das Verfahren angewandt auf $y' = \lambda y$ liefert $\sum_{i=0}^{m} \alpha_i y_{k+i} = h \sum_{i=0}^{m} \beta_i \lambda y_{k+i}$, also mit $\gamma = h\lambda$ $\sum_{i=0}^{m} (\alpha_i - \gamma\beta_i) y_{k+i} = 0$. Dies hat Lösungen der Form $y_k = \sum_{j=1}^{r} p_j(k) \lambda_j^k$ für Nullstellen $\lambda_1, \dots, \lambda_r$ des charakteristischen Polynoms $\sum_{i=0}^{m} (\alpha_i - \gamma\beta_i) z^i = \rho(z) - \gamma\sigma(z)$ und Polynome $p_j \in \mathbb{P}_{m_j - 1}$, wobei m_j die Vielfachheit von λ_j ist. $\Rightarrow$ $|y_k|$ ist genau dann strikt monoton fallend für k groß genug, wenn $|\lambda_j| < 1 \ \forall j$. □

Thm: (2. Dahlquist-Barriere) · Kein explizites lineares Mehrschritt-Verfahren ist A- oder A(0)-stabil.

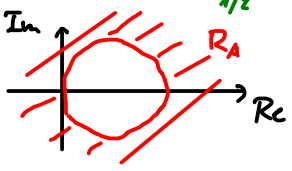
· A-stabile lineare Mehrschritt-Verfahren haben Konsistenzordnung ≤ 2.

· Das einzige A(0)-stabile lineare m-Schritt-Verfahren mit Konsistenzordnung $> m$ ist die Trapezregel $y_{k+1} - y_k = \frac{f(t_k, y_k) + f(t_{k+1}, y_{k+1})}{2}$.

Bsp: Die zweite Ordnung Rückwärts-Differenzen-Formel ist $y_{k+2} - \frac{4}{3} y_{k+1} + \frac{1}{3} y_k = \frac{2}{3} h f(t_{k+2}, y_{k+2})$.

$\pi(z;\gamma) = (1 - \frac{2}{3}\gamma) z^2 - \frac{4}{3} z + \frac{1}{3}$ hat Nullstellen $z_{1/2} = \frac{2 \mp \sqrt{1 + 2\gamma}}{3 - 2\gamma} = \frac{1}{2 \mp \sqrt{1 + 2\gamma}}$

$\Rightarrow R_A = \{\gamma \in \mathbb{C} \mid |2 \mp \sqrt{1+2\gamma}| > 1\}$ $\Rightarrow$ A-stabil



Wir wollen noch eine Methode angeben, wie man das Gebiet der absoluten Stabilität eines Mehrschritt-Verfahrens ermitteln kann.

Def: (Schur-Polynom) $\phi(z) = c_m z^m + \dots + c_1 z + c_0$, $c_0 \neq 0$, $c_m \neq 0$, heißt Schur-Polynom, wenn alle Nullstellen betragsmäßig kleiner 1 sind.

Thm: (Schur-Kriterium) $\phi(z) = c_m z^m + \dots + c_1 z + c_0$ ist ein Schurpolynom genau dann, wenn $|\hat\phi(0)| > |\phi(0)|$ und ϕ_1 ein Schur-Polynom ist für $\hat\phi(z) = \overline{c_0} z^m + \dots + \overline{c_{m-1}} z + \overline{c_m}$ und $\phi_1(z) = \frac{1}{z}[\hat\phi(0)\phi(z) - \phi(0)\hat\phi(z)] \in \mathbb{P}_{m-1}$. $\Rightarrow$ Schur-Kriterium reduziert Polynomgrad $\rightsquigarrow$ leichtere Nullstellenberechnung

Bsp: $y_{k+2} - y_k = \frac{h}{2}(f(t_{k+1}, y_{k+1}) + 3 f(t_k, y_k))$ hat $\pi(z;\gamma) = z^2 - \frac{\gamma}{2} z - (1 + \frac{3}{2}\gamma)$.

$\Rightarrow \hat\pi(z;\gamma) = -(1 + \frac{3}{2}\overline{\gamma}) z^2 - \frac{\overline{\gamma}}{2} z + 1$, $|\hat\pi(0;\gamma)| = 1 > |1 + \frac{3}{2}\gamma| = |\pi(0;\gamma)| \Leftrightarrow \gamma \in B = \{\gamma \in \mathbb{C} \mid |\gamma - (-\frac{2}{3})| < \frac{2}{3}\}$

& $\pi_1(z;\gamma) = \frac{1}{z}\left[z^2 - \frac{\gamma}{2} z - (1 + \frac{3}{2}\gamma) + (1 + \frac{3}{2}\gamma)\left(-(1 + \frac{3}{2}\overline{\gamma}) z^2 - \frac{\overline{\gamma}}{2} z + 1\right)\right] = -\left(\frac{\gamma + \overline{\gamma}}{2} + 3\left|\frac{\gamma}{2}\right|^2\right)(3z + 1)$ ist Schur $\Rightarrow R_A = B$.

Extrapolation & Schrittweitensteuerung

heute: Adaptivität

Mit Hilfe von Richardson-Extrapolation kann die Konvergenzordnung eines Verfahrens erhöht werden:
Sei $y_h(t)$ die diskrete Lösung eines Verfahrens mit Konvergenzordnung p, $y(t)$ die exakte Lösung, jeweils zur Zeit t. Hat der Diskretisierungsfehler eine Taylorentwicklung in der Schrittweite h, $e_h(t) = y_h(t) - y(t) = c(t)\,h^p + O(h^{p+1})$, so gilt bspw.

Extrapolation: $\bar{y}_h(t) = \dfrac{2^p y_{\frac{h}{2}}(t) - y_h(t)}{2^p - 1}$ hat einen Fehler $\bar{y}_h(t) - y(t) = O(h^{p+1})$ (*)

Fehlerschätzung: $y_{h/2}(t) - y(t) = \dfrac{y_{h/2}(t) - y_h(t)}{2^p - 1} + O(h^{p+1})$ (**)

Leider hat der Diskretisierungsfehler für lineare Mehrschritt-Verfahren im Allgemeinen keine Taylorentwicklung, jedoch für Einschritt-Verfahren:

Thm: (Taylorentwicklung Diskretisierungsfehler) Seien $f, \varphi \in C^{p+1}$ und sei $y_{k+1} = y_k + h\varphi(t_k, y_k, h)$ ein explizites Einschritt-Verfahren mit konstanter Schrittweite h und Konsistenzordnung p, d.h. der Abschneidefehler ist $\tau_h(t, y(t)) = \tau(t)\,h^p + O(h^{p+1})$. Sei D die Lösung von

$$D'(t) = \frac{\partial}{\partial y} f(t, y(t))\, D(t) - \tau(t), \quad D(t_0) = 0, \qquad \odot$$

dann ist $e_h(t_k) = y_h(t_k) - y(t_k) = y_k - y(t_k) = D(t_k)\,h^p + O(h^{p+1})$.

Bem: Für die höheren Terme in der Taylorentwicklung gelten entsprechende Resultate mit analogen Beweisen.

Bew:
$$e_h(t_{k+1}) = y_h(t_{k+1}) - y(t_{k+1}) = y_h(t_k) + h\,\varphi(t_k, y_h(t_k), h) - \big(y(t_k) + h\varphi(t_k, y(t_k), h) + \tau(t_k)h^{p+1} + O(h^{p+2})\big)$$
$$= e_h(t_k) + h\underbrace{\frac{\partial}{\partial y}\varphi(t_k, y(t_k), h)}_{= \frac{\partial}{\partial y}\varphi(t_k, y(t_k), 0) + O(h)}\, e_h(t_k) - \tau(t_k)h^{p+1} + O\Big(h^{p+2} + \Big\|\frac{\partial^2}{\partial y^2}\varphi\Big\| \|e_h(t_k)\|^2 h\Big)$$
$$\underbrace{= \frac{\partial}{\partial y}\varphi(t_k, y(t_k), 0)}_{= f(t_k, y(t_k))} + O(h)$$
$$= e_h(t_k) + h\,\frac{\partial}{\partial y} f(t_k, y(t_k))\, e_h(t_k) - \tau(t_k)h^{p+1} + O\Big(h^{p+2} + \Big\|\frac{\partial^2}{\partial y^2}\varphi\Big\| \|e_h(t_k)\|^2 h + \|e_h(t_k)\| h^2\Big)$$

Sehe $D_h(t_k) = e_h(t_k)\,h^{-p}$, dann ist

$$D_h(t_{k+1}) = D_h(t_k) + h\Big(\frac{\partial}{\partial y} f(t_k, y(t_k)) D_h(t_k) - \tau(t_k)\Big) + O\big(h^2 + \|D_h(t_k)\|^2 h^{p+1} + \|D_h(t_k)\| h^2\big)$$

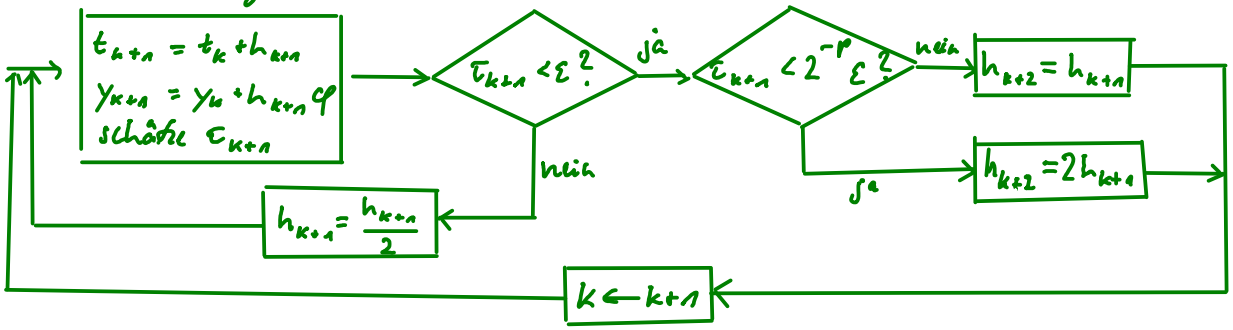
ein Einschritt-Verfahren zur Berechnung von $\odot$ (Euler-Verfahr plus $O(h)$-Term) mit Konsistenzordnung 1 $\Rightarrow D_h(t) = D(t) + O(h)$ und somit $e_h(t) = D(t)h^p + O(h^{p+1})$. $\square$

Somit lässt sich die Richardson-Extrapolation (*) und Fehlerschätzung (**) anwenden.
Man kann auch den lokalen Diskretisierungsfehler in jedem Schritt schätzen und diesen zur Anpassung der Schrittweite nutzen, so dass eine bestimmte Fehlerschranke nicht überschritten wird.
Wir wissen, dass ein Fehler δ im $k$ten Schritt eines Einschritt-Verfahrens zur Zeit T um $(T - t_k)\,e^{L(T - t_k)}$ verstärkt werden kann (mit L der Lipschitz-Konstante des Verfahrens).

$\Rightarrow$ Gibt man als Fehlerschranke für den lokalen Diskretisierungsfehler $\|\tau_k\| \le \frac{\varepsilon}{T(T-t_k)} e^{-L(T-t_k)}$ vor, so kann der Fehler zur Zeit T geschätzt werden durch $\sum_{k \text{ mit } t_k < T} \|h_k \tau_k\| (T-t_k) e^{L(T-t_k)} \le \sum_k \frac{h_k \varepsilon}{T} = \varepsilon$.

Oft wird der Einfachheit halber jedoch nur $\|\tau_k\| \le \varepsilon$ gefordert (insbesondere, da L ggfs. schwer zu ermitteln ist). Ein einfaches Verfahren zur Schrittweitensteuerung ist wie folgt.

Alg: (Schrittweitensteuerung) Sei $y_{k+1} = y_k + h_k \varphi(t_k, y_k, h_{k+1})$ ein Einschritt-Verfahren mit Konsistenzordnung p und $\varepsilon > 0$.



Wird h verdoppelt, wächst der Abschneidefehler etwa um 2^p $\Rightarrow$ h wird nur verdoppelt, wenn $\tau < 2^{-p}\varepsilon$

Der Abschneidefehler kann auf verschiedene Arten geschätzt werden:

- Mit Taylorentwicklung erhält man $\tau_k = C_k h_k^p + O(h_k^{p+1})$, wobei C_k von f und seinen ersten p Ableitungen an (t_k, y_k) abhängt. Dies wird selten genutzt, da die Ableitungen vom Nutzer erst bestimmt werden müssten und da es für hohes p viele zusätzliche Funktionsauswertungen kostet.
- Man kann einen besseren Wert $\hat{y}_{k+1} = y_k + h_{k+1} \hat{\varphi}(t_k, y_k, h_{k+1})$ mit einem Verfahren höherer Ordnung $q > p$ berechnen und dann $\tau_{k+1} \approx (\hat{y}_{k+1} - y_{k+1})/h_{k+1}$ schätzen. Dies ist der Standard; hierzu werden aus Effizienzgründen Runge-Kutta-Verfahren mit Ordnung p & q verwendet, die die selben Funktionsauswertungen nutzen (z.B. Dormand-Prince-Verfahren). Oft wird nach Akzeptanz einer Schrittweite h_{k+1} die (höhere Ordnungs-)Lösung $\hat{y}_{k+1}$ benutzt, auch wenn der Fehlerschätzer nur für y_{k+1} gilt.
- Man kann einen Wert $\hat{y}_{k+1}$ mit zwei Schritten halber Schrittweite berechnen. Mit der exakten Lösung $\tilde{y}$ zum Anfangswert $\tilde{y}(t_k) = y_k$ ergeben sich Fehler

$$y_{k+1} - \tilde{y}(t_{k+1}) = c\, h_{k+1}^{p+1} + O(h_{k+1}^{p+2}), \quad \hat{y}_{k+1} - \tilde{y}(t_{k+1}) = 2c\left(\frac{h_{k+1}}{2}\right)^{p+1} + O(h_{k+1}^{p+2}) \quad \text{(Hausaufgabe)}$$

$$\Rightarrow \tilde{y}(t_{k+1}) = \frac{2^p \hat{y}_{k+1} - y_{k+1}}{2^p - 1} + O(h_{k+1}^{p+2}), \quad \tau_{k+1} = \frac{\tilde{y}(t_{k+1}) - y_{k+1}}{h_{k+1}} = \frac{2^p}{2^p - 1} \frac{\hat{y}_{k+1} - y_{k+1}}{h_{k+1}} + O(h_{k+1}^{p+1}).$$

```
% Anfangswertproblem (Van der Pol-Oszillator)
f = @(y) [y(2);8*(1-y(1)^2)*y(2)-y(1)];
y = [2;0];
t = 0;

% Einschrittverfahren
phi = @(y,h) (f(y)+f(y+h*f(y)))/2;
p = 2;

% adaptive Lösung
epsilon = 1e-1;
h = .1;
T = zeros(1,1000); H = T; Y = [T;T]; counter = 0;
while t < 30
  yNew = y + h*phi(y,h);
  yNew2 = y + h/2*phi(y,h/2);
  yNew2 = yNew2 + h/2*phi(yNew2,h/2);
  tau = max(abs(2^p/(2^p-1)*(yNew2-yNew)/h));
  if tau > epsilon
    h = h/2;
  else
    t = t+h;
    y = yNew;
    if tau < epsilon/2^p
      h = h*2;
    end
    counter = counter+1;
    T(counter) = t;
    H(counter) = h;
    Y(:,counter) = y;
  end
end
T(counter+1:end) = []; H(counter+1:end) = []; Y(:,counter+1:end) = [];
subplot(1,2,1); plot(T,Y(1,:),'.','Markersize',20);
subplot(1,2,2); plot(T,H,'.','Markersize',20);
```

Randwertprobleme

Im Gegensatz zu Anfangswertproblemen werden bei Randwertproblemen Werte der gesuchten Funktion an zwei oder mehr Zeitpunkten vorgegeben.

Bsp: · $\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}'(t) = \begin{pmatrix} y_2 \\ y_3 \\ y_1 - y_2 + y_3 \end{pmatrix}(t)$, $y_1(0) = 1$, $y_1(1) = e$, $y_2(2) = e^2$ hat Vorgaben zu verschiedenen Funktionskomponenten an drei verschiedenen Zeitpunkten. Die Lösung ist $y_1 = y_2 = y_3 = e^t$.

· $y''' - y'' + y' - y = 0$, $y(0) = 1$, $y(1) = e$, $y(2) = e^2$ ist äquivalent zu obigem Problem mit $y_i = y^{(i-1)}$.

· $y'' - 2y^3 = 0$, $y(1) = 1$, $y'(2) + y(2)^2 = 0$ hat nichtlineare Randbedingungen (am Rand des Intervalls $[1,2]$). Die Lösung ist $y(t) = \frac{1}{t}$.

Randwertprobleme sind partiellen Differentialgleichungsproblemen sehr ähnlich - dementsprechend sind Wohlgestelltheit und Numerik erheblich komplizierter und von Fall zu Fall verschieden.

Bsp: · $-y'' + \alpha^2 y = 0$, $\alpha \in \mathbb{R}$, $y(a) = y_0$, $y(b) = y_1$ hat eine eindeutige Lösung:

Sei $y'(a) = z$; wir wissen, dass das Anfangswertproblem $\begin{pmatrix} y \\ y' \end{pmatrix}'(t) = \begin{pmatrix} y' \\ \alpha^2 y \end{pmatrix}(t)$, $\begin{pmatrix} y \\ y' \end{pmatrix}(a) = \begin{pmatrix} y_0 \\ z \end{pmatrix}$ eine eindeutige Lösung hat - diese ist $y(t) = \frac{y_0 + z/\alpha}{2} e^{\alpha(t-a)} + \frac{y_0 - z/\alpha}{2} e^{-\alpha(t-a)}$.

$y(b) = y_1$ kann nun für z gelöst werden.

· $-y'' - \alpha^2 y = 0$, $\alpha \in \mathbb{R}$, $y(0) = y_0$, $y(\pi) = y_1$ ist nicht immer wohlgestellt:

Sei $y'(0) = z$; wieder hat das Anfangswertproblem eine eindeutige Lösung $y(t) = z \sin \alpha t + y_0 \cos \alpha t$. Dies kann $y(\pi) = y_1$ i.A. nur für ein z erfüllen, wenn $\alpha \notin \mathbb{Z}$.

Wir geben ohne weitere Analysis Verfahren zur numerischen Lösung an.

Schießverfahren

Betrachte das Randwertproblem $y'(t) = f(t, y(t))$, $g(y(a), y(b)) = 0$ (RWP)

mit $f: \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$, $g: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$, die n Bedingungen von (AWP) wurden also durch n Bedingungen in $y(a)$ und $y(b)$ ersetzt. Sei nun $y(t; y_0)$ die Lösung des Anfangswertproblems

$$y'(t) = f(t, y(t)), \quad y(a) = y_0. \qquad \text{(AWP)}$$

Def: (Schießverfahren) Ein Schießverfahren zu (RWP) ist ein Verfahren, das iterativ eine Nullstelle y_0 von $g(y_0, y(b; y_0))$ sucht. Hierzu löst es (AWP) mit den bekannten Methoden und verbessert iterativ y_0, bis $g(y_0, y(b; y_0)) = 0$.

Ein oft genutztes Verfahren zur Nullstellensuche einer Funktion $F(x)$ ist das Newton-Verfahren

$$x_{neu} = x_n - \frac{F'(x_n)}{F(x_n)}$$

Seine Anwendung auf $F(x) = g(x, y(b;x))$ setzt Existenz von $F'(x) = \partial_1 g(x, y(b;x)) + \partial_2 g(x, y(b;x))\, \partial_x y(b;x)$ voraus.

Thm: (Glattheit des Anfangswertproblems) Sei $y(t;y_0)$ die eindeutige Lösung zu (AWP), f differenzierbar in y. Das (lineare) Matrix-wertige Anfangswertproblem $D'(t) = \frac{\partial f}{\partial y}(t, y(t;y_0))\, D(t)$, $D(a) = I \in \mathbb{R}^{n\times n}$ besitzt eine eindeutige Lösung $D(t;y_0)$, und es ist $\partial_{y_0} y(t;y_0) = D(t;y_0)$.

Bew: Eindeutige Lösung folgt aus Satz von Picard und Lindelöf. Sei o.B.d.A. $n=1$, d.h. $y_0 \in \mathbb{R}$ (sonst betrachte Komponenten von y_0 einzeln). $w_h(t) = \frac{y(t;y_0+h) - y(t;y_0)}{h} - D(t;y_0)$ erfüllt

$$w_h'(t) = \frac{f(t, y(t;y_0+h)) - f(t, y(t;y_0))}{h} - \frac{\partial f}{\partial y}(t, y(t;y_0))\, D(t;y_0)$$

$$= \frac{\partial f}{\partial y}(t; y(t;y_0)) \underbrace{\left[\frac{y(t;y_0+h) - y(t;y_0)}{h} - D(t;y_0)\right]}_{w_h(t)} + o(1) \quad , \; w_h(a) = 0.$$

Nach Gronwalls Lemma gilt $w_h(t) \to 0$ für alle t. □

Damit das Newton-Verfahren auch nach der Diskretisierung funktioniert, sollte die Approximation D_k von $D(t_k)$ die Ableitung von y_k nach y_0 sein. Hierzu kann das Verfahren, z.B. $y_{k+1} = y_k + \varphi(t_k, y_k, h_{k+1})$, nach y_0 abgeleitet werden, $D_{k+1} = D_k + \frac{\partial}{\partial y}\varphi(t_k, y_k, h_{k+1}) D_k$.

Ist der erste Versuch y_0 schlecht gewählt, kann es passieren, dass $y(b;y_0)$ viel zu weit von dem gewünschten Wert entfernt ist oder dass $y(t;y_0)$ noch nicht einmal existiert. Dieses Risiko ist beim Mehrfach-Schießverfahren reduziert.

Def: (Mehrfach-Schieß-Verfahren) Sei $a = t_0 < t_1 < \ldots < t_m = b$ und $y^i(t;y_i)$ die Lösung von $y' = f(t,y)$, $y(t_i) = y_i$. Das Mehrfach-Schieß-Verfahren sucht eine Nullstelle $(y_1, \ldots, y_m)$ von $(y(t_1;y_0) - y_1, \ldots, y(t_m; y_{m-1}) - y_m, g(y_0, y_m))$.

Matrix-Verfahren (Finite Differenzen-Verfahren)

Der Einfachheit halber betrachten wir eine lineare gewöhnliche Differentialgleichung zweiter Ordnung – das Konzept lässt sich leicht auf nichtlineare Differentialgleichungen oder auf Gleichungen erster Ordnung übertragen. Wir betrachten zweite Ordnung, da dies die klassische Situation ist: Klassische Randwertprobleme sind oft elliptische partielle Differentialgleichungen 2. Ordnung, bei denen der Funktionswert, die Ableitung oder eine Kombination am Rand vorgegeben ist; der Spezialfall mit nur einer Raumdimension ist dann das gewöhnliche Differentialgleichungs-Randwertproblem.

Bsp:

$$\left.\begin{aligned} &y'' + p(t)y' + q(t)y = f(t) \quad , \ t \in [a,b] \\ &a_0 y(a) + b_0 y'(a) = c_0 \\ &a_1 y(b) + b_1 y'(b) = c_1 \end{aligned}\right\} (*)$$

Ein Matrix-Verfahren ersetzt zu einem gegebenen Gitter $t_0 = a < t_1 < \dots < t_N = b$ alle Ableitungen und Funktionswerte von y in der Differentialgleichung und den Randbedingungen durch Finite-Differenzen-Approximationen mit Hilfe der Stützstellen $t_0, \dots, t_N$ und wertet die neuen Gleichungen an den Stützstellen aus. Dies gibt ein System von $N+1$ Gleichungen für die Unbekannten $y(t_0), \dots, y(t_N)$.

Bsp: Sei $h = \frac{b-a}{N}$, $t_k = a + kh$. Mit Taylorentwicklung erhält man

$$y''(t_k) = \frac{y(t_{k+1}) - 2y(t_k) + y(t_{k-1})}{h^2} + O(h^2)$$

$$y'(t_k) = \frac{y(t_{k+1}) - y(t_{k-1})}{2h} + O(h^2)$$

$$y'(t_0) = \frac{-3y(t_0) + 4y(t_1) - y(t_2)}{2h} + O(h^2)$$

$$y'(t_N) = \frac{y(t_{N-2}) - 4y(t_{N-1}) + 3y(t_N)}{2h} + O(h^2)$$

Das Matrix-Verfahren zu $(*)$ lautet also

$$\frac{y_{k+1} - 2y_k + y_{k-1}}{h^2} + p(t_k)\frac{y_{k+1} - y_{k-1}}{2h} + q(t_k)y_k = f(t_k) \quad , \ k = 1, \dots, N-1,$$

$$a_0 y_0 + b_0 \frac{-3y_0 + 4y_1 - y_2}{2h} = c_0 \ ,$$

$$a_1 y_N + b_1 \frac{y_{N-2} - 4y_{N-1} + 3y_N}{2h} = c_1 \ ,$$

zu lösen für $y_0, \dots, y_N$.

Für nichtlineare Probleme würde sich ein Satz nichtlinearer, zu lösender Gleichungen ergeben. Der Name „Matrix-Verfahren" stammt von der Matrix-Schreibweise des Verfahrens,

$$L_h Y_h = F_h \quad \text{für} \quad Y_h = \begin{pmatrix} y_0 \\ \vdots \\ y_N \end{pmatrix}, \ F_h = \begin{pmatrix} c_0 \\ f(t_1) \\ \vdots \\ f(t_{N-1}) \\ c_1 \end{pmatrix},$$

$$L_h = \frac{1}{h^2}\begin{pmatrix} 0 & 0 & & & \\ 1 & -2 & 1 & & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & 1 & -2 & 1 \\ & & & 0 & 0 \end{pmatrix} + \frac{1}{2h}\operatorname{diag}(b_0, p(t_1), \dots, p(t_N), b_1)\begin{pmatrix} -3 & 4 & -1 & & \\ 1 & 0 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & -1 \\ & & & 1 & -4 & 3 \end{pmatrix} + \operatorname{diag}(a_0, q(t_1), \dots, q(t_N), a_1)$$

L_h ist (nach einem Eliminationsschritt in der ersten und letzten Zeile) tridiagonal, kann also mit dem Thomas-Algorithmus (Gauß-Elimination ohne Pivotisierung) in $O(N)$ Schritten gelöst werden (falls L_h invertierbar).

Thm: Sei $q < 0$, $p = 0$, $b_0 = b_1 = 0$, $a_0 a_1 < 0$. Dann ist L_h invertierbar.

Bew: - L_h ist strikt diagonaldominant. □

Bem: Für andere Parameter muss man ggfs. vorsichtig sein - z.B. hatten wir gesehen, dass für $q > 0$ das Randwertproblem eventuell keine Lösung hat.

Ist das Verfahren konvergent, also $y_h(t) \xrightarrow{h\to 0} y(t)$ für die diskrete Lösung y_h mit Schrittweite h?

Sei $y_h = (y_0, \dots, y_N)$ und $I_h = (t_0, \dots, t_N)$. Es ist (in einer beliebigen Vektor- und induzierten Matrixnorm $\|\cdot\|$)

$$\|y(I_h) - y_h\| = \|L_h^{-1}(L_h y(I_h) - F_h)\| \le \|L_h^{-1}\| \, \|L_h y(I_h) - F_h\|.$$

Hier ist $L_h y(I_h) - F_h$ der Vektor an lokalen Diskretisierungsfehlern; sind seine Einträge $O(h^p)$, ist die Methode konsistent von Ordnung p (für obiges Beispiel sind sie $O(h^2)$). Die Methode ist stabil, wenn ein Datenfehler nicht mehr als unvermeidbar verstärkt wird.

Thm: (Stabilität) Sei $\tilde f: I=[a,b] \to \mathbb{R}$ mit $\|\tilde f - f\|_\infty \le \delta$, $\tilde c_0, \tilde c_a \in \mathbb{R}$ mit $|\tilde c_0 - c_0|, |\tilde c_a - c_a| \le \varepsilon$ eine Störung.
Für die numerische Lösung $\tilde y_h$ des gestörten Systems gilt

$$\|\tilde y_h - y_h\| \le \|L_h^{-1}\| \left\|\begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix}\right\| (\delta + \varepsilon).$$

Bem: · I. A. gilt ebenfalls $\|\tilde y(I_h) - y(I_h)\| \le C \left\|\begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix}\right\| (\delta+\varepsilon)$ für die Lösungen $y(t)$ und $\tilde y(t)$ des ursprünglichen und des gestörten Systems, sodass das Verfahren stabil (bzgl. Störungen in f, c_0, c_a) ist, wenn $h_0, C > 0$ existieren mit $\|L_h^{-1}\| \le C$ für alle $h \le h_0$.

· Auf ähnliche Weise kann/muss auch die Stabilität bzgl. der anderen Daten untersucht werden.

Bew: $\|\tilde y_h - y_h\| = \|L_h^{-1}(\tilde F_h - F_h)\| \le \|L_h^{-1}\| \, \|\tilde F_h - F_h\| \le \|L_h^{-1}\| \left\|\begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix}\right\| (\varepsilon + \delta)$ □

Wieder gilt „Stabilität + Konsistenz = Konvergenz".

Thm: (Konvergenz) Das Matrixverfahren sei konsistent (von Ordnung p) und stabil, d.h. es existieren $h_0, C > 0$ mit $\|L_h^{-1}\| \le C$ $\forall h \le h_0$. Dann ist es konvergent (von Ordnung p),

$$\|y(I_h) - y_h\| \Big/ \left\|\begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix}\right\| \xrightarrow{h\to 0} 0 \qquad \left(\text{bzw. } \|y(I_h) - y_h\| \Big/ \left\|\begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix}\right\| = O(h^p)\right).$$

Bew: $\|y(I_h) - y_h\| \le \|L_h^{-1}\| \, \|L_h y(I_h) - F_h\| \le C \, O(h^p) \left\|\begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix}\right\|$ □

Ist für unser Beispiel Stabilität erfüllt (zumindest in bestimmten Normen)?

Thm: (∞-Norm der Inversen) Sei $A \in \mathbb{R}^{n\times n}$ strikt diagonaldominant, d.h. $|A_{i,i}| > \sum_{j\ne i} |A_{i,j}|$. Dann gilt

$$\|A^{-1}\| \le \left[\min_i |A_{i,i}| - \sum_{j\ne i} |A_{i,j}|\right]^{-1}.$$

Bew: Für alle $x \in \mathbb{R}^n$ gilt $\|Ax\|_\infty = \max_i \left|\sum_{j=0}^{n} A_{i,j} x_j\right| \ge \left|\sum_{j=0}^{n} A_{i,j} x_j\right| \quad \forall i.$

Sei $\|x\|_\infty = |x_k|$, dann ist $\|Ax\|_\infty \ge \left|\sum_{j=0}^{n} A_{kj} x_j\right| \ge \left[|A_{kk}| - \sum_{j\ne k} |A_{kj}|\right] |x_k| \ge \min_i \left(|A_{ii}| - \sum_{j\ne i} |A_{ij}|\right) \|x\|.$

Für $y = Ax$ gilt somit $\|y\|_\infty \ge \min_i \left(|A_{ii}| - \sum_{j\ne i} |A_{ij}|\right) \|A^{-1} y\|$, und zu jedem $y \in \mathbb{R}^n$ existiert $x = A^{-1}y$. □

Thm: Sei $p = 0$, $b_0 = b_n = 0$, q stetig, $q < 0$, $a_0, a_n < 0$. Dann existiert $C > 0$, sodass $\|L_h^{-1}\|_\infty \le C$ für alle $h > 0$.

Bem: Somit ist die Diskretisierung unseres Beispiels konvergent von Ordnung 2 in der $\|\cdot\|_\infty$-Norm (also insbesondere punktweise).

Bew: Sei $c = \min_{t\in[a,b]} |q(t)| > 0$. L_h ist strikt diagonal-dominant, somit

$$\|L_h^{-1}\|_\infty \le \frac{1}{\min_i \left(|L_{h\,i,i}| - \sum_{j\neq i} |L_{h\,i,j}|\right)} = \frac{1}{\min\left(|a_0|, |a_1|, \min_{i=2,\dots,N-1} |q(t_i)|\right)} = \frac{1}{\min(|a_0|, |a_1|, c)} . \qquad \square$$

Kollokations- & Galerkinmethoden

Die Idee hier ist, y als Linearkombination $\hat{y}(t) = \sum_{i=0}^{N} \xi_i \cdot \psi_i(t)$ linear unabhängiger Funktionen $\psi_0, \dots, \psi_N : [a,b] \to \mathbb{R}$ zu approximieren. Beispielsweise können $\psi_0, \dots, \psi_N$ eine Basis von P_N sein.

Bei der Kollokationsmethode wählt man Kollokationspunkte $t_0 = a < t_1 < \dots < t_N = b$ und fordert die Erfüllung der Differentialgleichung an diesen Punkten. Dies führt auf ein Gleichungssystem, zu lösen in den unbekannten Koeffizienten $\xi_0, \dots, \xi_N$.

Bsp: Für obiges Beispiel lautet die Kollokationsmethode $\hat{y}''(t_k) + p(t_k)\hat{y}'(t_k) + q(t_k)\hat{y}(t_k) = f(t_k)$, $k = 1, \dots, N-1$,

$$a_0 \hat{y}(a) + b_0 \hat{y}'(a) = c_0 ,$$

$$a_1 \hat{y}(b) + b_1 \hat{y}'(b) = c_1 .$$

Dies ist ein System aus $N+1$ linearen Gleichungen in $\xi_0, \dots, \xi_N$.

Spektralmethoden sind ein Spezialfall hiervon: Hier bilden $\psi_0, \dots, \psi_N$ eine Basis von P_N, und als Kollokationspunkte werden Čebyšev-Punkte $\left(t_j = -\cos\frac{j\pi}{N}, j = 0, \dots, N, \text{ auf } [-1,1]\right)$ genutzt. Wir wissen bereits, dass diese Stützstellen für Polynom-Interpolation anhand der Funktionswerte oder Ableitungen erheblich bessere Konvergenz als äquidistante Stützstellen liefern; Spektralmethoden finden im Grunde Polynome, die eine Kombination aus Funktionswerten und Ableitungen interpolieren. Dabei arbeitet man typischerweise mit den Funktionswerten $y_k = \hat{y}(t_k)$, $k = 0, \dots, N$.

Def: (Čebyšev-Differentiationsmatrix) Seien Čebyšev-Stützstellen $t_0, \dots, t_N$ gegeben. Die Abbildung, die gegebenen Stützwerten $(y_0, \dots, y_N)$ die Ableitung des eindeutigen Interpolationspolynoms $p \in P_N$ an $t_0, \dots, t_N$ zuordnet, heißt Čebyšev-Differentiationsmatrix,

$$D_N \in \mathbb{R}^{(N+1)\times(N+1)}, \quad \begin{pmatrix} y_0 \\ \vdots \\ y_N \end{pmatrix} \mapsto D_N \begin{pmatrix} y_0 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} p'(t_0) \\ \vdots \\ p'(t_N) \end{pmatrix}.$$

Bsp: $N = 1$, $t_0 = -1$, $t_1 = 1$: Interpolationspolynom $p(t) = \frac{1}{2}(1-t)y_0 + \frac{1}{2}(1+t)y_1$

$p'(t) = \frac{y_1 - y_0}{2} \Rightarrow D_1 = \frac{1}{2}\begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix}$ > Finite Differenzen-Formel

- $N = 2$, $(t_0, t_1, t_2) = (-1, 0, 1)$: $p(t) = \frac{1}{2}t(1-t)y_0 + (1+t)(1-t)y_1 + \frac{1}{2}t(t+1)y_2$

$p'(t) = (t - \frac{1}{2})y_0 - 2t\,y_1 + (t + \frac{1}{2})y_2 \Rightarrow D_2 = \frac{1}{2}\begin{pmatrix} -3 & 4 & -1 \\ -1 & 0 & 1 \\ 1 & -4 & 3 \end{pmatrix}$ > Rückwärtsdifferentiationsformel

Thm: (Čebyšev-Differentiationsmatrix) $D_N = (D_N)_{i,j=0,\dots,N} \in \mathbb{R}^{(N+1)\times(N+1)}$ ist gegeben durch

Zeile \ Spalte	0	$j=1,\dots,N-1$	N
0	$\frac{2N^2+1}{6}$	$2\frac{(-1)^j}{1-t_j}$	$\frac{1}{2}(-1)^N$
$i=1,\dots,N-1$	$-\frac{1}{2}\frac{(-1)^i}{1-t_i}$	$\frac{(-1)^{i+j}}{t_i-t_j}$ (Diagonale: $\frac{-t_j}{2(1-t_j^2)}$)	$\frac{1}{2}\frac{(-1)^{N+i}}{1+t_i}$
N	$-\frac{1}{2}(-1)^N$	$-2\frac{(-1)^{N+j}}{1+t_j}$	$-\frac{2N^2+1}{6}$

Diagonale

Bew: Das j-te Lagrangepolynom ist $p_j(t) = \frac{1}{w_j}\prod_{\substack{k=0\\k\neq j}}^{N}(t-t_k)$ mit $w_j = \prod_{\substack{k=0\\k\neq j}}^{N}(t_j-t_k)$.

$$p_j'(t)/p_j(t) = (\log p_j)'(t) = \sum_{\substack{k=0\\k\neq j}}^{N}\frac{1}{t-t_k} \quad\Rightarrow\quad p_j'(t) = \sum_{\substack{k=0\\k\neq j}}^{N}\frac{p_j(t)}{t-t_k} = \sum_{\substack{k=0\\k\neq j}}^{N}\frac{1}{w_j}\prod_{\substack{\ell=0\\\ell\neq j,k}}^{N}(t-t_\ell)$$

$$\Rightarrow (D_N)_{ij} = p_j'(t_i) = \begin{cases}\sum_{\substack{k=0\\k\neq j}}^{N}\frac{1}{t_i-t_k} & , i=j\\ \frac{1}{w_j}\prod_{\substack{\ell=0\\\ell\neq i,j}}^{N}(t_i-t_\ell) = \frac{w_i}{w_j(t_i-t_j)} & , i\neq j\end{cases}$$

Die Werte erhält man nun durch Einsetzen der Čebyšev-Knoten. □

Bsp: Für obiges Beispiel ergibt sich das für $y_0,\dots,y_N$ zu lösende System

$$\begin{bmatrix}(a_0,0,\dots,\rho) + b_0(D_N)_{0,:}\\ \left[D_N^2 + \mathrm{diag}(p(t_0),\dots,p(t_N))D_N + \mathrm{diag}(q(t_0),\dots,q(t_N))\right]_{1:N-1,:}\\ (a_N,0,\dots,0) + b_N(D_N)_{N,:}\end{bmatrix}\begin{bmatrix}y_0\\\vdots\\y_N\end{bmatrix} = \begin{bmatrix}c_0\\f(t_1)\\\vdots\\f(t_{N-1})\\c_N\end{bmatrix}$$

Bem: Im Allgemeinen ist das für Kollokationsmethoden zu lösende Gleichungssystem voll besetzt.

- Das Gleichungssystem ist (anders als bei Galerkin-Methoden oder Finite Differenzen) typischerweise auch nicht symmetrisch, selbst wenn der zugehörige Differentialoperator selbstadjungiert ist (z.B. für $p \equiv 0$).

Galerkinmethoden suchen im Gegensatz zu Kollokationsmethoden nicht eine die Differentialgleichung an einzelnen Punkten erfüllende Funktion, sondern eine die Differentialgleichung schwach erfüllende Funktion. Hierzu wird die Differentialgleichung mit einer glatten Funktion $\varphi \in C^\infty(I)$ multipliziert und dann partiell integriert.

Bsp: Im obigen Beispiel ergibt sich

$$\int_I y''(t)\varphi(t) + p(t)y'(t)\varphi(t) + q(t)y(t)\varphi(t)\,dt = \int_I f(t)\varphi(t)\,dt \quad \forall \varphi \in C^\infty(I)$$

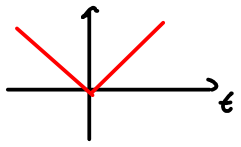
$$\Rightarrow \int_I -y'(t)\varphi'(t) + p(t)y'(t)\varphi(t) + q(t)y(t)\varphi(t)\,dt + \Big[y'(t)\varphi(t)\Big]_{t=a}^{b} = \int_I f(t)\varphi(t)\,dt \quad \forall \varphi \in C^\infty(I)$$

Der Einfachheit halber sei im Folgenden $b_0 = b_1 = c_0 = c_1 = 0$. Um eine schwache Lösung zu definieren, müssen wir erst die nötigen Funktionenräume definieren.

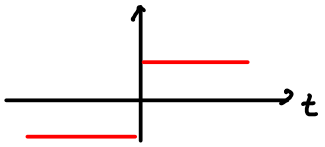
Def: (Hilbertraum H^1) $\cdot\ L^2(I) = \{ y: I \to \mathbb{R} \mid y \text{ ist messbar}, \int_I y^2\,dx < \infty \}$

- $y: I \to \mathbb{R}$ hat eine <u>schwache Ableitung</u> y', wenn y und y' messbar sind und
  $\int_I y'(t)\varphi(t)\,dt = -\int_I y(t)\varphi'(t)\,dt \quad \forall \varphi \in C_0^\infty(I)$
- $H^1(I) = \{ y \in L^2(I) \mid y \text{ hat eine schwache Ableitung } y' \in L^2(I) \}$.
- $H_0^1(I) = \{ y \in H^1(I) \mid y|_{\partial I} = 0 \}$

Bsp: $\cdot\ y: (-1,1) \to \mathbb{R},\ t \mapsto |t|$ liegt in $H^1((-1,1))$ mit schwacher Ableitung $y'(t) = \operatorname{sign}(t)$



$\cdot\ y: (-1,1) \to \mathbb{R},\ t \mapsto \operatorname{sign}(t)$ liegt nicht in $H^1((-1,1))$, da es keine schwache Ableitung besitzt.



Thm: (Hilbertraum H^1) $H^1(I)$ bildet mit dem Skalarprodukt $(u,v)_{H^1} = \int_I u'v' + uv\,dt$ einen Hilbertraum, $H_0^1(I)$ bildet mit $(u,v)_{H_0^1} = \int_I u'v'\,dt$ einen Hilbertraum.

Def: (Schwache Lösung) Eine Funktion $y \in H_0^1(I)$ heißt schwache Lösung von $(*)$ mit $b_0 = b_1 = c_0 = c_1 = 0$, $p, q \in C(I)$, $f \in L^2(I)$, wenn

$$\int_I -y'(t)\varphi'(t) + p(t)y'(t)\varphi(t) + q(t)y(t)\varphi(t)\,dt = \int_I f(t)\varphi(t)\,dt \quad \forall \varphi \in H_0^1(I)$$

Sei $V \subset H_0^1(I)$ ein endlichdimensionaler Teilraum mit Basis $\psi_0, \dots, \psi_N$. Die Galerkin-Approximation der Lösung von $(*)$ ist diejenige Funktion $y \in V$, für die gilt

$$\int_I -y'(t)\varphi'(t) + p(t)y'(t)\varphi(t) + q(t)y(t)\varphi(t)\,dt = \int_I f(t)\varphi(t)\,dt \quad \forall \varphi \in V.$$

Mit $y = \sum_{i=0}^{N} \xi_i \psi_i$ führt dies auf das lineare Gleichungssystem

$$\left(\int_I -\psi_i'\psi_j' + p\psi_i'\psi_j + q\psi_i\psi_j\,dt \right)_{ji} \begin{pmatrix} \xi_0 \\ \vdots \\ \xi_N \end{pmatrix} = \left(\int_I f\psi_j\,dt \right)_j .$$