1 Newton Polynomische Interpolation

1.1 Collocation

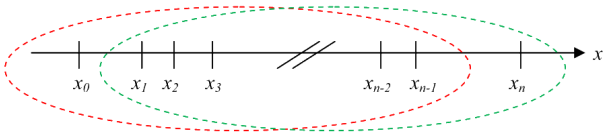
Collocation = Alle Messpunkte werden von angenäherter Funktion (z.B. Polynom) getroffen. Das Polynom

$$y(x) = p(x) = c_0 + c_1x^1 + c_2x^2 + \ldots + c_mx^m \qquad \text{mit} \quad y(x_k) = p(x_k) = y_k \quad (k = 0, 1, \ldots, n)$$

ergibt ein lineares Gleichungssystem von $n + 1$ Gleichungen.

1.2 Aitken-Neville Rekursionsformel

Sich an den Messpunkten $x_1, \ldots, x_{n-1}$ überlappende polynomiale Interpolationsfunktionen $f_1 = p_{0,1,\ldots,n-1}$, $f_2 = p_{1,\ldots,n}$ können zusammengefasst werden.



$$p(x) = p_{0,1,\ldots,n}(x) = \frac{(x - x_0) \cdot \overbrace{p_{1,2,\ldots,n}}^{\text{green}}(x) - (x - x_n) \cdot \overbrace{p_{0,1,\ldots,n-1}}^{\text{red}}(x)}{(x_n - x_0)}$$

1.3 Newton Polynome

Eigenschaften

+ Eingebettet (Wenn ein Messwert dazukommt muss nicht alles neu berechnet werden)
+ Eine einzige Formel
+ Praktisch: Messungen sind gleichverteilt.
– Runge-Phänomen (Schwingungen am Rand)
– Aufwändig zum rechnen

$$\begin{aligned}
y_0 &= a_0 \cdot \pi_0 \\
&\downarrow \\
y_1 &= a_0 \cdot \pi_0 + a_1 \cdot \pi_1 \\
&\downarrow \quad \downarrow \\
y_2 &= a_0 \cdot \pi_0 + a_1 \cdot \pi_1 + a_2 \cdot \pi_2 \\
&\vdots \ \downarrow \quad \searrow \quad \searrow \\
y_n &= a_0 \cdot \pi_0(x_n) + a_1 \cdot \pi_1(x_n) + a_2 \cdot \pi_2(x_n) + \ldots + a_n\pi_n(x_n)
\end{aligned}$$

$$\begin{aligned}
\pi_0 &= 1 \\
\pi_1 &= (x - x_0) \\
\pi_2 &= (x - x_0)(x - x_1) \\
&\vdots \\
\pi_n &= (x - x_0)(x - x_1)\ldots(x - x_{n-1}) \\
\pi_{n+1} &= (x - x_0)(x - x_1)\cdots(x - x_{n-1})(x - x_n)
\end{aligned}$$

Annäherung an "Messreihe" mit Polynomen:

$$y(x) \approx p(x) = a_0\pi_0(x) + a_1\pi_1(x) + \ldots + a_n\pi_n(x)$$

1.4 Dividierte Differenzen

Bemerkung:

$a_{p-1} = y(x_0, x_1, x_2, ..., x_p)$ gehört zu $\pi_{p-1} = (x - x_0) \cdot (x - x_1) \cdot ... \cdot (x - x_{p-2})$

Elegantere und schnellere Auflösung der Newton Polynome.

$$y(x_\mathbf{0}, x_1, \ldots x_\mathbf{k}) = \frac{y(x_\mathbf{1}, x_2, \ldots x_\mathbf{k}) - y(x_\mathbf{0}, x_1, \ldots x_\mathbf{k-1})}{(x_k - x_0)} \quad (k = 0, 1, \ldots, n)$$

$$a(x_\mathbf{0}, \ldots, x_\mathbf{n}) = \frac{a(x_\mathbf{1}, \ldots, x_\mathbf{n}) - a(x_\mathbf{0}, \ldots, x_\mathbf{n-1})}{x_n - x_0}$$

$k = 0$ $y(x_0)$

$k = 1$ $\frac{y(x_1) - y(x_0)}{(x_1 - x_0)}$

$k = 2$ $\frac{y(x_1, x_2) - y(x_0, x_1)}{(x_2 - x_0)}$

$k = 3$ $\frac{y(x_1, x_2, x_3) - y(x_0, x_1, x_2)}{(x_3 - x_0)}$

BY-NC-SA

Symmetrie: $y(x_0, x_1) = y(x_1, x_0) \Longrightarrow y(x_0, x_1, ..., x_n)$ ist unabhängig von der Reihenfolge der x-Werte! (Die letze Differenz a_n stimmt überein, die anderen Differenzen weichen ab.)

Beispiel:

	x_k	y_k				
x_0	0	(1) a_0				
			(0) a_1			
x_1	1	1		($\frac{1}{2}$) a_2		
			1		($-\frac{1}{12}$) a_3	
x_2	2	2		$\frac{1}{6}$		
			$\frac{3}{2}$			
x_3	4	5				

$$y(x) \approx p(x) = a_0 \cdot \pi_0(x) + a_1 \cdot \pi_1(x) + a_2 \cdot \pi_2(x) + a_3\pi_n(x)$$
$$= 1 \cdot \pi_0(x) \ + 0 \cdot \pi_1(x) \ + \frac{1}{2} \cdot \pi_2(x) \ - \frac{1}{12}\pi_n(x)$$
$$y(x) \approx p(x) = 1 + \frac{1}{2}(x - x_0)(x - x_1) - \frac{1}{12}(x - x_0)(x - x_1)(x - x_2)$$
$$= 1 + \frac{1}{2}(x - 0)(x - 1) \quad - \frac{1}{12}(x - 0)(x - 1)(x - 2)$$

1.5 Collocation Fehlerformel

$$y(x) - p(x) = \frac{y^{(n+1)}(\xi)}{(n+1)!}(x - x_0)(x - x_1)\dots(x - x_{n-1})(x - x_n) = \frac{y^{(n+1)}(\xi)}{(n+1)!}\pi_{n+1}(x) \quad \xi \in (\min(x_i), \max(x_i))$$

Herleitung: $y = p + \frac{y^{(n+1)}(\xi)}{(n+1)!} \cdot \pi_{n+1} = y(x_0, x_1, ..., x_n) + \frac{y^{(n+1)}(\xi)}{(n+1)!} \cdot \pi_{n+1} \Longrightarrow y(x_0, x_1, ..., x_{n+1}) = \frac{y^{(n+1)}(\xi)}{(n+1)!}$

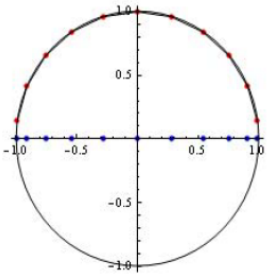
Der Fehlerterm kann als Ableitung interpretiert werden. Es ist aber nicht klar wo die Ableitung im Bereich $[x_0, x_n]$ ausgewertet wird, da ξ nicht bekannt ist!

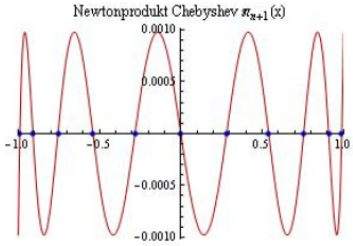
1.6 Runge Phänomen und Chebyshev Argumente

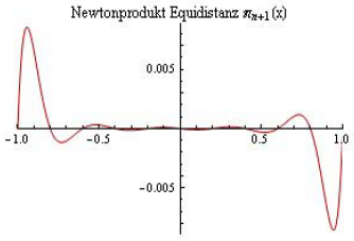
Runge Phänomen: Potenzielle Oszillationen an den Definitionsrändern von Polynom-Collocations. Verursacht durch hohe n und hohe $\frac{y^{(n+1)}(x)}{(n+1)!}$.
Chebyshev-Argumente: Anpassung der Messpunkte durch andere Verteilung (nicht mehr gleichverteilte x_k, sondern am Einheitskreis gleichverteilte)

$$x_k = \cos(\frac{2k+1}{2(n+1)}\pi), \quad (k = 0, 1, \dots, n))$$

ergibt maximale Amplitude $\max_{-1 \le x \le 1} |\pi_{n+1}(x)| = \frac{1}{2^n}$. Mit affiner Transformation von Einheitsintervall $[-1, 1]$ zu $[a, b]$: $x \to a + \frac{b-a}{2}(x + 1)$



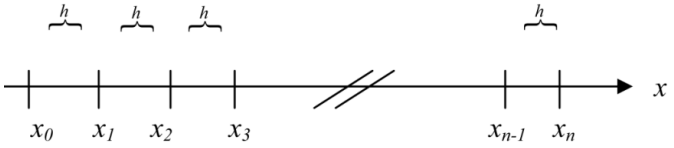




Eigenschaften
- \+ Kein Runge-Phänomen (Schwingungen am Rand)
- \+ Eine einzige Formel
- – „Nicht eingebettet“ (neue Messungen bedeutet, neue Polynomberechnung!)
- – Praktisch: Messungen sind oft nur gleichverteilt möglich (Randauflösung beschränkt)
- – Der Bereich der Interpolation ist fixiert (Extrapolation schlecht! (ausserhalb vom Rand))

BY-NC-SA

1.7 Gleichverteilte Argumente



$$x_k = x_0 + kh \qquad k = 0, 1, 2, \ldots n$$

$$\boxed{y(x_0, x_1, \ldots, x_k) = \frac{\Delta^k y_0}{h^k k!}}$$

$$p(x) = y_0 + \frac{\Delta^1 y_0}{h^1 1!}\pi_1(x) + \frac{\Delta^2 y_0}{h^2 2!}\pi_2(x) + \ldots + \frac{\Delta^n y_0}{h^n n!}\pi_n(x) = \sum_{k=0}^{n} \frac{\Delta^k y_0}{h^k k!}\pi_k(x)$$

Δ^0	y_0	y_1	y_2	y_3	y_4
Δ^1	$y_1 - y_0$ Δy_0	$y_2 - y_1$ Δy_1	$y_3 - y_2$ Δy_2	$y_4 - y_3$ Δy_3	
Δ^2	$\Delta y_1 - \Delta y_0$ $\Delta^2 y_0$	$\Delta y_2 - \Delta y_1$ $\Delta^2 y_1$	$\Delta y_3 - \Delta y_2$ $\Delta^2 y_2$		

1.8 Taylor Reihe

$$f(x) = \underbrace{y_0 + \frac{y'(x_0)}{1!}(x - x_0) + \frac{y''(x_0)}{2!}(x - x_0)^2 + \ldots + \frac{y^{(n)}(x_0)}{n!}(x - x_0)^n}_{\text{Taylor Polynom p(x)}} + \underbrace{\frac{y^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1}}_{\text{Lagrange Fehlerterm}}$$

2 Hermite Interpolation (Osculation)

Erweiterung der dividierten Differenzen: Es sind jetzt auch Ableitungen an Messpunkten als Bedingungen möglich.
Aus der Definition von $y(x)$ ist $yx_0, x_0) = y'(x_0)$. Verallgemeinert ergibt das

$$y(\underbrace{x_0, \ldots x_n, x_{n+1}}_{(n+2)}) = \frac{y^{(n+1)}(\xi)}{(n+1)!}$$

$$y(\underbrace{x_0, \ldots x_0}_{(n+1)}) = \lim_{\xi \to x_0} \frac{y^{(n)}(\xi)}{(n)!} = \frac{y^{(n)}(x_0)}{n!}$$

Für die Hermite Interpolation werden dieselben Newton Tableaus verwendet, aber mit Wiederholungen (siehe Beispiel). Die rot markierten Werte werden wie üblich berechnet.
Wichtig ist, dass bei der Berechnung der dividierten Differenzen keine Löcher entstehen (Ableitungen müssen Lückenlos vorhanden sein (z.B.: y', y'', y''' good; y', y''' bad))! Dann ist das Gleichungssystem nicht lösbar! Es kann bei unbestimmten Resultaten eine Variable eingesetzt werden, welche am Schluss ermittelt werden kann.

x	y		
$x_0 = 2$	$y(x_0) = 1$		
		$\frac{y^{(1)}(x_0)}{1!} = 1$	
$x_0 = 2$	$y(x_0) = 1$		$\frac{y^{(2)}(x_0)}{2!} = 0$
		$\frac{y^{(1)}(x_0)}{1!} = 1$	
$x_0 = 2$	$y(x_0) = 1$		
$x_1 = 4$	$y(x_1) = 2$		
		$\frac{y^{(1)}(x_1)}{1!} = 0$	
$x_1 = 4$	$y(x_1) = 2$		$\frac{y^{(2)}(x_1)}{2!} = 0$
		$\frac{y^{(1)}(x_1)}{1!} = 0$	
$x_1 = 4$	$y(x_1) = 2$		

x	y				
2	(1) a_0				
		(1) a_1			
2	1		(0) a_2		
		1		$(-\frac{1}{8})$ a_3	
2	1		$-\frac{1}{4}$		$(\frac{1}{16})$ a_4
	$\frac{1}{2}$		0		(0) a_5
4	2		$-\frac{1}{4}$		$\frac{1}{16}$
		0		$\frac{1}{8}$	
4	2		0		
		0			
4	2				

BY-NC-SA

Es werden die **modifizierten** Newton Polynome verwendet. In diesem Fall:

$\pi_0 = 1$	$\pi_4 = (x-x_0)(x-x_0)(x-x_0)(x-x_1)$
$\pi_1 = (x-x_0)$	$\pi_5 = (x-x_0)(x-x_0)(x-x_0)(x-x_1)(x-x_1)$
$\pi_2 = (x-x_0)(x-x_0)$	$\pi_6 = (x-x_0)(x-x_0)(x-x_0)(x-x_1)(x-x_1)(x-x_1)$
$\pi_3 = (x-x_0)(x-x_0)(x-x_0)$	

$$
\begin{aligned}
p_2(x) &= a_0 \cdot \pi_0 + a_1 \cdot \pi_1 + a_2 \cdot \pi_2 + a_3 \cdot \pi_3 + a_4 \cdot \pi_4 + a_5 \cdot \pi_5 \\
&= a_0 \cdot 1 + a_1 \cdot (x-x_0) + a_2 \cdot (x-x_0)^2 + a_3 \cdot (x-x_0)^3 + a_4 \cdot (x-x_0)^3(x-x_1) + a_5 \cdot (x-x_0)^3(x-x_1)^2 \\
&= 1 + (x-2) - \frac{1}{8}(x-2)^3 + \frac{1}{16}(x-2)^3(x-4)
\end{aligned}
$$

2.3 Fehlerformel

$$y(x) - p(x) = \frac{y^{(d)}(\xi)}{d!}(x-x_0)^{d_0}(x-x_1)^{d_1}\cdots(x-x_n)^{d_n} \qquad x, \xi \in \{\min x_i, \max x_i\} \quad \text{für} \quad i = 0, 1, \ldots, 2$$

d ist die total Anzahl Bedingungen und d_i die Anzahl Bedingungen pro Stützstelle x_i. ($d = \sum_{i=0}^{n} d_i$)

2.4 Fehlende Ableitungen

Bei fehlenden Ableitungsvorgaben werden Variablen eingesetzt und das Tableau mit diesen durchgerechnet. Am Schluss wird von hinten her auf die fehlenden Variablen geschlossen.

Beispiel:
Gesucht: Polynom 2. Ordnung, welches durch die Punkte $y(2) = 1$ und $y(4) = 1$ geht und die Ableitung $y'(4) = 1$ aufweist.

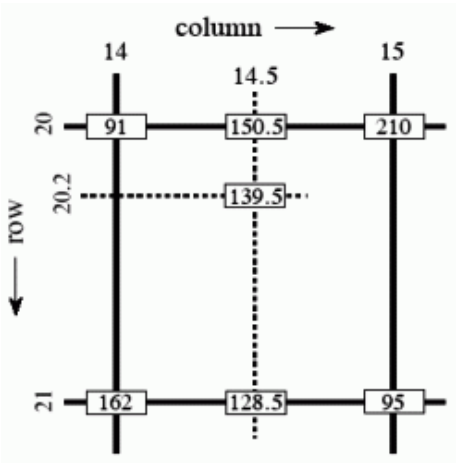
x	y			
2	$\textcircled{1}\,a_0$			
		$\textcircled{\beta}\,a_1$		
2	1		$\textcircled{-\frac{\beta}{2}}\,a_2$	
		0		$\textcircled{\frac{1+\beta}{4}}\,a_3 \overset{!}{=} 0$
4	1		$\frac{1}{2}$	
		1		
4	1			

- Weil das Polynom die Ordnung $d = 2$ aufweisen muss, gilt: $\frac{1+\beta}{4} = 0$
  Daraus folgt: $\beta = -1$
- Das Lösungspolynom ist:

$$
\begin{aligned}
p(x) &= a_0 + a_1(x-x_0) + a_2(x-x_0)^2 + a_3(x-x_0)^2(x-x_1) \\
&= 1 + \beta(x-x_0) - \frac{\beta}{2}(x-x_0)^2 + \frac{1+\beta}{4}(x-x_0)^2(x-x_1)\Big|_{\beta=-1} \\
&= 1 - (x-2) + \frac{1}{2}(x-2)^2
\end{aligned}
$$

BY-NC-SA

3 Multi-variate Polynomial Interpolation



1. Die Basis der Polynome kann durch das *2-fold tensor product* berechnet werden ⇒ Bilineare Interpolation
   $\{1, x, x^2\} \otimes \{1, y, y^2\} = \{1, x, y, x^2, y^2, xy, x^2y, xy^2, x^2y^2\}$;
   Dadurch kann man die Anzahl Gleichungen n berechnet werden:
   $\mathbb{R}^n \otimes \mathbb{R}^k = \mathbb{R}^{nk}$
2. Interpolation der gewünschten Ordnung (linear, quadratisch, ...) in X-Richtung an Stelle y_0 berechnen: $p(x, y_0)$
3. Interpolation der gewünschten Ordnung (linear, quadratisch, ...) in X-Richtung an Stelle y_1 berechnen: $p(x, y_1)$
4. Tableau für dividierte Differenzen berechnen (hier linearer Fall):

y	z	
y_0	$p(x, y_0) = a_{y0}$	
y_1	$p(x, y_1)$	$\frac{p(x,y_1)-p(x,y_0)}{y_1-y_0} = a_{y1}$

5. Polynom aufstellen:
   $p(x, y) = a_{y0}\pi_0(y) + a_{y1}\pi_1(y) + a_{y2}\pi_2(y) + \ldots$
6. Polynom ausmultiplizieren:
   $p(x, y) = a_{0,0}\pi_0(x)\pi_0(y) + a_{1,0}\pi_1(x)\pi_0(y) + a_{0,1}\pi_0(x)\pi_1(y) + a_{1,1}\pi_1(x)\pi_1(y) + \ldots$

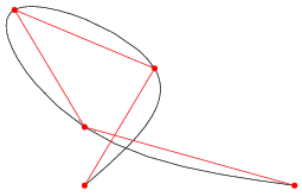
3.1 Beispiel:

Die Punkte $(14, 20), (14, 21), (15, 20)(15, 21)$ sind gegeben: $\{91/255, 162/255, 210/255, 95/255\}$. Gesucht ist eine lineare interpolation. Das ergibt folgende Polynom Basen: $(\{1, x\} \times \{1, y\} = \{1, x, y, xy\})$

1. Interpolation für $y = y_0 = 20$: $p(x, y_0) = p(x_0, y_0)\pi_0(x) + \frac{p(x_1,y_0)-p(x_0,y_0)}{x_1-x_0}\pi_1(x) = \frac{1}{255}\left(91 \cdot 1 + \frac{210-91}{15-14} \cdot (x - 14)\right)$
2. Das selbe für $y = y_1 = 21$: $p(x, y_1) = \frac{1}{255}\left(162 \cdot 1 + \frac{95-162}{15-14} \cdot (x - 14)\right)$
3. Dann die schlussendliche Interpolation: $p(x, y) = p(x, y_0)\pi_0(y) + \frac{p(x,y_1)-p(x,y_0)}{y_1-y_0}\pi_1(y)$
   $= \frac{1}{255}\left((91 \cdot 1 + \frac{210-91}{15-14} \cdot 1) \cdot 1 + \frac{162 \cdot 1 \frac{95-162}{15-14} \cdot (x-14) - 91 \cdot 1 + \frac{210-91}{15-14} \cdot (x-14)}{21-20} \cdot (y - 20)\right) = -215.98 + 15.0549x + 10.4902y - 0.7294xy$

BY-NC-SA

4 Spline-Interpolation



Die Idee der Spline-Interpolation ist, die Daten nicht mit einem Polynom hohen Grades zu interpolieren, sondern mit mehreren Polynomen tiefen Grades. Daduch kann die Tendenz zum Schwingen von Polynomen hohen Grades umgangen werden. Bei den "Bruchstellen", den Übergängen von einem Patch zum nächsten, müssen die Ableitungen der benachbarten Patches bis zu einem vorgegeben Ableitung übereinstimmten.

Eigenschaften

+ Kein Runge-Phänomen (Schwingungen am Rand)
+ Polynome haben tiefen Grad
+ Aufwand zur Berechnung geringer als Newton [Spline: $O(n)$ (wegen Tridiagonaler Band-Matrix), Newton: $O(n^2)$]
– „Nicht eingebettet" (neue Messungen bedeutet, neue Polynomberechnung!)
– Polynome müssen zur Auswertung zusammengesetzt werden (Post-Processing Aufwand gross)

4.1 Ein-Dimensionale Splines

4.1.1 Prinzip

Pro *Patch* (von total n) wird ein Polynom vom Grad d (meist kubisch, $d = 3$) berechnet. An den Übergängen bei x_i können Bedingungen C^k $(k = 1, \ldots, d-1)$ definiert werden.

Freiheitsgrade n Patches mit je 1 Polynom mit je $(d+1)$ Koeffizienten, gibt total $n(d+1)$ Freiheitsgrade.

Bedingungen n Patches mit Anfang und Ende $= 2n$; $d-1$ Ableitungen pro innerem Punkt $(n-1)(d-1)$ ergibt total $((n(d+1)-(d-1)))$ Bedingungen.

Zusatzbedingungen Vergleicht man Freiheitsgrade und Bedinungen, sind $d-1$ Bedingungen zu wählen. Beim *natural spline* werden dazu die zweiten Ableitungen auf 0 gesetzt; beim *clamped splines* werden die 1. Ableitungen vorgegeben.

4.1.2 Allgemeines Vorgehen (kubische Splines)

Die endgültige Interpolation hat folgende Form (Beschreibung **eines** Patches):

$$\boxed{S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3}$$

$$S_i' = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2$$
$$S_i'' = 2c_i + 6d_i(x - x_i)$$

Gebraucht wird auch (Patch-Breite) $h_i = x_{i+1} - x_i = \Delta x_i$

1. $a_i = y_i \qquad (i = 0, \ldots n-1)$
2. Für kubische Splines hat es $d - 1 = 3 - 1 = 2$ Zusatzbedingungen.

$$\begin{pmatrix} h_0 & 2(h_0+h_1) & h_1 & & & \\ & h_1 & 2(h_1+h_2) & h_2 & & \\ & & h_2 & 2(h_2+h_3) & h_3 & \\ & & & \ddots & \ddots & \ddots \\ & & & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} \\ \\ 3\left(\frac{y_{i+1}-y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right) \\ \\ \\ \end{pmatrix}_{i=1,\ldots,n-2}$$

BY-NC-SA

- **Natural Splining** (Randkrümmung): $y_0'' = 0 = y_n''$ (minimiert das Energiemass! ($\min \int |f''(x)|^2 \mathrm{d}x$))
  (a) $c_0 = c_n = 0$
  (b) Gleichungssystem nach c_i auflösen:

$$\begin{pmatrix} 2(h_0+h_1) & h_1 & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & h_2 & 2(h_2+h_3) & h_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} \\ & & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} 3\left(\frac{y_{i+1}-y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right) \end{pmatrix}_{i=1,\dots,n-1}$$

- **Clamped Splining:**
  (a) $b_0 = y_0'; b_n = y_n'$
  (b) Gleichungssystem nach c_i auflösen:

$$\begin{pmatrix} 2h_0 & h_0 & & & & \\ h_0 & 2(h_0+h_1) & h_1 & & & \\ & h_1 & 2(h_1+h_2) & h_2 & & \\ & & h_2 & 2(h_2+h_3) & h_3 & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} \\ & & & & 2h_{n-2} & 4h_{n-2}+3h_{n-1} \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} 3\left(\frac{a_1-a_0}{h_0} - y_0'\right) \\ \vdots \\ \left(3\left(\frac{y_{i+1}-y_i}{h_i} - \frac{y_i-y_{i-1}}{h_{i-1}}\right)\right)_{i=1,\dots,n-2} \\ \vdots \\ 9\frac{y_n - a_{n-1}}{h_{n-1}} - 6\frac{a_{n-1}-a_{n-2}}{h_{n-2}} - 3y_n' \end{pmatrix}$$

3. $b_{i-1} = \frac{a_i - a_{i-1}}{h_{i-1}} - \frac{2c_{i-1}+c_i}{3}h_{i-1} \qquad (i = 1, \dots n-1)$
4. $b_{n-1} = \frac{y_n - a_{n-1}}{h_{n-1}} - c_{n-1}h_{n-1} - d_{n-1}h_{n-1}^2 = \frac{y_n - y_{n-1}}{h_{n-1}} - \frac{2}{3}c_{n-1}h_{n-1}$
5. $d_{i-1} = \frac{c_i - c_{i-1}}{3h_{i-1}} \qquad (i = 1, \dots n-1)$
6. Clamped splining: $d_{n-1} = \frac{y_n' - b_{n-1} - 2c_{n-1}h_{n-1}}{3h_{n-1}^2}$ Natural Splining: $d_{n-1} = -\frac{c_{n-1}}{3h_{n-1}}$

Weitere Methoden
Bei *Hermite cubic splines* werden C^1-Funktionen aus gepatchten kubischen Polynomen mit vorgegebenen ersten Ableitungen gebildet.
Periodic splines sind C^2 Funktionen aus gepatchten kubischen Polynomen mit Periodizität ($S'(x_0) = S'(x_n)$ und $S''(x_0) = S''(x_n)$).

4.1.3 Fehlerabschätzung

Für kubische Splines mit C^2 gilt folgende Fehlerabschätzung ($H = \max h_i$ $(i = 0, \dots n-1)$):
$|y(x) - S(x)| \leq \max|y^{(4)}(x)| \frac{5}{384} H^4 \qquad |y'(x) - S'(x)| \leq \max|y^{(4)}(x)| \frac{1}{24} H^3 \qquad |y''(x) - S''(x)| \leq \max|y^{(4)}(x)| \frac{3}{8} H^2$

BY-NC-SA

4.2 Bernstein-Bézier Splines (B-B Splines)

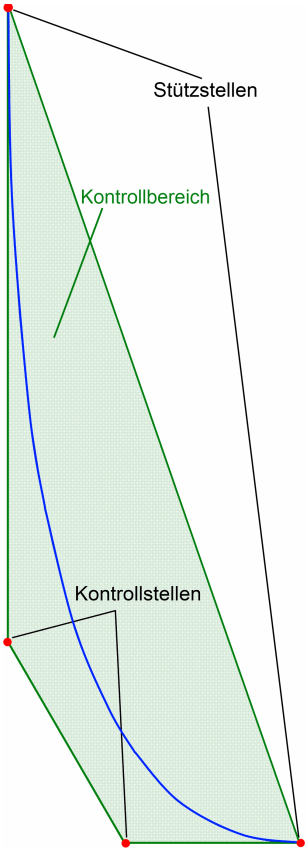
4.2.1 Bernstein-Polynome

Die Bernstein-Polynome sind im Intervall $t \in [0,1]$ definiert als

$$B_{i,n}(t) = \binom{n}{i}(1-t)^{n-i}t^i \qquad t \in [0,1]\ i = 0,1,\ldots,n$$

mit i: Nr und n: Ordnung des Splines.
Mittels einer affinen Transformation in den Bereich $t \in [a,b]$ folgt

$$B_{i,n}(u,a,b) = \frac{1}{(b-a)^n}\binom{n}{i}(b-u)^{n-i}(u-a)^i \qquad u \in [a,b]\ i = 0,1,\ldots,n$$



Eigenschaften Bernstein Polynome...

- ergeben eine lineare Basis für Polynome der Ordnung n (man kann mit ihnen jedes Polynom der Ordnung n zusammenbauen)
- haben genau eine Maximalstelle bei $t = \frac{i}{n}$
- haben eine Nullstelle bei 0 (Ordnung i) und bei 1 (Ordnung $n-1$)
- sind symmetrisch: $B_{i,n}(t) = B_{n-i,n}(1-t)$
- sind zwischen $t \in [0,1]$ begrenzt auf $[0,1]$
- ergeben in der Summe: $\sum_{i=0}^{n} B_{i,n}(t) = 1$

$\frac{d}{dt}B_{i,n}(t) = n(B_{i-1,n-1}(t) - B_{i,n-1}(t)) = -n\Delta B_{i-1,n-1}(t)$
$\frac{d^2}{dt^2}B_{i,n}(t) = n(n-1)(B_{i-2,n-2}(t) - 2B_{i-1,n-2}(t) + B_{i,n-2}(t)) = n(n-1)\Delta^2 B_{i-2,n-2}(t)$
$\frac{d^k}{dt^k}B_{i,n}(t) = (-1)^k n(n-1)\ldots(n-k+1)\Delta^k B_{i-k,n-k}(t)$

Pascal'sches Dreieck

$n=0$:				1				
$n=1$:			1		1			
$n=2$:		1		2		1		
$n=3$:	1		3		3		1	
$n=4$:	1		4		6		4		1

Bernstein Polynome $0 \le t \le 1$

$B_{0,0}(t) = 1$				
$B_{0,1}(t) = 1-t$	$B_{1,1}(t) = t$			
$B_{0,2}(t) = (1-t)^2$	$B_{1,2}(t) = 2t(1-t)$	$B_{2,2}(t) = t^2$		
$B_{0,3}(t) = (1-t)^3$	$B_{1,3}(t) = 3t(1-t)^2$	$B_{2,3}(t) = 3t^2(1-t)$	$B_{3,3}(t) = t^3$	
$B_{0,4}(t) = (1-t)^4$	$B_{1,4}(t) = 4t(1-t)^3$	$B_{2,4}(t) = 6t^2(1-t)^2$	$B_{3,4}(t) = 4t^3(1-t)$	$B_{4,4}(t) = t^4$

4.2.2 Simple Bézier Kurven

Eine Bézier-Kurve wird über Kontrollpunkte ($\vec{P}_0, \vec{P}_1, \ldots \vec{P}_n$ ($n \ge 2$) in R^d) sowie die Bernstein Polynome definiert:

$$\vec{r}(t) = \sum_{i=0}^{n} \vec{P}_i B_{i,n}(t) \quad t \in [0,1]$$

Eigenschaften

- Die Bézier-Kurven liegen immer innerhalb der konvexen Hülle der Kontrollpunkte.
- $\vec{r}(0) = \vec{P}_0$ $\qquad \vec{r}(1) = \vec{P}_n$
  $\vec{r}'(0) = n(\vec{P}_1 - \vec{P}_0)$ $\qquad \vec{r}'(1) = n(\vec{P}_n - \vec{P}_{n-1})$
  $\vec{r}''(0) = n(n-1)(\vec{P}_2 - 2\vec{P}_1 + \vec{P}_0)$ $\qquad \vec{r}''(1) = n(n-1)(\vec{P}_n - 2\vec{P}_{n-1} + \vec{P}_{n-2})$
- Wenn C^k Übergang zu einem Punkt gefordert ist, sind k Gleichungen bzw. k Kontrollpunkte pro Punkt nötig

BY-NC-SA

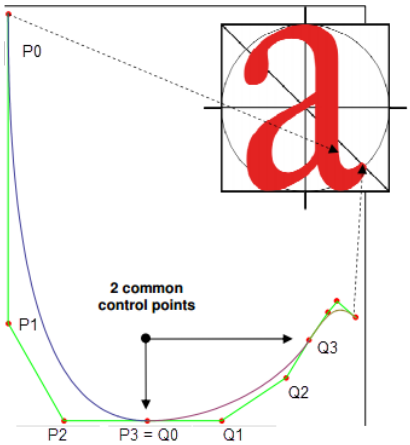
4.2.3 Casteljau recurrence

Die Castaljau recurrence ist eine ähnliche Idee wie Neville-Aitken. Damit kann ein Punkt auf einer Bézier Kurve als Linearkombination zweier Punkte aus Bézier Kurven tieferer Ordnung berechnet werden.

$$\vec{r}_{\vec{P}_0,\vec{P}_1,\ldots,\vec{P}_n}(t) = (1-t)\cdot\vec{r}_{\vec{P}_0,\vec{P}_1,\ldots,\vec{P}_{n-1}}(t) + t\cdot\vec{r}_{\vec{P}_1,\vec{P}_2,\ldots,\vec{P}_n}(t) \qquad t\in[0,1]$$

4.2.4 Zusammengesetzte (Composite) Bézier Kurven

Die zusammengesetzten (simplen) Bézier-Kurven sollen die bekannte Bedingungen (C^k) erfüllen. Dies sind die Gleichungen um den Punkt Q_0 mit Kontrollpunkten $Q_{1,\ldots,m-1}$ nach Q_0 sowie den Kontrollpunkten $P_{1,\ldots,n-1}$ vor Q_0.



Hier sind $m = n = 3$

$$\begin{aligned}
C^0: &\quad \boxed{\vec{P}_n = \vec{Q}_0}\\
C^1: &\quad \vec{r}'_P(1) = \boxed{n(\vec{P}_n - \vec{P}_{n-1}) = m(\vec{Q}_1 - \vec{Q}_0)} &= \vec{r}'_Q(0)\\
C^2: &\quad \vec{r}''_P(1) = \boxed{n(n-1)(\vec{P}_n - 2\vec{P}_{n-1} + \vec{P}_{n-2}) = m(m-1)(\vec{Q}_2 - 2\vec{Q}_1 + \vec{Q}_0)} &= \vec{r}''_Q(0)\\
C^k: &\quad \vec{r}^{(k)}_P(1) = \boxed{n(n-1)\ldots(n-k+1)(\Delta^k\vec{P}_{n-k}) = m(m-1)\ldots(m-k+1)(\Delta^k\vec{Q}_0)} &= \vec{r}^{(k)}_Q(0)
\end{aligned}$$

m ist dabei der Grad von $\vec{r}_Q$, n derjenige von $\vec{r}_P$.

4.2.5 Zusammengesetzte (Composite) Bézier Kurven mit $P_{i,j}$

Wird $\vec{r_j}(t) = \sum_{i=0}^{n}\vec{P}_{i,j}B_{i,n}(t) \quad t\in[0,1]$ mit den Fixpunkten Q_j definiert (Grad: n), ergeben sich folgende Formeln:

$$\begin{aligned}
C^0: &\quad \boxed{\vec{P}_{n,j-1} = \vec{P_{0,j}} = \vec{Q}_j}\\
C^1: &\quad \vec{r}'_{j-1}(1) = \boxed{\vec{Q}_j - \vec{P}_{n-1,j-1} = \vec{P}_{1,j} - \vec{Q}_j} &= \vec{r}'_j(0)\\
C^2: &\quad \vec{r}''_{j-1}(1) = \boxed{\vec{Q}_j - 2\vec{P}_{n-1,j-1} + \vec{P}_{n-2,j-1} = \vec{P}_{2,j} - 2\vec{P}_{1,j} + \vec{Q}_j} &= \vec{r}''_j(0)\\
C^k: &\quad \vec{r}^{(k)}_{j-1}(1) = \boxed{\Delta^k\vec{P}_{n-k,j-1} = \Delta^k\vec{P}_{0,j}} &= \vec{r}^{(k)}_j(0)
\end{aligned}$$

4.2.6 Bézier Oberflächen als Tensor Splines

Bézier Oberflächen werden gleich wie Bézier Kurven mittels einem Tensorprodukt aus Bernstein Polynomen gebildet. Eine Oberfläche im Intervall $[0,1]\times[0,1]$ wird gebildet durch

$$\vec{z}(s,t) = \sum_{i=0}^{n}\sum_{j=0}^{m}\vec{P}_{ij}B_{in}(s)B_{jm}(t) \qquad s,t\in[0,1]$$

Details und Formeln siehe Skript S. 19 – 23.

4.2.7 Vergleich Bézier- und Newton-Interpolation mit gleichverteilten Argumenten auf X-Achse

Aus Sicht Bézier:

+ Kleiner Fehler, keine Oszillation, kein Runge (bei kleinem Grad, z.B. 3)
+ $O(n)$ (linearer Rechenaufwand)
- Nicht eingebettet (neue Daten brauchen Neuberechnung der Interpolation)
- Post-Processing (Grafik, Integrations, Ableitungen) sind komplexer, da jedes Kurvenstück einzeln betrachtet werden muss.

BY-NC-SA

5 Least-Squares Approximation

5.1 Lineare Least-Squares

Ziel: Approximation von Messpunkten durch Minimieren einer Funktion, die die Abweichung in y angibt (Residuen). Die Datenpunkte werden bei dieser Methode nicht mehr genau getroffen, ausserdem ist der Grad m des Approximation-Polynoms meist massiv kleiner als die Anzahl der gegebenen Datenpunkte $N+1$.

Aus einer Menge von Basisfunktionen $g_0, g_1, \ldots, g_m$ mit $m \ll N$ und den Messungen $(x_0, y_0), (x_1, y_1), \ldots, (x_N, y_N)$ wird das überdeterminierte $(m < N)$ Gleichungssystem aufgestellt und nach den unbekannten a_i gelöst.

$$\overbrace{\begin{pmatrix} g_0(x_0) & g_1(x_0) & \ldots & g_m(x_0) \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ g_0(x_N) & g_1(x_N) & \ldots & g_m(x_N) \end{pmatrix}}^{\text{Designmatrix } G} \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ \vdots \\ y_N \end{pmatrix} \quad \Leftrightarrow \quad G \cdot a = y$$

Zu minimierende Fehlerfunktion:

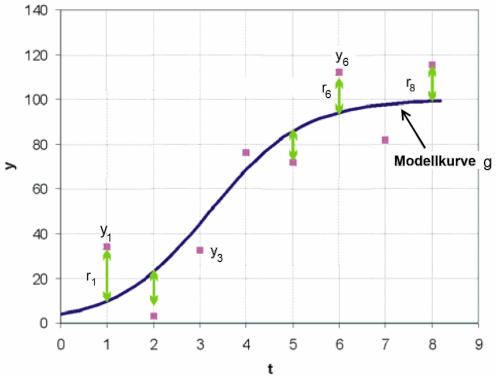
$$\underbrace{S}_{Fehler} = \sum_{i=0}^{N} \Big(y_i - \underbrace{\sum_{j=0}^{m} a_j g_j}_{Modell} \Big)^2 = \sum_{i=0}^{N} r_i^2$$

Modellfunktion:

$$y = \sum_{j=0}^{m} a_j g_j \overset{Poly!}{=} \sum_{j=0}^{m} a_j x^j$$

Abweichungen (Residuen):

$$r_i = y_i - \sum_{j=0}^{m} a_j g_j$$

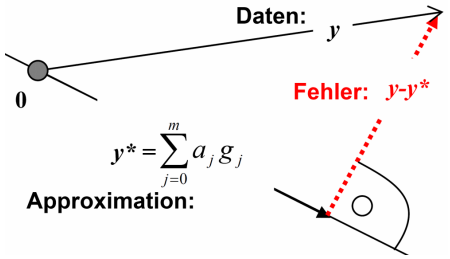


5.1.1 Normalgleichung

Minimieren des quadratischen Fehlers mit „Normalmatrix“ G (Designmatrix):

$$\underbrace{\boldsymbol{G^T G}}_{Normalmatrix} \cdot \boldsymbol{a} = \boldsymbol{G^T y} \quad \Rightarrow \quad \boldsymbol{a} = (\boldsymbol{G^T G})^{-1} \boldsymbol{G^T y}$$

$$\underbrace{\underbrace{\boldsymbol{G^T}}_{(m+1)\times(N+1)} \cdot \underbrace{\boldsymbol{G}}_{(N+1)\times(m+1)}}_{(m+1)\times(m+1)} \cdot \underbrace{\boldsymbol{a}}_{(m+1)\times 1} = \underbrace{\underbrace{\boldsymbol{G^T}}_{(m+1)\times(N+1)} \cdot \underbrace{\boldsymbol{y}}_{(N+1)\times 1}}_{(m+1)\times 1}$$



Die symmetrische $(m+1)\times(m+1)$ Matrix $G^T \cdot G$ ist positiv definit wenn g_i linear unabhängig sind. Das neue Gleichungssystem ist regulär und hat eine eindeutige Lösung für die Koeffizienten a_j.

$$\overbrace{\begin{pmatrix} \langle g_0|, g_0| \rangle & \langle g_1|, g_0| \rangle & \ldots & \langle g_m|, g_0| \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle g_0|, g_m| \rangle & \langle g_1|, g_m| \rangle & \ldots & \langle g_m|, g_m| \rangle \end{pmatrix}}^{G^T \cdot G} \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} \langle y|, g_0| \rangle \\ \vdots \\ \vdots \\ \langle y|, g_m| \rangle \end{pmatrix}$$

5.1.2 QR-Zerlegung

Eine quadratische $n \times n$ Matrix Q ist orthogonal, falls $Q^T \cdot Q = Q \cdot Q^T = I$ bzw. $Q^T = Q^{-1}$. Jede beliebige Matrix G mit den Dimensionen $(N+1) \times (m+1)$ mit $N \geq m$ und Rang $m+1$ kann als Produkt einer orthogonalen $(N+1) \times (N+1)$ Matrix Q und einer $(N+1) \times (m+1)$ oberen Dreiecksmatrix R.
Aus dem Gleichungssystem $Ga| = y|$ wird dann $Ra| = Q^T y|$, was einfacher lösbar ist[Citation needed].

BY-NC-SA

5.1.3 Singulärwertzerlegung (singular value decomposition, SVD)

Jede Matrix $\boldsymbol{G}$ mit Dimensionen $(N+1)\times(m+1)$ kann so zerlegt werden:

$$\boldsymbol{G} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T = \boldsymbol{U}\cdot\begin{bmatrix} d_{00} & 0 & \dots & 0 \\ 0 & d_{11} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & \dots & 0 & d_{mm} \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & 0 \end{bmatrix}\cdot\boldsymbol{V}^T$$

$\boldsymbol{D}$ $((N+1)\times(m+1))$ ist die zentrale Diagonalmatrix mit Singulärwerten, $\boldsymbol{U}$ $((N+1)\times(N+1))$ und $\boldsymbol{V}$ $((m+1)\times(m+1))$ sind orthogonale und quadratische Matrizen (d.h. u.a. $\boldsymbol{U}^T=\boldsymbol{U}^{-1}$).

Die Zerlegung an sich wird hier nicht weiter ausgeführt, diese kann anderswo nachgeschlagen werden. SVD kann gebraucht werden, um Gleichungssystem zu lösen (wie hier), um Eigenvektoren und Eigenwerte sowie die Inverse zu berechnen.
Aus $Ga| = y|$ wird $UDV^Ta| = y|$. Dies lässt sich auflösen zu

$$a| = VD^{-1}U^T\cdot y|$$

5.1.4 Monomiale

Mit $g = a_0\underbrace{1}_{g_0} + a_1\underbrace{x}_{g_1} + a_2\underbrace{x^2}_{g_2} + \dots + a_m\underbrace{x^m}_{g_2}$ wird die Designmatrix zu:

$$G \underbrace{=}_{\text{When Monomials}} \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^m \\ 1 & x_1 & x_1^2 & \dots & x_1^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \dots & x_N^m \end{bmatrix}$$

5.1.5 Gleichförmige Argumente (Orthogonalitäts-Eigenschaften)

Mit gleichförmigen Elementen $x_i - x_j = (j-i)h$ für alle $i,j \to \{x_0 ... x_N\} = \{x_0 + t\cdot h\}_{t=0\dots N}$ und **orthogonalen Polynomen** kann GG^T diagonalisiert und so die Gleichungen rechnerisch einfacher gelöst werden.

$$p_{k,N}(t) = \sum_{i=0}^{k}(-1)^i\binom{k}{i}\binom{k+i}{i}\frac{t^{(i)}}{N^{(i)}} = 1 + \sum_{i=1}^{k}(-1)^i\binom{k}{i}\binom{k+i}{i}\frac{t(t-1)(t-2)\dots(t-i+1)}{N(N-1)(N-2)\dots(N-i+1)} \qquad (k=1,\dots,N)$$

mit $t = \frac{x-x_0}{h}$ $\qquad \binom{k}{i} = \frac{k!}{i!(k-i)!} = nCr(k,i)$
$p_{k,N}$ kann jetzt als g_k in die Designmatrix eingesetzt werden und das Produkt G^TG wird zu einer $(m+1)\times(m+1)$ Diagonalmatrix. Schlussendlich kann a über die bekannte Formel $G^TGa = G^Ty$ berechnet werden.

Beispiel:

$$\boldsymbol{x} = \begin{bmatrix}3 & 4 & 5 & 6 & 7\end{bmatrix} \quad \Rightarrow \quad \boldsymbol{t} = \boldsymbol{x} - 3 = \begin{bmatrix}0 & 1 & 2 & 3 & 4\end{bmatrix}$$

$$\Rightarrow P_0(x) = 1 \qquad P_1(x) = 1 - \frac{x-3}{2} \qquad P_2(x) = 1 - \frac{2\cdot 3(x-3)}{4} + \frac{1\cdot 3(x-3)(x-3-1)}{4\cdot 3} = 1 - \frac{3x-9}{2} + \frac{1}{2}(x-4)(x-3)???$$

$$\boldsymbol{G} = \begin{matrix}P_0 & P_1 & P_2\end{matrix}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1/2 & -1/2 \\ 1 & 0 & -1 \\ 1 & -1/2 & -1/2 \\ 1 & -1 & 1 \end{bmatrix} \quad \Rightarrow \quad \boldsymbol{G}^T\boldsymbol{G} = \begin{bmatrix} \|P_0\|^2 & 0 & 0 \\ 0 & \|P_1\|^2 & 0 \\ 0 & 0 & \|P_2\|^2 \end{bmatrix} = \begin{bmatrix} 5 & 0 & 0 \\ 0 & \frac{5}{2} & 0 \\ 0 & 0 & \frac{7}{2} \end{bmatrix}$$

BY-NC-SA

5.1.6 Chebyshev Orthogonale Polynome

Idee: Approximation eines kontinuierlichen Polynoms durch Chebyshev-Polynome.
Definition
Chebyshev Polynome sind definiert als $\boxed{T_n(x) = \cos(n \arccos(x))}$ mit $(n = 0, 1, \ldots)$ und $(-1 \leq x \leq 1)$. Dieses Polynom bewirkt eine Häufung der Messwerte in den Randbereichen. Die ersten paar Polynome T_n sowie deren Umformungen nach x:

$T_0 = 1$	$x^0 = 1 = T_0$
$T_1 = x$	$x^1 = x = T_1$
$T_2 = 2x^2 - 1$	$x^2 = \frac{1}{2}T_2 + \frac{1}{2}T_0$
$T_3 = 4x^3 - 3x$	$x^3 = \frac{1}{4}T_3 + \frac{3}{4}T_1$
$T_4 = 8x^4 - 8x^2 + 1$	$x^4 = \frac{1}{8}T_4 + \frac{1}{2}T_2 + \frac{3}{8}T_0$
$T_5 = 16x^5 - 20x^3 + 5x$	$x^5 = \frac{1}{16}T_5 + \frac{5}{16}T_3 + \frac{5}{8}T_1$

Weitere Chebyshev Polynome können mit der Rekursionsformel $T_{n+1}(x) = 2xT_n(x) - T_{n-1}$ $(n \geq 2)$ mit Initialbedingungen $T_1(x) = x,\ T_0(x) = 1$ berechnet werden.

Eigenschaften:

- Die maximale Amplitude des Chebyshev-Polynoms ist $\frac{1}{2^n}$ (max Amplitude des Fehlerterms).
- Amplitude: $T_n(x) \in [-1, +1]$
- Nullstellen: $T_n(x) = 0 \Leftrightarrow x = \cos(\frac{2i+1}{2n}\pi) \qquad i = 0, 1, \ldots, n-1$ (Diese Nullstellen sind die Chebyshev-Knoten)
- $T_n(x) = \pm 1 \Leftrightarrow x = \cos(\frac{i\pi}{n})$ $(i = 0, 1, \ldots, n)$

Rezept
Chebyshev Polynome können nur benutzt werden, wenn x **mit Chebyshev-Abständen** $x_i = \cos(\frac{2i+1}{2n}\pi)$ $(i = 0, 1, \ldots, n-1)$ und nicht mit gleichverteilten Abständen gewonnen wurden.
Ziel: $y(t) = p_N(t)$ (Polynom, definiert in $[a, b]$) mit Chebyshev-Polynomen des Grades m approximieren.

1. $y(t)$ auf das Standardintervall $[-1, 1]$ normieren (affine Transformation) $t = a + \frac{b-a}{2}(x + 1)$.
2. $y(x)$ aus T_n zusammensetzen
3. $y(x)$ auf Grad m kürzen (truncate): $y_m(x)$
4. Zurücktransformieren: $x = 2\frac{t-a}{b-a} - 1$
5. T_n in y_m einsetzen (siehe Tabelle oben)
6. Fehlerabschätzung für truncate Methode: $\max_t |y(t) - y_m(t)|$ (Abgeschnittener Teil)

Beispiel
Gesucht: Approximation $y(t) = t^3$ mit Grad $m = 2$ für Intervall $(a, b) = (0, 1)$

1. Transformation mit $t = \frac{x+1}{2}$:
   $y(x) = \left(\frac{x+1}{2}\right)^3 = \frac{1}{8}(x^3 + 3x^2 + 3x + 1)$
2. Mit Tabelle erweitern:
   $y(x) = \frac{1}{8}\left(\frac{T_3(x)+3T_1(x)}{4} + 3\frac{T_2(x)+T_0}{2} + 3T_1(x) + T_0(x)\right)$
   $= \frac{1}{32}T_3(x) + \frac{3}{16}T_2(x) + \frac{15}{32}T_1(x) + \frac{5}{16}T_0(x)$
3. Kürzen auf Grad $m = 2$:
   $y(x) \approx \frac{3}{16}T_2(x) + \frac{15}{32}T_1(x) + \frac{5}{16}T_0(x)$
4. Zurücktransformieren mit $x = 2t - 1$:
   $y(t) \approx \frac{3}{16}T_2(2t-1) + \frac{15}{32}T_1(2t-1) + \frac{5}{16}T_0(2t-1)$
5. $T_n(2t-1)$ ersetzen:
   $y(t) \approx \frac{3}{16}(2(2t-1)^2 - 1) + \frac{15}{32}(2t-1) + \frac{5}{16}$
6. Fehlerabschätzung:
   $\max_t \left|\frac{1}{32}T_3(2t-1)\right|$

BY-NC-SA

Die **Designmatrix** sieht so aus:

$$G \underbrace{=}_{\text{When Chebyshev}} \begin{bmatrix} T_0(x_0)=1 & T_1(x_0)=x_0 & \dots & T_m(x_0) \\ T_0(x_1)=1 & T_1(x_1)=x_1 & \dots & T_m(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ T_0(x_N)=1 & T_1(x_N)=x_N & \dots & T_m(x_N) \end{bmatrix}_{N\times m}$$

Damit wird G^TG zu:

$$G^TG \underbrace{=}_{\text{When Chebyshev}} \begin{bmatrix} N+1 & 0 & \dots & 0 \\ 0 & \frac{N+1}{2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{N+1}{2} \end{bmatrix}_{m\times m}$$

und $(G^TG)^{-1}G^T$ zu:

$$(G^TG)^{-1}G^T \underbrace{=}_{\text{When Chebyshev}} \begin{bmatrix} \frac{1}{N+1}T_0(x_0)=\frac{1}{N+1} & \frac{1}{N+1}T_0(x_1)=\frac{1}{N+1} & \dots & \frac{1}{N+1}T_0(x_N)=\frac{1}{N+1} \\ \frac{2}{N+1}T_1(x_0)=\frac{2}{N+1}x_0 & \frac{2}{N+1}T_1(x_1)=\frac{2}{N+1}x_1 & \dots & \frac{2}{N+1}T_1(x_N)=\frac{2}{N+1}x_N \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{N+1}T_m(x_0) & \frac{2}{N+1}T_m(x_1) & \dots & \frac{2}{N+1}T_m(x_N) \end{bmatrix}_{m\times N}$$

5.1.7 Diskrete Least-Squares Chebyshev Approximation

Folgt aus $a=(G^TG)^{-1}G^Ty$ für $x_i=\cos(\frac{2i+1}{2(N+1)}\pi)$ $(i=0,1,\dots,N)$:

$$p(x)=\sum_{j=0}^{m} a_jT_j(x) \quad \text{wobei} \quad a_j=\begin{cases} \frac{1}{N+1}\sum_{i=0}^{N} y(x_i) & j=0 \\ \frac{2}{N+1}\sum_{i=0}^{N} T_j(x_i)y(x_i) & j>0 \end{cases}$$

5.1.8 Kontinuierliche Least-Squares

Für kontinuierliche Funktionen, die **keine Polynome** sind (!) kann auch die kontinuierliche Version verwendet werden.
S: zu minimierende Fehlerfunktion, w: Gewicht mit $(-1\le x\le 1)$

$$S=\int_{-1}^{1}(y(x)-p(x))^2\cdot w(x)\cdot \mathrm{d}x$$

Kontinuierliche Least-Squares Chebyshev Approximation
Das Gewicht wird besonders auf den Rand gelegt (wie bei Chebyshev üblich). $(w(x)=\frac{1}{\sqrt{1-x^2}})$

$$p(x)=\sum_{j=0}^{m} a_jT_j(x) \quad \text{wobei} \quad a_j=\begin{cases} \frac{1}{\pi}\int_{-1}^{1}\frac{y(x)}{\sqrt{1-x^2}}dx & j=0 \\ \frac{2}{\pi}\int_{-1}^{1}\frac{y(x)T_j(x)}{\sqrt{1-x^2}}dx & j>0 \end{cases}$$

Kontinuierliche Least-Squares Legendre Approximtion:
Hier bei wird das Gewicht $w(t)=1$ gesetzt.

Legendre Polynome:

$$P_n(x)=\frac{1}{2^n n!}\cdot\frac{\mathrm{d}^n}{\mathrm{d}x^n}(x^2-1)^2 \qquad P_0(x)=1 \qquad P_1(x)=x \qquad P_2(x)=\frac{1}{2}(3x^2-1)$$

$$P_3(x)=\frac{1}{2}(5x^3-3x) \qquad P_4(x)=\frac{1}{8}(35x^4-30x^2+3) \qquad P_5(x)=\frac{1}{8}(63x^5-70x^3+15x)$$

Koeffizenten:

$$p(x)=\sum_{j=0}^{m} a_jT_j(x) \quad \text{wobei} \quad a_j=\frac{2j+1}{2}\cdot\int_{-1}^{1} y(x)P_j(x)\cdot \mathrm{d}x \qquad (j=0,1,...,m)$$

BY-NC-SA

5.2 Multi-Variate Lineare Least Squares Approximation

Multi-Variate Least Square Probleme werden genau gleich wie die Uni-Variaten Least Square Probleme gelöst (selbe Methoden)! Die Variable x ist keine Zahl mehr, sondern ein Vector mit d Dimensionen!
Basisgewinnung: Die Basen werden mittel Tensorenprodukt aus Uni-Variaten Basen berechenet (Alle mit allen Multiplizieren (d-Basen in d-Variabeln)).

$$\sum_{j_1=0}^{m_1}\sum_{j_2=0}^{m_2}...\sum_{j_d=0}^{m_d} a_{j_1,j_2,...,j_d}\cdot g_{j_1}(x^{(1)})g_{j_2}(x^{(2)})\cdot\cdot...\cdot g_{j_d}(x^{(d)})$$

Beispiele

Basis 3. Grad:	$\{1, x, y,\ x^2, 2xy, x^2,\ x^3, 3x^2y, 3xy^2, y^3\}$
Basis 4. Grad:	$\{1, x, y,\ x^2, 2xy, x^2,\ x^3, 3x^2y, 3xy^2, y^3,\ x^4, 4x^3y, 6x^2y^2, 4xy^2, y^4\}$

5.3 Legendre Polynome

Die Legendre P-Polynome werden durch die Rodriguez' Formel bestimmt

$$P_n(x) = \frac{1}{2^n n!}\cdot\frac{d^n}{dx^n}\left(x^2-1\right)^n$$

$$P_0(x)=1 \qquad P_1(x)=x \qquad P_2(x)=\frac{1}{2}(3x^2-1) \qquad P_3(x)=\frac{1}{2}(5x^3-3x) \quad P_4(x)=\frac{1}{8}(35x^4-30x^2+3)$$

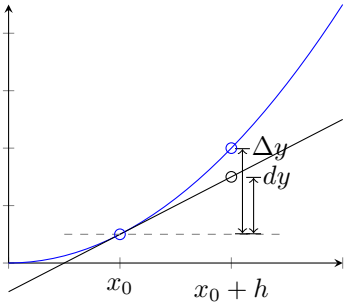
BY-NC-SA

6 Fehlerfortpflanzung

6.1 Differential

Das Differential *df* bezeichnet den linearen Anteil des Zuwachses einer Variablen einer Funktion.

$$\Delta f = f(x_0 + h) - f(x_0) \approx df = f'(x_o)dx = f'(x_0)h = f'(x_0)\Delta x$$



Δf ist der *absolute* Fehler von f bei x_0. Der *relative* Fehler ist $\frac{\Delta f}{f} \approx \frac{df(x_0)}{f(x_0)}$
Jede n-fach ableitbare Funktion kann mit einer **Taylor**-Reihe approximiert werden:

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x_0) + \frac{h^3}{3!}f'''(x_0) + \ldots + \frac{h^n}{n!}f^{(n)} + R_n(x_0, h)$$

Wobei $R_n(x_0, h)$ das Restglied bezeichnet und mit $n \to \infty$ gegen 0 mit Geschwindigkeit $o(h^n)$ („schnell“) konvergiert. Es gibt bspw. die

$$\text{Lagrange Form: } R_n(x_0, h) = \frac{h^{n+1}}{(n+1)!}f^{(n+1)}(\underbrace{x_0 + \vartheta h}_{\xi}) \qquad (\xi \in (x_0, x_0 + h), \vartheta \in (0, 1)) \qquad \text{oder die}$$

$$\text{Cauchy Form: } R_n(x_0, h) = \frac{h^n(1-\vartheta)^n}{(n+1)!}f^{(n+1)}(\underbrace{x_0 + \vartheta h}_{\xi}) \qquad (\xi \in (x_0, x_0 + h), \vartheta \in (0, 1))$$

Mit der Taylor-Reihe wird der absolute Fehler zu

$$\Delta f = \frac{1}{1!}df(x_0) + \frac{1}{2!}d^2f(x_0) + \ldots + \frac{1}{n!}d^nf(x_0) + R_n(x_0, h)$$

6.2 Multivariate Differentiale, Taylor

Das Differential sieht im Multidimensionalen so aus:

$$\Delta f \approx df(\vec{x}) = \sum_{i=1}^{n} \frac{\partial f(\vec{x})}{\partial x_i}h_i = h_1\frac{\partial f(\vec{x})}{\partial x_1} + \ldots + h_n\frac{\partial f(\vec{x})}{\partial x_n} \qquad \text{mit} \quad \vec{h} = [h_1, \ldots, h_d]^T,\ \vec{x} = [x_1, \ldots x_d]^T$$

Das Vektorfeld der partiellen Ableitungen wird das Gradientenfeld von f genannt und mit $\text{grad}(f)$ oder $\vec{\nabla}f(\vec{x})$ bezeichnet. Damit ist $\Delta f \approx \vec{\nabla}f(\vec{x}) \cdot \vec{h}$.
Im Multi-variaten Fall wird die Taylor-Reihe für Multiindizes $\alpha = \{\alpha_1, \ldots, \alpha_n\}$ und $|\alpha| = \alpha_1 + \ldots + \alpha_n$ zu

$$f(\vec{x}_0 + h) = f(\vec{x}_0) + \frac{1}{1!}\vec{\nabla}f(\vec{x}_0) \cdot \vec{h} + \sum_{|\alpha|=2}^{N} \frac{1}{\alpha!}\frac{\partial^{|\alpha|}f(\vec{x}_0)}{\partial x^\alpha} \cdot \vec{h}^\alpha + \sum_{|\alpha|=N+1} R_\alpha(\vec{x}_0, \vec{h}) \cdot \vec{h}^\alpha$$

oder vereinfacht für 2. Ordnung

$$f(\vec{x}_0 + \vec{h}) = f(\vec{x}_0) + \vec{h}\nabla f(\vec{x}_0) + \underbrace{\left(\frac{h_1^2}{2!}\frac{\partial^2 f}{\partial x_1^2} + \frac{h_2^2}{2!}\frac{\partial^2 f}{\partial^2 x_2^2} + \frac{h_1 h_2}{1!1!}\frac{\partial^2 f}{\partial x_1 \partial x_2}\right)}_{\text{für 2. Ordnung}} +$$

$$R_{(3,0)}(\vec{x}_0, \vec{h}) \cdot \vec{h}_1^3 + R_{(2,1)}(\vec{x}_0, \vec{h}) \cdot \vec{h}_1^2\vec{h}_2 + R_{(1,2)}(\vec{x}_0, \vec{h}) \cdot \vec{h}_1\vec{h}_2^2 + R_{(0,3)}(\vec{x}_0, \vec{h}) \cdot \vec{h}_2^3$$

6.3 Jacobi-Matrix

Die Jacobi-Matrix bildet alle ersten Ableitungen einer Funktion ab. Wenn die Jacobi-Matrix quadratisch ist ($m = n$), dann kann dessen Determinante $\det(J)$ als transformiertes Volumen interpretiert werden:

$$\underbrace{dy_1 dy_2 \ldots dy_n}_{\text{„transformiertes Volumenelement“}} = \det(J_f(\vec{x}))\ \underbrace{dx_1 dx_2 \ldots dx_n}_{\text{„Volumenelement“}}$$

$$J_f(\vec{x}) := \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\vec{x}) & \frac{\partial f_1}{\partial x_2}(\vec{x}) & \ldots & \frac{\partial f_1}{\partial x_n}(\vec{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(\vec{x}) & \frac{\partial f_m}{\partial x_2}(\vec{x}) & \ldots & \frac{\partial f_m}{\partial x_n}(\vec{x}) \end{bmatrix}$$

Die Jacobi-Determinante kann daher auch als ein Mass für Fehlerfortpflanzung sein. Die Berechnung erfolgt direkt über die Determinanten-Regel, über

$$\frac{D(f_1, f_2, \ldots, f_n)}{D(x_1, x_2, \ldots, x_n)} = \det(J_f(\vec{x})) \quad \text{oder} \quad \det(J_f) = \sqrt{\det\left(\underbrace{J_f^T(\vec{x})J_f(\vec{x})}_{\text{Diagonalmatrix}}\right)}.$$

Ausserdem gilt für die Transformation T und deren Inverse T^{-1}: $\det(J_T) = \frac{1}{\det J_{T^{-1}}}$

BY-NC-SA

7 Numerische Ordinäre Differentialgleichungen

7.1 Definitionen

Ordinäre DGL (ODE - ordinary differential equation)

$$\boxed{y'(x) = f(x, y(x)) \qquad \text{mit} \qquad y(x_0) = y_0}$$

$f(x, y)$ wird auch als r.h.s. Funktion bezeichnet (right hand side).

7.2 Explizite Methoden

Es wird vorwärts in der Zeit gerechnet, was keine Solver braucht. Ansatz: Taylor-Approximation. Dies wird aber sehr schnell sehr aufwendig, da schnell höhere Ableitungen in mehrere Dimensionen nötig werden.

$$\boxed{y(x_0 + h) = y(x_0) + \frac{h}{1!}y'(x_0) + \frac{h^2}{2!}y''(x_0) + \ldots + \frac{h^p}{p!}y^{(p)}(x_0) + \underbrace{\frac{h^{p+1}}{(p+1)!}y^{(p+1)}(\xi)}_{\text{Restterm}}}$$

Die Herleitung weiterer Ableitungen erfolgt mithilfe der Kettenregel:

$$\boxed{y''(x) = \frac{\partial f(x,y)}{\partial x}1 + \frac{\partial f(x,y)}{\partial y}f(x,y) = \frac{\partial y'(x)}{\partial x}1 + \frac{\partial y'(x)}{\partial y}y'(x)} \qquad \boxed{y'''(x) = \frac{\partial y''(x)}{\partial x}1 + \frac{\partial y''(x)}{\partial y}y'(x)} \qquad \ldots$$

Ableitungen bis zum dritten Grad sind hier aufgelistet (Immenser Rechenaufwand nötig):

$$\begin{aligned} y(x+h) = \quad & y(x) + \tfrac{f(x,y)}{1!}h + \\ & \tfrac{1}{2!}\left(\tfrac{\partial f(x,y)}{\partial x}1 + \tfrac{\partial f(x,y)}{\partial y}f(x,y)\right)h^2 + \\ & \tfrac{1}{3!}\left(\tfrac{\partial^2 f(x,y)}{\partial x^2}1 + 2\tfrac{\partial^2 f(x,y)}{\partial x \partial y}f(x,y) + \tfrac{\partial^2 f(x,y)}{\partial y^2}f(x,y)^2 + \left(\tfrac{\partial f(x,y)}{\partial y}\right)^2 f(x,y) + \tfrac{\partial f(x,y)}{\partial x}\tfrac{\partial f(x,y)}{\partial y}\right)h^3 + \ldots + \\ & \tfrac{1}{4!}y^{(4)}(x)h^4 + \ldots + \tfrac{1}{p!}y^{(p)}(x)h^p + \underbrace{\frac{1}{(p+1)!}y^{(p+1)}(\xi)h^{p+1}}_{\text{Restterm}} \end{aligned}$$

7.2.1 Explizite Euler-Methode

Spezialfall der expliziten Methoden:
Taylor mit Ordnung $p = 1$
Relativ ungenaue Approximation, da hier die Ableitung pro Schritt unverändert bleibt.

Globaler Fehler: $\max_{0 \le i \le k} |y_i - y(x_i)|$ mit Anzahl Schritten k
Lokaler Fehler: $y(x_n + h) - y_{n+1}$ wenn Initialwert des aktuellen Schritts is genau richtig

$$\begin{aligned} y_0 &= y(x_0) \\ y(x_0 + h) \approx y_1 &= y_0 + f(x_0, y_0)h \approx y(x_0) + y'(x_0)h \\ y(x_1 + h) \approx y_2 &= y_1 + f(x_1, y_1)h \approx y(x_1) + y'(x_1)h \\ &\vdots \\ y(x_{n-1} + h) \approx y_n &= y_{n-1} + f(x_{n-1}, y_{n-1})h \approx y(x_{n-1}) + y'(x_{n-1})h \end{aligned}$$

7.2.2 Explizite Taylor-Methoden höherer Ordnung

Wenn $p > 1$ wird eine bessere Approximation erreicht. Allerdings steigt hier der rechnerische Aufwand enorm. Siehe Section 7.2 für Berechnung.

$$\begin{aligned} y_0 &= y(x_0) \\ y_{k+1} &= y_k + \frac{y'_k}{1!}h_k + \frac{y''_k}{2!}h_k^2 + \ldots + \frac{y_k^{(p)}}{p!}h_k^p \end{aligned}$$

BY-NC-SA

7.2.3 Explizite Runge-Kutta (RK) Methoden S8

Es werden s rekursive Stages k definiert, welche zusammenaddiert die Taylor-Approximation ergeben. Hier ist $s = 4$

$$k_1 = f(x, y)$$
$$k_2 = f(x + mh, y + mhk_1)$$
$$k_3 = f(x + nh, y + nhk_2)$$
$$k_4 = f(x + ph, y + phk_3)$$

$$y(x + h) \approx y(x) + ahk_1 + bhk_2 + chk_3 + dhk_4$$

Im Gegensatz zum Skript (S. 10) sind die h nicht in den k-Definitionen enthalten, sondern erst in der finalen Formel. Mit den richtigen Nebenbedingungen (S. 11) entspricht diese Formel dem Taylor Polynom bis zur Ordnung 4
Mit der Gleichsetzung zu Taylor definieren die expliziten RK-Methoden ein überbestimmtes Gleichungssystem mit 8 Gleichungen und 7 Unbekannten m, n, p und a, b, c, d.
Es gibt daher verschiedene **Lösungen**, die hier auch für die Ordnung $s = 2$ und $s = 4$ aufgeführt sind:

Name der Lösungen	# Stages (s)	Lösungen
Heun	2	$a = b = \frac{1}{2}$, $m = 1$ $(n = p = c = d = 0)$
Explicit Midpoint	2	$a = 0$, $b = 1$, $m = \frac{1}{2}$ $(n = p = c = d = 0)$
Klassische Runge-Kutta	4	$m = n = \frac{1}{2}$, $p = 1$, $a = d = \frac{1}{6}$, $b = c = \frac{1}{3}$

7.2.4 Generelles Framework für Runge-Kutta Methoden (Butcher Tableaus) S13

$$k_1 = f(x + c_1 h, y + h\underbrace{\sum_{j=1}^{s} a_{1,j} k_j}_{g_1})$$
$$k_2 = f(x + c_2 h, y + h\underbrace{\sum_{j=1}^{s} a_{2,j} k_j}_{g_2})$$
$$k_3 = f(x + c_3 h, y + h\underbrace{\sum_{j=1}^{s} a_{3,j} k_j}_{g_3})$$
$$\vdots$$
$$k_s = f(x + c_s h, y + h\underbrace{\sum_{j=1}^{s} a_{s,j} k_j}_{g_s})$$

} s stages

$$y(x + h) \approx y(x) + h\left(\sum_{j=1}^{s} b_j k_j\right)$$

$$\begin{array}{c|cccc|c} 0 & 0 & 0 & \cdots & 0 & 0 \\ c_2 & a_{2,1} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s-1} & 0 \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array}$$

$\boldsymbol{a}$ = Aufteilung in Y
$\boldsymbol{b}^T$ = Lösungen, durch Methode gegeben
$\boldsymbol{c}$ = Aufteilung in X

FSAL (First Same As Last) Spezialform:

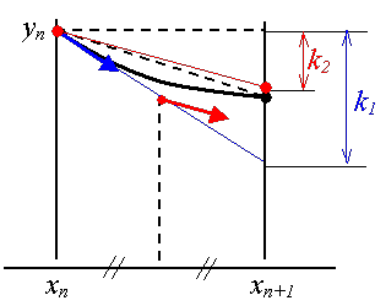
$$\begin{array}{c|cccc|c} 0 & 0 & 0 & \cdots & 0 & 0 \\ c_2 & a_{2,1} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{s-1} & a_{s-1,1} & a_{s-1,2} & \cdots & 0 & 0 \\ 1 & b_1 & b_2 & \cdots & b_{s-1} & 0 \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & 0 \end{array}$$

Bedingungen (Für Explizite Methode):

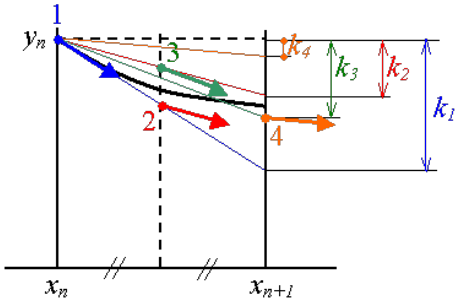
- $c_i = \sum_{j=1}^{s} a_{ij} = \sum_{j=1}^{i-1} a_{ij}$ $(i = 2, \ldots, s)$
- $\sum_{j=1}^{s} b_j = 1$
- $c_1 = 0$
- $a_{1j} = 0$ $\quad 1 \le j \le s$
- $a_{ij} = 0$ $\quad j \ge i$

$\begin{array}{c\|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$	$\begin{array}{c\|cc} 0 & & \\ \hline & 1 & \end{array}$ $\begin{array}{c\|cc} 0 & & \\ 1/2 & 1/2 & \\ \hline & 0 & 1 \end{array}$	$\begin{array}{c\|cc} 0 & & \\ 2/3 & 2/3 & \\ \hline & 1/4 & 3/4 \end{array}$
Classical Runge-Kutta of order 4 (cf. 1.10c)	Explicit Euler method (cf. 1.3A') Explicit Midpoint method (1.9b)	Yet another method (cf. 1.9c)

Heun Methode (RK-2): $b_1 = b_2 = \frac{1}{2}; c_2 = \frac{1}{2}; a_{2,1} = \frac{1}{2}$



Runge-Kutta 2ème ordre :
$k_2 = h\, f\,(x_n + h/2,\ y_n + k_1/2)$



Runge-Kutta 4ème ordre

BY-NC-SA

7.2.5 Adaptive Explizite Methoden S17

Idee: Automatische Adaptierung der Step-Size h. Dies macht eine Definition des maximalen Fehlers der Approximation nötig. Das *Accuracy Goal* (ag) gibt die minimal übereinstimmende Anzahl Nachkommastellen an während das *Precision Goal* (pg) die Anzahl signifikanter Stellen des Resultats repräsentiert. Der Toleranz-Paramter ist:

$$\varepsilon = \varepsilon_a + |y|\varepsilon_r = 10^{-ag} + |y|10^{-pg} \geq |e|.$$

Eingebettete Paare von expliziten Runge-Kutta Methoden

0	0	0	$\cdots$	0	0
c_2	$a_{2,1}$	0	$\cdots$	0	0
$\vdots$	$\vdots$	$\vdots$	$\ddots$	0	$\vdots$
c_{s-1}	$a_{s-1,1}$	$a_{s-1,2}$	$\cdots$	0	0
c_s	$a_{s,1}$	$a_{s,2}$	$\cdots$	$a_{s,s-1}$	0
	b_1	b_2	$\cdots$	b_{s-1}	b_s
	$\hat{b}_1$	$\hat{b}_2$	$\cdots$	$\hat{b}_{s-1}$	$\hat{b}_s$

Die Methode führt eine zweite Approximation, eine *eingebettete Ordnung* $\hat{p}$ ein, welche für die Fehlerrechnung zuständig ist. Die *primäre Ordnung* p wird benötigt, um die Schrittweite zu berechnen. Meist ist $\hat{p} = p - 1$.
Das Butcher-Tableau wird dabei um eine Zeile der b-Zahlen erweitert.

Lokaler Fehler:

$$e_n = y(x+h) - \hat{y}(x+h) = h_n \sum_{j=1}^{s} (b_j - \hat{b}_j) k_j \Rightarrow \|e_n\| = \left\| h_n \sum_{j=1}^{s} (b_j - \hat{b}_j) k_j \right\|$$

Wenn bei aktuellem Schritt $\boxed{\frac{\|e_n\|}{\varepsilon} > 1}$, dann war die Schätzung von h_n zu optimistisch und der Schritt muss mit kleinerer Schrittweite wiederholt werden (**Reject**).

Ansonsten $\boxed{\frac{\|e_n\|}{\varepsilon} \leq 1}$ wird fortgesetzt mit dem Update der Step-Size (**Proceed**):

$$h_{n+1} = h_n \left(\frac{\varepsilon}{\|e_n\|} \right)^{\frac{1}{\tilde{p}}} = h_n \left(\frac{\|e_n\|}{\varepsilon} \right)^{-\frac{1}{\tilde{p}}}$$

Beispiel

x	$\boldsymbol{y}$	h_n	$\left(\frac{\|e_n\|}{\varepsilon}\right)^{-\frac{1}{\tilde{p}}}$	h_{n+1}	Status
10	$[1, -1]^T$	1	0.4	0.4	Reject
10	$[1, -1]^T$	0.4	1.5	0.6	Proceed
10.4	$[0.420958, -1.40852]^T$	0.6	0.5	0.3	Reject

$\varepsilon = \varepsilon_a + \varepsilon_r |y_n|$ mit $\tilde{p} = \min(p, \hat{p}) + 1$ (Ordnung des Primärverfahrens)

7.2.6 Stabilität von expliziten Methoden S27

Globaler relativer Fehler darf nicht divergieren, d.h. muss beschränkt sein. Da die Analyse mit zu untersuchender ODE wegen fehlender Kenntnis über die Lösung sehr schwierig ist, wird mit allgemeiner ODE gerechnet, dem Dahlquist Modell:

$$y' = Ay \quad y(0) = 1 \qquad \text{mit } A = \Re\{A\} + j\Im\{A\} \in \mathbb{C}$$

Dessen Lösung ist $y = e^{\Re\{A\}x} (\cos(\Im\{A\}x + j \sin(\Im\{A\}x))$, also eine Schwingung mit exponentieller Amplitude $e^{\Re\{A\}x}$ und Frequenz $\Im\{A\}$.
Mit der zu untersuchenden Methode kann eine *lineare Stabilitätsfunktion* $F(z)$ berechnet werden.
Beispiel Anhand der Heun-Methode (RK-2) wird die Untersuchung gezeigt:

$$k_1 = Ay_k$$
$$k_2 = A(y_k + 1hk_1) = A(y_k + Ahy_k) \qquad \Longrightarrow$$

$$y_0 = y(x_0) = y(0) = 1$$
$$y_{k+1} = y_k \underbrace{\left(1 + hA + (hA)^2 \frac{1}{2}\right)}_{F(hA)=F(z), \quad z=hA \in \mathbb{C}}$$

Die lineare Stabilitätsfunktion kann jetzt auch so eingesetzt werden:

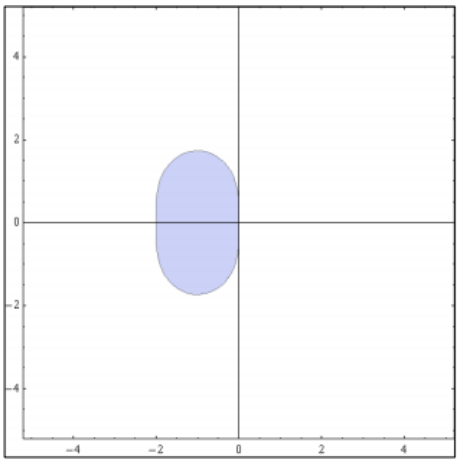
$$y_{k+1} = \Big(|F(z)| \exp\left(j \arg(F(z))\right) \Big) y_k$$

BY-NC-SA

Jetzt können drei Fälle unterschieden werden:

- $\Re\{A\} < 0$: Die Amplitude der ODE wird exponentiell kleiner. Die Stabilitätsbedingung verlangt, dass die approximierten Werte y_k $(k = 0, 1, \ldots)$ ebenfalls exponentiell gedämpft werden. Daher und aus $y_{k+1} \propto |F(z)|$ folgt, dass $|F(z)| < 1$.
- $\Re\{A\} = 0$: Die Amplitude der ODE ist konstant 1. Die Stabilitätsbedingung verlangt, dass die approximierten Werte y_k $(k = 0, 1, \ldots)$ ebenfalls konstant sind. Daher und aus $y_{k+1} \propto |F(z)|$ folgt, dass $|F(z)| = 1$.
- $\Re\{A\} > 0$: Die Amplitude der ODE wird exponentiell grösser. Die Stabilitätsbedingung verlangt, dass die approximierten Werte y_k $(k = 0, 1, \ldots)$ ebenfalls exponentiell wachsen. Daher und aus $y_{k+1} \propto |F(z)|$ folgt, dass $|F(z)| > 1$.

Die Fälle 2 und 3 bereiten keine Probleme (Bedingungen sind immer erfüllt), nur die erste Fall ist kritisch.



Fall 1 der Stabilitätsbedingung ($\Re\{A\} < 0$):
$1 > |F(z)| = |1 + hA + \frac{1}{2}(hA)^2| \Rightarrow -2 < h\Re\{A\} < 0$.
Fälle 2 und 3 sind in Abb. nicht berücksichtigt.

Rekursive Formel S29 Für das Stabilitätspolynom $F(z) = 1 + b_1 k_1(z) + \ldots + b_s k_s(z)$ existieren rekursive Formeln:
$k_1(z) = z, \qquad k_{j+1}(z) = z(1 + a_{j+1,1} k_1(z) + a_{j+1,2} k_2(z) + \ldots + a_{j+1,j} k_j(z))$

Schwäche der expliziten Methoden: Stabilität!

7.2.7 Stiffness (Steifigkeit) S33

Definition Stiffness = sehr schwierig (gemäss Wikipedia). Ungefähre Idee: Die expliziten Methoden werden zu inakzeptabel kleinen Step-Sizes gezwungen (Computational complexity). Dies trotz eigentlichen „smoothen“ Oberflächen der Funktionen, die intuitiv nicht sehr schwierig berechenbar sein sollten.

Wenn Stabilitätsverletzungen (Stiffness) detektiert werden, soll zu impliziten Verfahren (Blackbox-Solver) gewechselt werden, um dann mit non-Stiffness-Detektierung zurück zu expliziten Verfahren umzustellen.

Stiffness ist abhängig von:

- Konkrete ODE
- Konkrete Schrittweite h
- Konkretes Verfahren (bspw. RK-4)

Stiffness Detektierung Voraussetzung: $c_{s-1} = c_s = 1$. Durch Testen von $\left|h\frac{\partial f(x,y)}{\partial y}\right|$ auf die absolute Grenze der Stabilitätsregion kann Stiffness detektiert werden. Stiffness liegt vor, falls $|h\tilde{\lambda}|$ ausserhalb oder an der Grenze des Stabilitätsbereiches liegt.
Beispiel explizite Runge-Kutta:

$$k_{s-1} = f(x + c_{s-1}h, \underbrace{y + h\sum_{j=1}^{s} a_{s-1,j}k_j}_{g_{s-1}}) \qquad k_s = f(x + c_s h, \underbrace{y + h\sum_{j=1}^{s} a_{s,j}k_j}_{g_s}) \qquad \Rightarrow \qquad \tilde{\lambda} = \frac{\|k_s - k_{s-1}\|}{\|g_s - g_{s-1}\|}$$

$\tilde{\lambda}$ ist eine Schätzung für $f_y = \frac{\partial}{\partial y} f(x, y)$ z.B. $y' = x^4 - 25y^4 = f(x, y)$ und übernimmt die Rolle von $\Re\{A\}$ bei der Stabilitätsanalyse.

7.3 Systeme von ODEs und ODEs höherer Ordnungen S37

Systeme von ODEs werden in Vektor-Notation aufgeschrieben und wie üblich berechnet. That's all.

BY-NC-SA

8 Ableitungstabelle

	$F(x)$	$F'(x)$
Addition	$f(x) \pm g(x)$	$f'(x) \pm g'(x)$
Linearity	$af(x)$	$af'(x)$
Product Rule	$f(x)g(x)$	$f'(x)g(x) + f(x)g'(x)$
Quotient Rule	$\frac{f(x)}{g(x)}$	$\frac{f'(x)g(x)-f(x)g'(x)}{(g(x))^2}$
Chain Rule	$f(g(x))$	$f'(g(x)) \cdot g'(x)$
	$f^{-1}(x)$	$\frac{1}{f'(f^{-1}(x))}$
Basic functions	x^n for any real n	nx^{n-1}
	e^x	e^x
	a^x $(a > 0)$	$(\ln a)a^x$
	$\ln x$	$\frac{1}{x}$
Trig functions	$\sin x$	$\cos x$
	$\cos x$	$-\sin x$
	$\tan x$	$\frac{1}{\cos^2 x} = 1 + \tan^2 x$
	$\arctan x = \tan^{-1} x$	$\frac{1}{1+x^2}$
	$\arcsin x = \sin^{-1} x$	$\frac{1}{\sqrt{1-x^2}}$
Hyperbolic Trig	$\sinh x$	$\cosh x$
	$\cosh x$	$\sinh x$
	$\tanh x$	$\frac{1}{\cosh^2 x}$
	$\sinh^{-1} x$	$\frac{1}{\sqrt{1+x^2}}$
	$\tanh^{-1} x$	$\frac{1}{1-x^2}$

9 Übersicht der Polynome

Name	Formel	Beispiel	Referenz
Newton	$\pi_n(x) = \prod_{i=1}^{n}(x - x_{i-1})$	$\pi_2 = (x - x_0)(x - x_1)$	1.3, S. 1
Bernstein	$B_{i,n}(x) = \binom{n}{i}(1-x)^{n-i}x^i$	$B_{1,2} = 2x(1-x)$	4.2.1, S. 8
Monomials	x^n	x^2	5.1.4, S. 11
Orthogonale	$p_{k,N}(t) = \sum_{i=0}^{k}(-1)^i\binom{k}{i}\binom{k+i}{i}\frac{t^{(i)}}{N^{(i)}}$	$P_1(x) = 1 - \frac{x-3}{2}$	5.1.5, S. 11
Chebyshev	$T_n(x) = \cos(n \arccos(x))$	$T_2 = 2x^2 - 1$	5.1.6, S. 12

BY-NC-SA

10 Integrationstabelle

Linearity	$\int af(x) + bg(x)\,dx = a\int f(x)\,dx + b\int g(x)\,dx$
Substitution	$\int f(w(x))w'(x)\,dx = \int f(w)\,dw$
Integration by parts	$\int u(x)v'(x)\,dx = u(x)v(x) - \int u'(x)v(x)\,dx$

Basic Functions

$$\int x^n\,dx = \frac{x^{n+1}}{n+1} + C \qquad \int \frac{1}{x}\,dx = \ln|x| + C$$
$$\int e^{ax}\,dx = \frac{1}{a}e^x + C \qquad \int a^x\,dx = \frac{a^x}{\ln a} + C$$

Trigonometric functions

$$\int \sin x\,dx = -\cos x + C \qquad \int \cos x\,dx = \sin x + C$$
$$\int \frac{1}{\cos^2 x}\,dx = \tan x + C \qquad \int \tan x\,dx = -\ln|\cos x| + C$$
$$\int \cot x\,dx = \ln|\sin x| + C$$

Hyperbolic Trig functions

$$\int \sinh x\,dx = \cosh x + C \qquad \int \cosh x\,dx = \sinh x + C$$
$$\int \tanh x\,dx = \ln(\cosh x) + C \qquad \int \coth x\,dx = \ln|\sinh x| + C$$

Functions with $a^2 \pm x^2$

$$\int \frac{dx}{\sqrt{a^2 - x^2}} = \sin^{-1}\left(\frac{x}{a}\right) + C \qquad \int \frac{dx}{a^2 + x^2} = \frac{1}{a}\tan^{-1}\left(\frac{x}{a}\right) + C$$
$$\int \frac{dx}{a^2 - x^2} = \frac{1}{2a}\ln\left|\frac{x+a}{x-a}\right| + C$$
$$\int \frac{dx}{\sqrt{x^2 - a^2}} = \cosh^{-1}\left(\frac{x}{a}\right) + C \qquad \int \frac{dx}{\sqrt{x^2 + a^2}} = \sinh^{-1}\left(\frac{x}{a}\right) + C$$

Inverse Functions

$$\int \ln x\,dx = x\ln x - x + C \qquad \int \arcsin x\,dx = x\arcsin x + \sqrt{1 - x^2} + C$$
$$\int \arctan x = x\arctan x - \frac{1}{2}\ln(1 + x^2) + C$$

BY-NC-SA