Differentialgleichungen

Eine gewöhnliche Differentialgleichung ist eine Gleichung, welche nur nach einer Variable abgeleitet wird:

$$\frac{\partial y}{\partial t} = f(t, y(t))$$

Im oben Beispiel ist es eine gewöhnliche Differentialgleichung 1. Ordnung, da nur einmal Abgeleitet wurde.

Genereller ausgedrückt, folgendes ist eine gewöhnliche Differentialgleichung n-ter Ordnung:

$$y^{(n)}(x) = f(x, y(x), y'(x), \dots, y^{(n-1)}(x))$$

Eine allgemeine Lösung für eine Differentialgleichung n-ter Ordnung hat n unabhängige Parameter (von den Integrationskonstanten).

Differentialgleichungen, welche die folgende Form haben

$$\frac{\partial n}{\partial t} = -\lambda n$$

haben die Lösung $n(t)$:

$$n(t) = n_0 e^{-\lambda t}$$

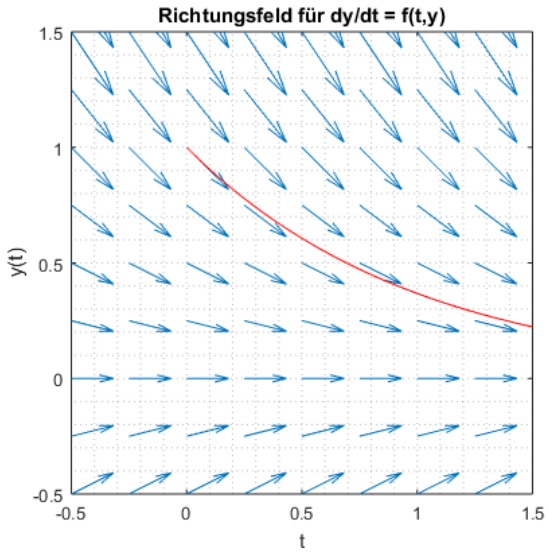
Anfangswertproblem

Bei einem Anfangswertproblem wird, zusätzlich zu der Gleichung, den Funktionswert bei x_0, wie auch den Wert für jede benützte Ableitung bei dem selben Wert x_0.

Als Beispiel für folgende Funktion s wird C_1 und C_2 benötigt, damit ein Resultat berechnet werden kann. Es wird also $s(t = 0)$ und $s'(t = 0)$ benötigt, um das Anfangswertproblem zu lösen.

$$s'' = g$$
$$s(t) = \frac{1}{2}gt^2 + C_1 t + C_2$$
$$s(t=0) = C_2$$
$$s'(t=0) = v(t=0) = C_1$$

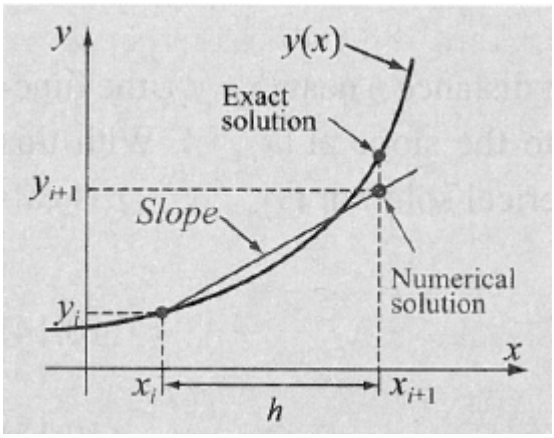
Richtungsfelder



Ein Richtungsfeld stellt die Steigung als Pfeile dar. Dafür wurde in diesem Beispiel alle y' für alle Punkte berechnet und eingezeichnet.

Eulerverfahren

Klassisch

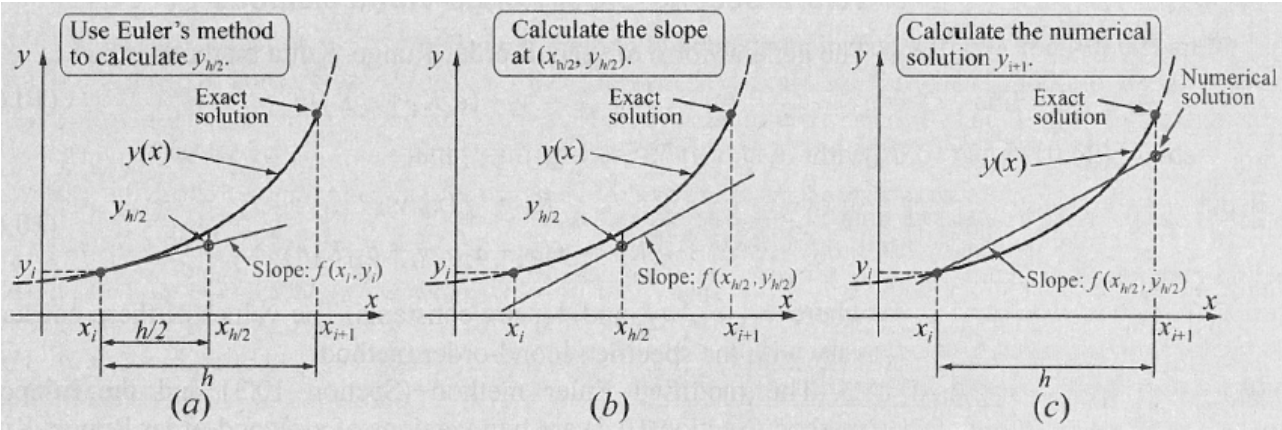


Um eine Lösung für eine Differentialgleichung mit einem Richtungsfeld zu finden, kann eine Schrittweite h definiert werden. Jeder Punkt (x_i, y_i) soll nun den Pfeilen im Feld folgen. Dies kann folgendermassen für eine Differentialgleichung $y' = f(x, y)$ erledigt werden:

$$\begin{aligned} x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + y' \cdot h \\ &= y_i + f(x_i, y_i) \cdot h \end{aligned}$$

Zusätzlich wird auch noch ein Startpunkt (x_0, y_0) benötigt.

Mittelpunkt



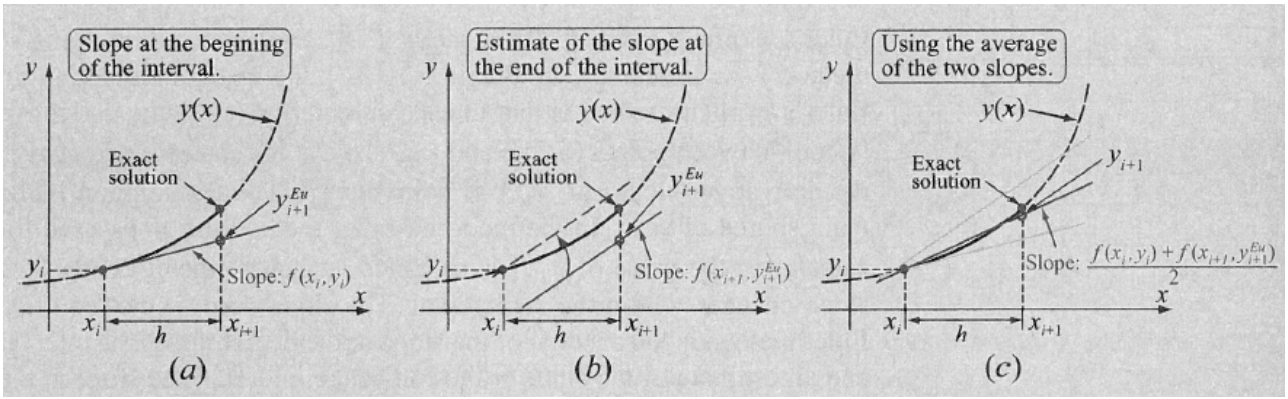
Im Vergleich zum Eulerverfahren, wo die Steigung beim Punkt (x_i, y_i) berechnet wird, wird beim Mittelpunkt-Verfahren die Steigung bei $(x_i + \frac{h}{2}, y_i + \frac{h}{2})$ berechnet.

Dafür muss aber der Punkt $(x_i + \frac{h}{2}, y_i + \frac{h}{2})$ zuerst berechnet werden. Daher ergibt sich folgendes:

$$\begin{aligned} x_{h/2} &= x_i + \frac{h}{2} \\ y_{h/2} &= y_i + \frac{h}{2} \cdot f(x_i, y_i) \end{aligned}$$

$$\begin{aligned} x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + f(x_{h/2}, y_{h/2}) \cdot h \end{aligned}$$

Modifiziert



Beim modifizierten Verfahren wird zuerst die Steigung bei (x_i, y_i) und bei (x_{i+1}, y_{i+1}) berechnet. Danach wird der nächste Punkt mit dem Mittel zwischen den beiden Steigungen den nächsten Punkt berechnet.

Algorithmus: Modifiziertes Euler-Verfahren für $y' = f(x,y)$ mit $y(a) = y_0$.

- Führe das klassische Euler-Verfahren durch und speichere die erste Tangengentsteigung in der Variable k_1:

$$\begin{aligned} x_{i+1} &= x_i + h \\ y_{i+1}^{Euler} &= y_i + h \cdot f(x_i, y_i) \\ k_1 &= f(x_i, y_i) \end{aligned}$$

- Berechne die zweite Tangentensteigung am Punkt $(x_{i+1}, y_{i+1}^{Euler})$ und speichere sie in k_2:

$$k_2 = f(x_{i+1}, y_{i+1}^{Euler})$$

- Bilde den Durchschnitt der Steigungen $(k_1 + k_2)/2$ und mache einen Schritt h ausgehend vom ursprünglichen Punkt (x_i, y_i) zur Berechnung der Näherung (x_{i+1}, y_{i+1}):

$$\begin{aligned} x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + h \cdot \frac{(k_1 + k_2)}{2} \end{aligned}$$

- Wiederhole diese Schritte ausgehend von $x_0 = a$ für $x_i = a + ih$ mit $i = 0, \ldots, n-1$

Fehler

Der **lokaler** Fehler ist definiert als:

$$\varphi(x_i, h) := y(x_{i+1}) - y_{i+1}$$

Wenn der lokaler Fehler folgendermassen schreiben kann, dann hat es die **Konsistenzordnung** p:

$$\varphi(x_i, h) \leq C \cdot h^{p+1}$$

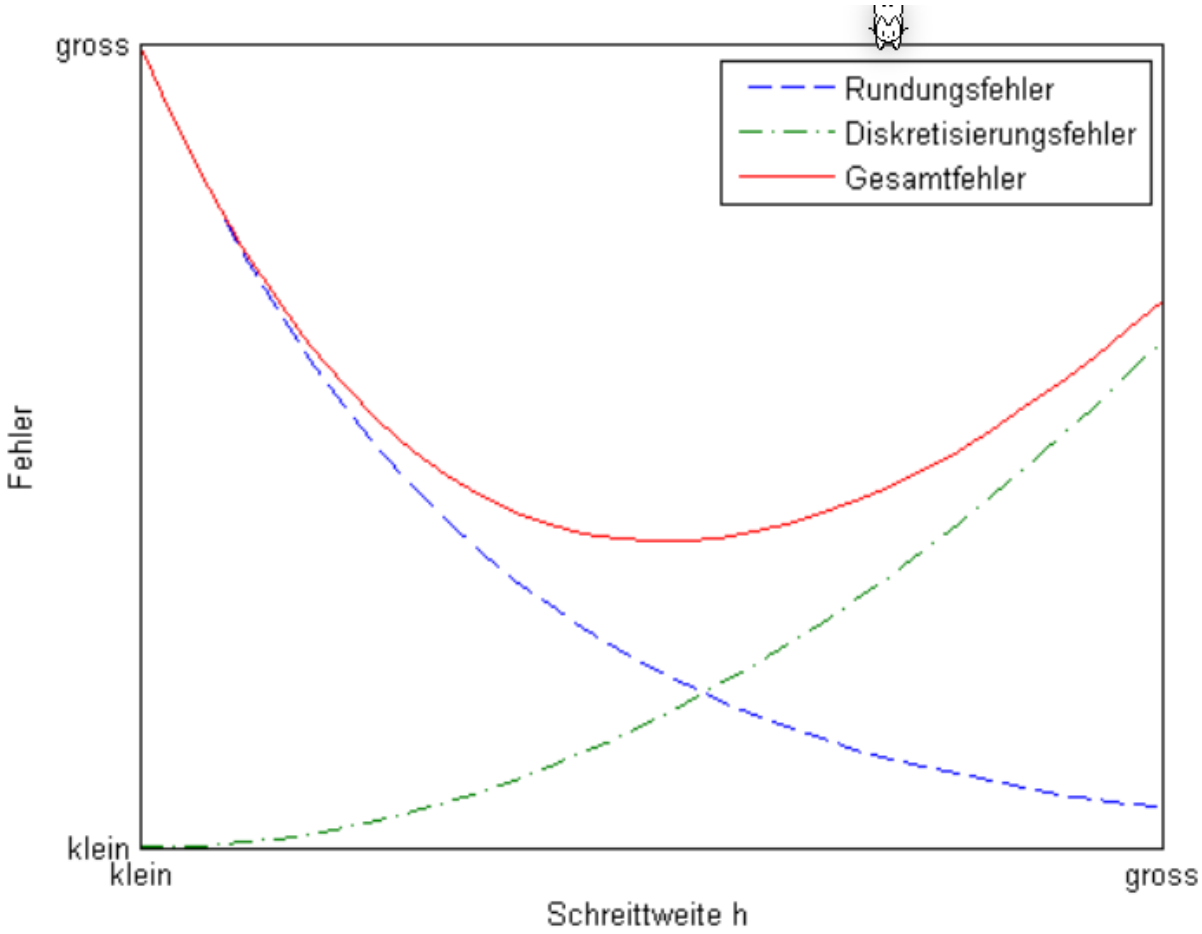
Ebenfalls gibt es ein **globalen** Fehler, welcher definiert ist als:

$$y(x_n) - y_n$$

Wenn der globalen Fehler folgendermassen schreiben kann, dann hat es folgende **Konvergenzordnung** p:

$$|y(x_n) - y_n| \leq C \cdot h^p$$

Wie auch an den Formeln von der Konsistenzordnung und Konvergenzordnung zu sehen ist, hängt dieser Fehler von der Schrittweite h ab.



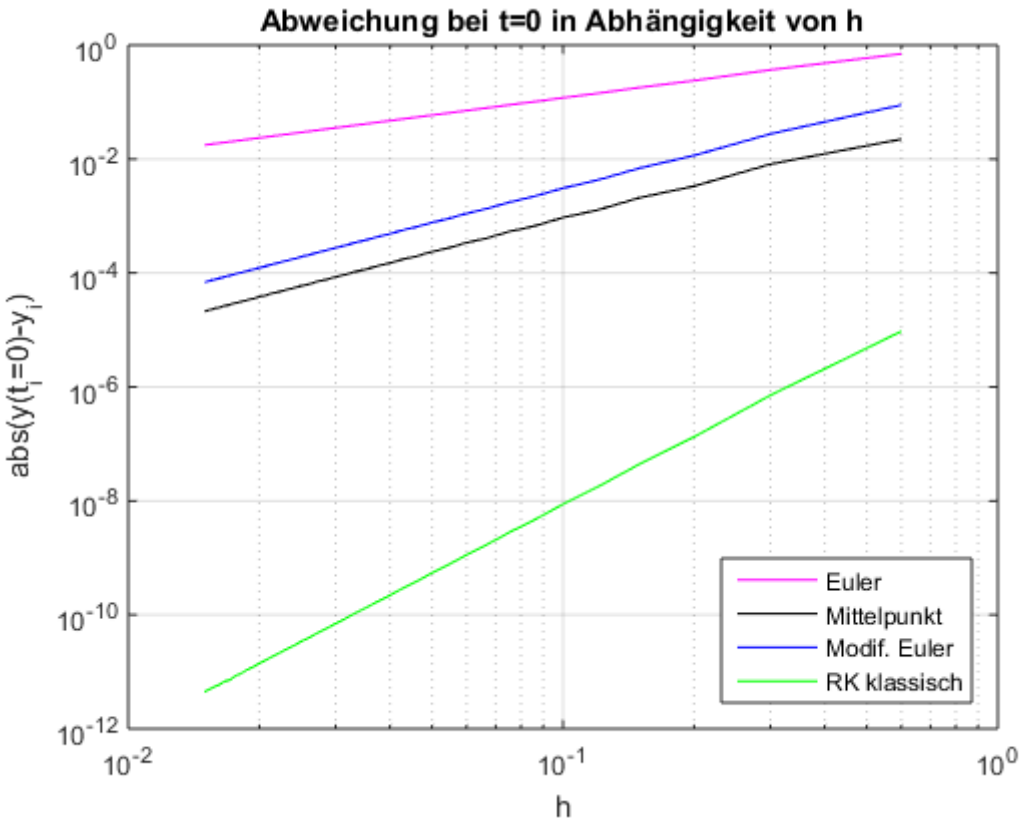
Es ist interesant ein Verfahren mit der Konvergenzordnung $p \geq 1$ und $h < 1$, da dann $C \cdot h^p$ gegen 0 strebt.

Für das Eulerverfahren gilt folgenden lokalen Fehler:

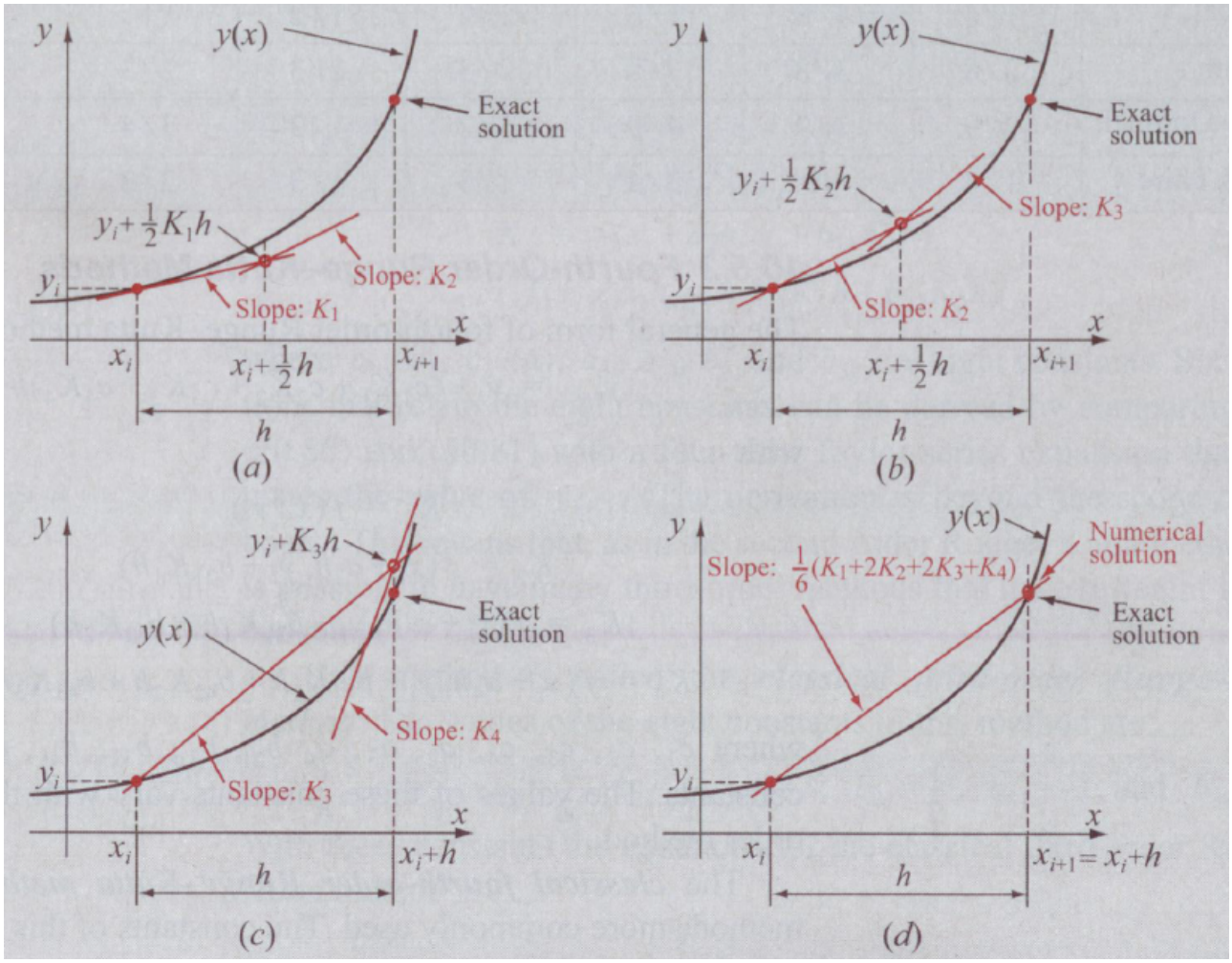
$$\varphi(x_n, h) = \frac{h^2}{2} y''(z) \qquad \text{, wobei } z \in [x_n, x_n + h]$$

Das Mittelpunkt und modifizierte Eulerverfahren haben eine Konsistenz- und Konvergenzordnung $p = 2$.

In der folgenden Abbildung ist der lokale Fehler für diverse Verfahren auf einem log-log Plot:



Runge-Kutta Verfahren



Im Runge-Kutta-Verfahren wird zuerst die Steigung k_1 bei x_i berechnet, dann k_2 in der Mitte zwischen x_i und x_{i+1}, k_3 ist ebenfalls beim Mittelpunkt, aber mit der Steigung k_2. Zuletzt wird k_4 am Punkt x_{i+1} berechnet.

Algorithmus: klassisches vierstufiges Runge-Kutta Verfahren [1]

- Gegeben sei für $x \in [a,b]$ das Anfangswertproblem

$$y' = f(x,y) \quad \text{mit } y(a) = y_0.$$

- Das klassische Runge-Kutta zur numerischen Lösung lautet

$$\begin{aligned}
k_1 &= f(x_i, y_i) \\
k_2 &= f(x_i + \frac{h}{2}, y_i + \frac{h}{2} k_1) \\
k_3 &= f(x_i + \frac{h}{2}, y_i + \frac{h}{2} k_2) \\
k_4 &= f(x_i + h, y_i + h k_3) \\
x_{i+1} &= x_i + h \\
y_{i+1} &= y_i + h \cdot \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}$$

wobei $x_0 = a$, $x_i = a + ih$ für $i = 0, \ldots, n-1$ $(n \in \mathbb{N})$ und $h = \frac{b-a}{n}$.

Die Konsistenz- und Konvergenzordnung von Runge-Kutta ist $p = 4$.

Allgemeines s-stufiges Runge-Kutta-Verfahren

Das allgemeine s-stufige Runge-Kutta-Verfahren:

$$k_n = f\left(x_i + c_n h, y + h \sum_{m=1}^{n-1} a_{nm} k_m\right) \qquad \text{für } n = 1, \ldots, s$$

$$y_{i+1} = y_i + h \sum_{n=1}^{s} b_n k_n$$

Dabei ist $s \in \mathbb{N}$ die Stufenzahl und a_{nm}, b_n und c_n sind Konstante.

$$\begin{array}{c|cccc|c}
c_1 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & & & & & \\
c_n & a_{n1} & a_{n2} & \ldots & a_{n,n-1} & \\
\vdots & & & & & \\
c_s & a_{s1} & a_{s2} & \ldots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \ldots & b_{s-1} & b_s
\end{array}$$

- Euler-Verfahren: $s = 1$

0	
	1

- Mittelpunkt-Verfahren: $s = 2$

0		
0.5	0.5	
	0	1

- Modifiziertes Euler-Verfahren: $s = 2$

0		
1	1	
	0.5	0.5

- Klassisches Runge-Kutta Verfahren: $s = 4$

0				
0.5	0.5			
0.5	0	0.5		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

Differentialgleichung-System

Um ein Differentialgleichung-System zu lösen, kann $y(x)$ als vektorwertige Funktion geschrieben werden.

Das Euler-Verfahren kann folgendermassen für Vektoren angepasst werden:

$$x_{i+1} = x_i + h$$
$$\vec{y}_{i+1} = \vec{y}_i + \vec{f}(x_i, y_i) \cdot h$$

Oben ist es mit dem klassischen Eulerverfahren beschrieben. Dies kann aber mit allen Eulerverfahren gelöst werden.

Beispiel

Das folgende Beispiel kommt aus dem nächsten Unterkapitel "Differentialgleichung k-ter Ordnung zu DGL-System".

$$\begin{aligned} z_1' &= z_2 \\ z_2' &= z_3 \\ z_3' &= 10e^{-x} - 5z_3 - 8z_2 - 6z_3 \end{aligned}$$

Zu dem gelten folgende Anfangswerte:

$$\vec{z}(0) = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix}$$

Nun kommen die Iterationen:

- $i = 0$: für den ersten Schritt erhalten wir

$$\begin{aligned} \boldsymbol{f}(x_0, \boldsymbol{z}^{(0)}) &= \begin{pmatrix} z_2^{(0)} \\ z_3^{(0)} \\ 10e^{-x_0} - 5z_3^{(0)} - 8z_2^{(0)} - 6z_1^{(0)} \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 0 \\ 10e^{-0} - 5 \cdot 0 - 8 \cdot 0 - 6 \cdot 2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -2 \end{pmatrix} \\ \boldsymbol{z}^{(1)} &= \boldsymbol{z}^{(0)} + h \cdot \boldsymbol{f}(x_0, \boldsymbol{z}^{(0)}) = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} + 0.5 \cdot \begin{pmatrix} 0 \\ 0 \\ -2 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ -1 \end{pmatrix} \\ x_1 &= x_0 + h = 0.5 \end{aligned}$$

- $i = 1$: für den zweiten Schritt erhalten wir

$$\begin{aligned} \mathrm{f}(x_1, \mathrm{z}^{(1)}) &= \begin{pmatrix} z_2^{(1)} \\ z_3^{(1)} \\ 10e^{-x_1} - 5z_3^{(1)} - 8z_2^{(1)} - 6z_1^{(1)} \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ -1 \\ 10e^{-0.5} - 5 \cdot (-1) - 8 \cdot 0 - 6 \cdot 2 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \\ -0.9347 \end{pmatrix} \\ \boldsymbol{z}^{(2)} &= \boldsymbol{z}^{(1)} + h \cdot \boldsymbol{f}(x_1, \boldsymbol{z}^{(1)}) = \begin{pmatrix} 2 \\ 0 \\ -1 \end{pmatrix} + 0.5 \cdot \begin{pmatrix} 0 \\ -1 \\ -0.9347 \end{pmatrix} \\ &= \begin{pmatrix} 2 \\ -0.5 \\ -1.4673 \end{pmatrix} \\ x_2 &= x_1 + h = 1.0 \end{aligned}$$

Differentialgleichung k-ter Ordnung zu DGL-System

Um eine Differentialgleichung mit Ableitungen höher als erster Ableitungen zu lösen gibt es einen Trick:

$$y''' + 5y'' + 8y' + 6y = 10e^{-x}$$

1. Nachh der höchsten Ableitung umformen:
   $y''' = 10e^{-x} - 5y'' - 8y' - 6y$

2. Alle Ableitungen von y tiefer als die höchste Ableitungen durch z_i ersetzen:
   $z_1 = y, z_2 = y', z_3 = y''$
3. Und in der Gleichung einsetzen
   $y''' = 10e^{-x} - 5y'' - 8y' - 6y \Rightarrow z_3' = y''' = 10e^{-x} - 5z_3 - 8z_2 - 6z_3$
4. Es sind nun drei Gleichungen:
   $z_1' = y' = z_2$
   $z_2' = y'' = z_3$
   $z_3' = 10e^{-x} - 5z_3 - 8z_2 - 6z_3$
5. In diesem Fall können sie auch vektoriel geschrieben werden:
   $$\begin{pmatrix} z_2 \\ z_3 \\ 10e^{-x} - 5z_3 - 8z_2 - 6z_3 \end{pmatrix} = \begin{pmatrix} z_1' \\ z_2' \\ z_3' \end{pmatrix}$$
   Mit der Start-Bedingungen: $\vec{z}(0) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

Stabilität

Wie stabil eine Lösung einer DGL ist hängt von dem benutzten Verfahren, der Schrittbreite und dem spezifischen Anfangsproblem ab.

Definition 8.6: Stabilitätsfunktion / Stabilitätsintervall

- Kann bei der Anwendung eines Verfahrens auf die DGL $y' = -\alpha y$ die numerische Lösung in der Form

  $$y_{i+1} = g(h\alpha) \cdot y_i$$

  geschrieben werden, so nennt man $g(z)$ die Stabilitätsfunktion des Verfahrens (mit $z = h\alpha$).
- Das offene Intervall $z \in (0, \alpha)$, in dem $\mid g(z) \mid < 1$ gilt, bezeichnet man als das Stabilitätsintervall des Verfahrens.

Interpolation

Gegeben sind $n+1$ Stützpunkte/Wertpaare (x_i, y_i), wobei $x_i \neq x_i$ für $i \neq j$ gelten muss. Gesucht ist nun eine stetige Funktion g mit der Eigenschaft $g(x_i) = y_i$ für alle $i = 0, \ldots, n$

Polynominterpolation

Wenn $n+1$ Stützpunkte gegen sind, kann das Polynom
$P_n(x) = a_0 + a_1 x + a_2 + x^2 + \ldots + a_n x^n$

Wenn x ein Vektor ist, kann auch eine Vandermonde-Matrix gebildet werden:

$$
\begin{gathered}
a_0 + a_1 x_0 + a_2 + x_0^2 + \ldots + a_n x_0^n \\
a_0 + a_1 x_1 + a_2 + x_1^2 + \ldots + a_n x_1^n \\
\ldots \\
a_0 + a_1 x_n + a_2 + x_n^2 + \ldots + a_n x_n^n
\end{gathered}
$$

$$
\begin{pmatrix}
1 & x_0 & x_0^2 & \ldots & x_0^n \\
1 & x_1 & x_1^2 & \ldots & x_1^n \\
 & & & \ldots & \\
1 & x_n & x_n^2 & \ldots & x_n^n
\end{pmatrix}
\cdot
\begin{pmatrix}
a_0 \\ a_1 \\ \ldots \\ a_n
\end{pmatrix}
=
\begin{pmatrix}
y_0 \\ y_1 \\ \ldots \\ y_n
\end{pmatrix}
$$

Diese Rechnung ist allerdings oft schlecht Konditioniert und wird für $n > 20$ Stützpunkte instabil. Ein möglichen Ersatz ist das Lagrange Polynom

Lagrange Interpolation

Das Lagrange Polynom kann für n Stützpunkte berechnet werden und ergibt ein Polynom mit dem Rang $n-1$.

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

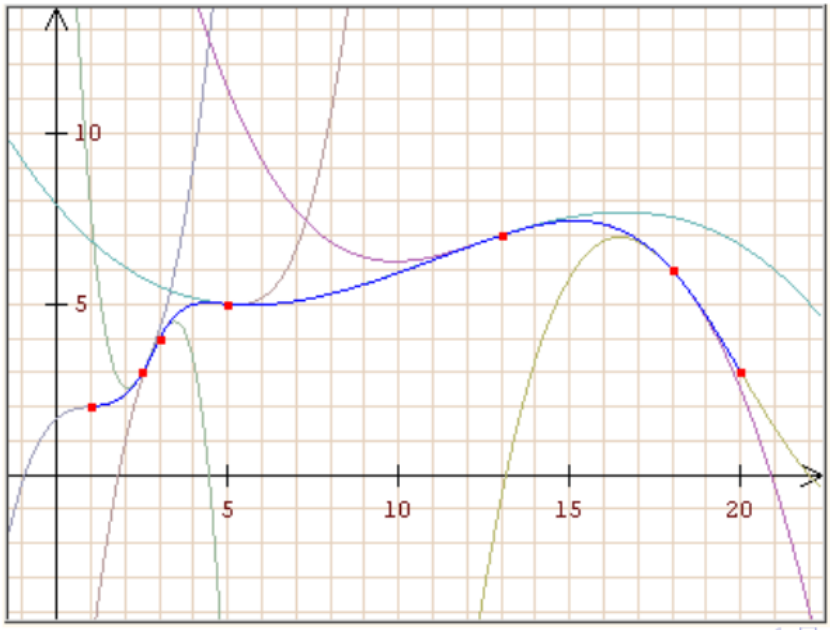
Der maximale absoluten Fehler der dabei entstehen kann ist:

$$
|f(x) - P_n(x)| \leq \frac{|(x-x_0)(x-x_1)\ldots(x-x_n)|}{(n+1)!} \cdot (\max_{x0 \leq \xi \leq x_n} |f^{(x+1)}(\xi)|)
$$

Als Bemerkung $f^{(x+1)}$ ist die $(x+1)$-te Ableitung

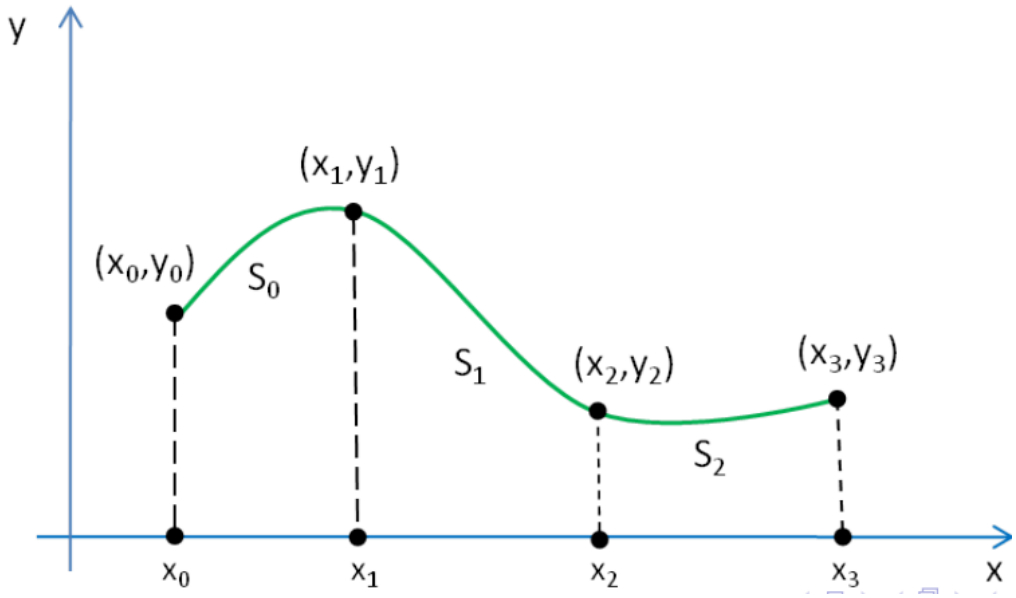
Da für die Fehlerberechnung die eigentliche Funktion f benötigt wird, ist dies recht nutzlos.

Spline Interpolation



Es wird für jedes Intervall $[x_i, x_{i+1}]$ (für $i = 0, 1, 2, , \ldots, n-1$) wird ein Polynom s_i angesetzt. Das Polynom muss folgende Gleichungen erfüllen:

- Es muss durch alle Punkte im Intervall $[x_i, x_{i+1}]$ gehen
  $s_i(x_i) = y, s_i(x_{i+1}) = y_{i+1}, \ldots$
- Der Übergang zwischen den Polynomen muss stetig sein
  $s_i(x_{i+1}) = s_{i+1}(x_{i+1})$
- Es darf kein Knick beinhalten
  $s_i'(x_{i+1}) = s_{i+1}'(x_{i+1})$
- Die Krümmung von zwei Splines soll auch gleich sein
  $s_i''(x_{i+1}) = s_{i+1}''(x_{i+1})$



Um die Spline von oben zu berechnen, können nun folgende Polynome definiert werden:

$$
\begin{aligned}
S_0 &= a_0 + b_0(x - x_0) + c_0(x - x_0)^2 + d_0(x - x_0)^3 \qquad &, x \in [x_0, x_1] \\
S_1 &= a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3 \qquad &, x \in [x_1, x_2] \\
S_2 &= a_2 + b_2(x - x_2) + c_2(x - x_2)^2 + d_2(x - x_2)^3 \qquad &, x \in [x_2, x_3]
\end{aligned}
$$

Aus diesen können nun folgendes Gleichungssytem aufgestellt werden:

$$
\begin{aligned}
S_0(x_0) &= y_0 \\
S_1(x_1) &= y_1 \\
S_2(x_2) &= y_2 \\
S_2(x_3) &= y_3
\end{aligned}
$$

$$
\begin{aligned}
S_0(x_1) &= S_1(x_1) \\
S_1(x_2) &= S_2(x_2)
\end{aligned}
$$

$$
\begin{aligned}
S_0'(x_1) &= S_1'(x_1) \\
S_1'(x_2) &= S_2'(x_2)
\end{aligned}
$$

$$
\begin{aligned}
S_0''(x_1) &= S_1''(x_1) \\
S_1'0(x_2) &= S_2''(x_2)
\end{aligned}
$$

Dies sind aber "nur" 10 Gleichungen, nicht die benötigten 12. Daher gibt es noch zusätzliche Bediungen:

- natürliche kubische Splinefunktion
  $S_0''(x_0) = 0, S_2(x_3)'' = 0$
- peridodische kubische Splinefunktion
  $S_0'(x0) = S_2'(x_3), S_0''(x0) = S_2''(x_3)$
- kubische Spliefunktion (mit not-a-knot Bedinungen)
  $S_0'''(x_1) = S_1'''(x_1), S_1'''(x_2) = S_2'''(x_2)$

Algorithmus

Für $n + 1$ Stützpunkte werden n Gleichungen nach der Form
$S_i = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, x \in [x_i, x_{i+1}]$ gesucht. Dafür kann folgender Algorithmus für jedes S_i angewendet werden:

1. Wenn die natürliche kubische Splinefunktion gesucht ist, wird c auf 0 gesetzt damit die zweite Ableitung 0 ergibt
   $c_0 = 0, c_n = 0$
2. Für jedes Polynom S_i

a. $a_i = y_i$

b. Die Breite des Intervalles
$h_i = x_{i+1} - x_i$

3. c_i bestimmen

4. b_i und d_i für jedes S_i bestimmen

a. $b_i = \frac{y_{i+1}-y_i}{h_i} - \frac{h_i}{3}(c_{i+1} + 2c_i)$

b. $d_i = \frac{1}{3h_i}(c_{i+1} - c_i)$

Für das Beispiel ...:

i	**0**	**1**	**2**	**3**
x_i	0	1	2	3
y_i	2	1	2	2
a_i	2	1	2	-
h_i	1	1	1	-
c_i	0	?	?	0

Um c_1 und c_2 zu finden kann folgendes Gleichungssystem gelöst werden:

$$A = \begin{pmatrix} 2(h_0 + g_1) & h_1 \\ h_1 & 2(h_1 + h_2) \end{pmatrix}$$
$$A \cdot \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = ()$$

Lineare Ausgleichunsrechnung

Es wird eine Funktion gesucht, in der Form:

$$f(x) = \lambda_1 f_1(x) + \ldots + \lambda_m f_m(x)$$

Ein mögliches Beispiel wäre: $f(x) = \lambda_1 \cdot \underbrace{1}_{f_1(x)} + \lambda_2 \cdot \underbrace{x}_{f_2(x)} + \lambda_3 \cdot \underbrace{x^2}_{f_3(x)}$

Um nun die λs zu finden, damit $f(x)$ Datenpunkte nachgeht, muss der Fehler $E(f)$ zu den Datenpunkten minimieren:

$$E(f) = w \cdot ||y_f - (x)||_2^2 = \sum_{i=1}^{n} w_i \cdot (y_i - f(x_i))^2$$

Mit w kann ein Punkt stärker oder schwächer gewichtet werden

Um dies zu minimieren, wird die Ableitung von $E(f) = 0$ gesetzt:

$$\begin{aligned}
0 &\overset{!}{=} \sum_{i=1}^{n} (y_i - (ax_i + b)) \cdot (x_i) = \sum_{i=1}^{n} \left(x_i y_i - a x_i^2 - b x_i\right) \\
&= \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} a x_i^2 - \sum_{i=1}^{n} b x_i = \sum_{i=1}^{n} x_i y_i - a \sum_{i=1}^{n} x_i^2 - b \sum_{i=1}^{n} x_i \\
0 &\overset{!}{=} \sum_{i=1}^{n} (y_i - (ax_i + b)) = \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} a x_i - \sum_{i=1}^{n} b \\
&= \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} a x_i - \sum_{i=1}^{n} b = \sum_{i=1}^{n} y_i - a \sum_{i=1}^{n} x_i - b \sum_{i=1}^{n} 1
\end{aligned}$$

Aus dem folgt:

$$\begin{aligned}
a \sum_{i=1}^{n} x_i^2 + b \sum_{i=1}^{n} x_i &= \sum_{i=1}^{n} x_i y_i \\
a \sum_{i=1}^{n} x_i + b \underbrace{\sum_{i=1}^{n} 1}_{=n} &= \sum_{i=1}^{n} y_i
\end{aligned}$$

$$\begin{pmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & n \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{pmatrix}$$

Dies funktioniert allerdings nur für eine Gerade. Die selbe Methode kann aber auch für höhere Polynomen verwendet werden:

$$E(f) = ||\vec{y} - f(\vec{x})||_2^2 = \sum_{i=1}^{n} (y_i - f(x_i))^2 = \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{m} \lambda_j f_j(x_i) \right)^2 = ||\vec{y} - A\lambda||_2^2$$

Mit folgender Matrix A:

$$\boldsymbol{A} = \begin{pmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_m(x_1) \\ f_1(x_2) & f_2(x_2) & \cdots & f_m(x_2) \\ \vdots & \vdots & \vdots & \vdots \\ f_1(x_n) & f_2(x_n) & \cdots & f_m(x_n) \end{pmatrix}, \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{pmatrix}$$

Die Gleichung $E(f) = 0$ hat nur im Spezialfall eine Lösung, wenn $m = n$ und wenn die Funktion f durch alle Punkte geht.

Normalgleichungen

Um $E(f)$ zu minimieren muss die erste Ableitung von $E'(f) = 0$ sein. Daher muss $E(f)$ nach jedem λ abgeleitet werden:

$$\frac{\partial E(f)(\lambda_1, \ldots, \lambda_m)}{\partial \lambda_j} = 0, j = 0, \ldots, m$$

Dies nennt sich eine Normalgleichung und lässt sich als $A^T A\lambda = A^T y$.

A ist oft schlecht konditioniert und die Lösung sollte daher mit dem QR-Verfahren gelöst werden.

Linearisieren

Falls eine f Funktion auf den ersten Blick nicht linear erscheint, kann sie eventuell linearisiert werden.

Z.B. die Funktion ae^{bx} kann mit $\log_e$ linearisiert werden.

$$\ln f(x) = \ln\left(ae^{bx}\right) = \ln(a) + b \cdot \ln\left(e^x\right) = \ln(a) \cdot \underbrace{1}_{f_1(x)} + b \cdot \underbrace{x}_{f_2(x)}$$

$$
\begin{aligned}
\boldsymbol{A}^T\boldsymbol{A} &= \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix} = \begin{pmatrix} 5 & 10 \\ 10 & 30 \end{pmatrix} \\
\boldsymbol{A}^T \cdot \widetilde{\boldsymbol{y}} &= \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \end{pmatrix} \begin{pmatrix} \ln 3 \\ \ln 1 \\ \ln 0.5 \\ \ln 0.2 \\ \ln 0.05 \end{pmatrix} = \begin{pmatrix} -4.1997 \\ -18.1975 \end{pmatrix} \\
\boldsymbol{A}^T\boldsymbol{A}\lambda &= \boldsymbol{A}^T\widetilde{\boldsymbol{y}} \Rightarrow \lambda = \begin{pmatrix} \ln a \\ b \end{pmatrix} = \begin{pmatrix} 1.11968... \\ -0.97981... \end{pmatrix} \\
\Rightarrow f(x) &= \mathrm{e}^{\ln a} \cdot \mathrm{e}^{bx} = 3.06388... \cdot \mathrm{e}^{-0.97981...x}
\end{aligned}
$$

Nicht-Lineare Ausgleichsrechnung

$$f(\lambda_1, \lambda_2, \ldots, \lambda_m, x) = \ldots$$

Das allgemeine Ausgleichsproblem besteht darin folgendes E zu minimieren:

$$
\begin{aligned}
E(f) &= \sum_{i=1}^{n} \left(y_i - f_p(\lambda_1, \lambda_2, ..., \lambda_m, x_i)\right)^2 = \| \begin{pmatrix} y_1 - f_p(\lambda_1, \lambda_2, ..., \lambda_m, x_1) \\ y_2 - f_p(\lambda_1, \lambda_2, ..., \lambda_m, x_2) \\ \vdots \\ y_n - f_p(\lambda_1, \lambda_2, ..., \lambda_m, x_n) \end{pmatrix} \|_2^2 \\
&= \| \boldsymbol{y} - \boldsymbol{f}(\lambda) \|_2^2
\end{aligned}
$$

Die Ableitung von E wird auf $E'(f) = 0$ gesetzt. Dafür kann das Gauss-Newton-Verfahren.

Gauss-Newton-Verfahren

Das Quadratmittelproblem ist es einen Vektor $x \in \mathbb{R}^m$ zu finden, welcher die Fehlerfunktional $E : \mathbb{R}^m \to \mathbb{R} := ||g(x)||_2^2$ minimiert. E gehört zur Funktion $g : \mathbb{R}^m \to \mathbb{R}^n$

g wird nun definiert als $g(\lambda) := y - f(\lambda)$.

Um nun für eine nicht lineare Funktionen f eine Lösung zu finden, muss f linearisiert werden:

$$g(\lambda) \approx g(\lambda_0) + Dg(\lambda_0) \cdot (\lambda - \lambda_0)$$

$$Dg(x) = \begin{pmatrix} \frac{\partial g_1}{\partial x_1}(\vec{x}) & \frac{\partial g_1}{\partial x_2}(\vec{x}) & \dots & \frac{\partial g_1}{\partial x_n}(\vec{x}) \\ \frac{\partial g_2}{\partial x_1}(\vec{x}) & \frac{\partial g_2}{\partial x_2}(\vec{x}) & \dots & \frac{\partial g_2}{\partial x_n}(\vec{x}) \\ \dots & \dots & \dots & \dots \\ \frac{\partial g_m}{\partial x_1}(\vec{x}) & \frac{\partial g_m}{\partial x_2}(\vec{x}) & \dots & \frac{\partial g_m}{\partial x_n}(\vec{x}) \end{pmatrix}$$

E kann nun folgendermassen definiert werden:

$$\tilde{E}(\lambda) = ||\underbrace{g(\lambda_k)}_{\tilde{y}} + \underbrace{Dg(\lambda_k)}_{-\tilde{A}} \cdot \underbrace{(\lambda - \lambda_k)}_{\delta}||_2^2$$

Wobei $k = 0, 1, \dots$ ist.

Dies kann nun wie eine lineare Gleichung gelöst werden:

$$Dg(\lambda_k)^T Dg(\lambda_k)\delta_k = -Dg(\lambda_k)^T \cdot g(\lambda_k)$$

Oder mit dem QR-Verfahren:

$$Dg(\lambda_k) = Q_k R_k$$
$$R_k \lambda_k = -Q_k^T g(\lambda_k)$$

Für jedes k wird nun $\tilde{E}$ minimiert, bzw. die obere Gleichung aufgelöst.

Das nächste λ kann wie folgt ausgerechnet wird:

$$\lambda_{k+1} = \lambda_k + \delta_k$$

Gedämpftes Gauss-Newton-Verfahren

Das gedämpte Gauss-Netwon-Verfahren funktioniert gleich, wie das "normale" Verfahren, nur das δ_k verkleinert wird.

Um das δ_k für die nächste Iteration zu finden, soll folgende für folgende Formel das minimale $p \in 0, 1, \dots, p_{max}$ gefunden werden

$$||g\left(\lambda_k + \frac{\delta_k}{2^p}\right)||_2^2 < ||g(\lambda_k)||_2^2$$

λ_{k+1} wird nun folgendermassen berechnet:

$$\lambda_{k+1} = \lambda_k + \frac{\delta_k}{2^p}$$

Falls kein minimales p gefunden werden kann, wird mit $p = 0$ gerechnet.

Nichtlineare Gleichungssysteme

Multivariate Funktionen

Skalarwertige Funktion

Eine Funktion, welche mehrere x-Werte nimmt und ein y-Wert zurück gibt.

$$f : \mathbb{R}^n \to \mathbb{R}$$
$$y = f(x_1, x_2, \ldots, x_n)$$

Vektorwertige Funktion

Eine Funktion, welche mehrere x-Werte nimmt und mehrere y-Werte zurück gegeben

$$f : \mathbb{R}^n \to \mathbb{R}^m$$
$$(y_1, y_2, \ldots, y_m) = f(x_1, x_2, \ldots, x_n)$$

Explizite und implizite Funktionen

Explizite Funktionen haben die folgende Form: $y = f(x_1, x_2, \ldots, x_n)$

Implizite Funktionen haben die folgende Form: $F(x_1, x_2, \ldots, x_n, y) = 0$

Partielle Ableitung

Um die Funktion $z = f(x, y) = 2x^2 + 5y$ abzuleiten, kann nach x und y separat abgeleitet werden:

$$\text{nach } x: \frac{\partial f}{\partial x} = 4x + 0$$
$$\text{nach } y: \frac{\partial f}{\partial y} = 0 + 5$$

Diese Ableitung kann folgendermassen visualisiert werden:

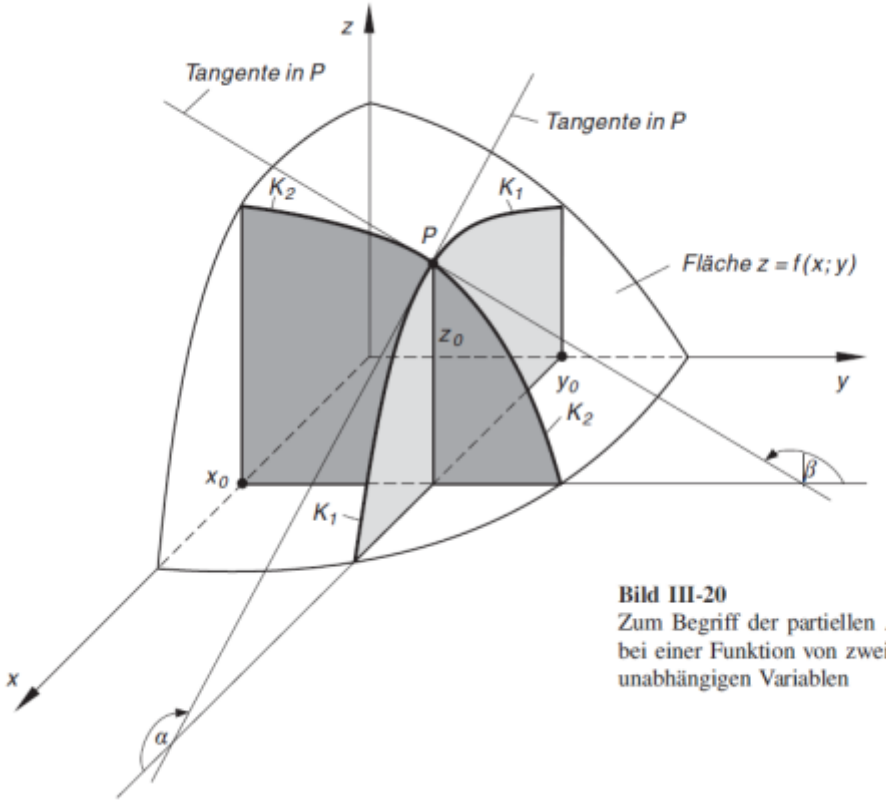


Bild III-20
Zum Begriff der partiellen Ableitung bei einer Funktion von zwei unabhängigen Variablen

Jacobi-Matrix

Für die Funktion $f: \mathbb{R}^n \to \mathbb{R}^m$ mit $\vec{y} = f(\vec{x}) = \begin{pmatrix} y_1 = f_1(\vec{x}) \\ y_2 = f_2(\vec{x}) \\ \dots \\ y_m = f_m(\vec{x}) \end{pmatrix}$ und

$\vec{x} = (x_1, x_2, \dots, x_n)^T$ ist die Jacobi-Matrix das folgende:

$$Df(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\vec{x}) & \frac{\partial f_1}{\partial x_2}(\vec{x}) & \dots & \frac{\partial f_1}{\partial x_n}(\vec{x}) \\ \frac{\partial f_2}{\partial x_1}(\vec{x}) & \frac{\partial f_2}{\partial x_2}(\vec{x}) & \dots & \frac{\partial f_2}{\partial x_n}(\vec{x}) \\ \dots & \dots & \dots & \dots \\ \frac{\partial f_m}{\partial x_1}(\vec{x}) & \frac{\partial f_m}{\partial x_2}(\vec{x}) & \dots & \frac{\partial f_m}{\partial x_n}(\vec{x}) \end{pmatrix}$$

In dieser Matrix ist in einer Reihe alle möglichen partiellen Ableitungen für $f_1(\vec{x})$

Linearisierung

Eine Approximation für $y = f(x)$ kann mit $f(x) \approx f(x_0) + f'(x - x_0)$.

Dasselbe kann auch für eine multivariante Funktion mithilfe der Jacobi-Matrix getan werden: $g(\vec{x}) = f(\vec{x_0}) + Df(\vec{x_0}) \cdot (\vec{x} - \vec{x_0})$

($Df(\vec{x})$ ist die Jacobi-Matrix)

Nach dem Linearisieren wird ein nichtlineare Funktion lineare und kann mit bekannten Verfahren gelöst werde

Newton-Verfahre

> *Das Newton-Verfahren erwartet, dass $f(\vec{x}_n) = \vec{0}$ gilt.*

$$\vec{x}_{n+1} = \vec{x}_n - (Df(\vec{x}_n))^{-1} \cdot f(\vec{x}_n)$$

Um nicht die Jacobi-Matrix invertieren zu müssen, kann folgender Trick angewendet werden:

$$\vec{\delta}_n := -(Df(\vec{x}_n))^{-1} \cdot f(\vec{x}_n)$$

Dies kann in folgendes Umgewandlet werden:

$$Df(\vec{x}_n) \cdot \vec{\delta} x_n = -f(\vec{x}_n)$$

Die Gleichung $Df(\vec{x}_n) \cdot \vec{\delta} x_n = -f(\vec{x}_n)$ ist ein lineares Gleichungssystem, welches relativ einfach gelöst werden kann. Danach kann $\vec{\delta}_n$ anstelle von $-(Df(\vec{x}_n))^{-1} \cdot f(\vec{x}_n)$ verwendet werden: $\vec{x}_{n+1} = \vec{x}_n + \vec{\delta}_n$

Das Newton-Verfahren konvergiert quadratisch für nah genug an einer Nullstelle $\overline{x}$ liegende Startwerte, wenn $Df(\overline{x})$ regulär und f dreimal stetig differenzierbar ist.

> *Für eine Nichtreguläre Matrix A gilt* $\det(A) \neq 0$

Mögliche Abbruchskriterien sind:

- $n > n_{max}$
- $||\vec{x}_{n+1} - \vec{x}_n|| \leq ||\vec{x}_{n+1}|| \cdot \varepsilon$
- $||\vec{x}_{n+1} - \vec{x}_n|| \leq \varepsilon$
- $||f(\vec{x}_{n+1})|| \leq \varepsilon$$

Vereinfachtes Newton-Verfahren

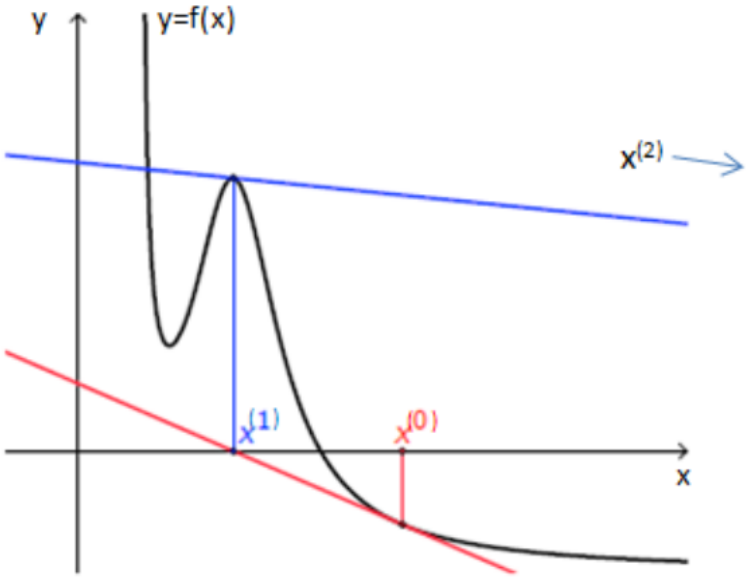
Beim regulären Newton-Verfahren muss bei jeden Iterationsschritt die Jacobi-Matrix $Df(\vec{x})$ neuberechnen. Beim vereinfachten Newton-Verfahren wird $Df(\vec{x})$ nur für den Startvektor berechnet.

$$\vec{x}_{n+1} = \vec{x}_n - (Df(\vec{x}_0))^{-1} \cdot f(\vec{x}_n)$$
$$Df(\vec{x}_0) \cdot \vec{\delta x}_n = -f(\vec{x}_n)$$

Wegen dieser Vereinfachung kovergiert das Verfahren nur noch linear gegen die Nullstelle, wenn $Df(\overline{x})$ nicht regulär ist.

Gedämptes Newton-Verfahren

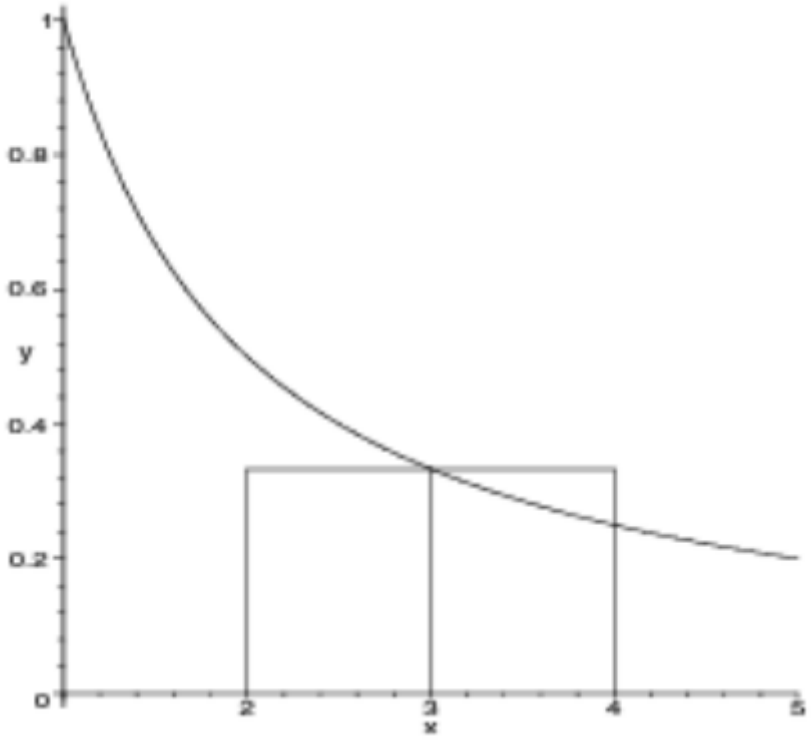
Wenn $Df(\vec{x}_n)$ schlecht konvergiert, dann kann nicht generell erwartet werden, dass $\vec{x}_{n+1} = \vec{x}_n + \vec{\delta}_n$ nicht gilt.

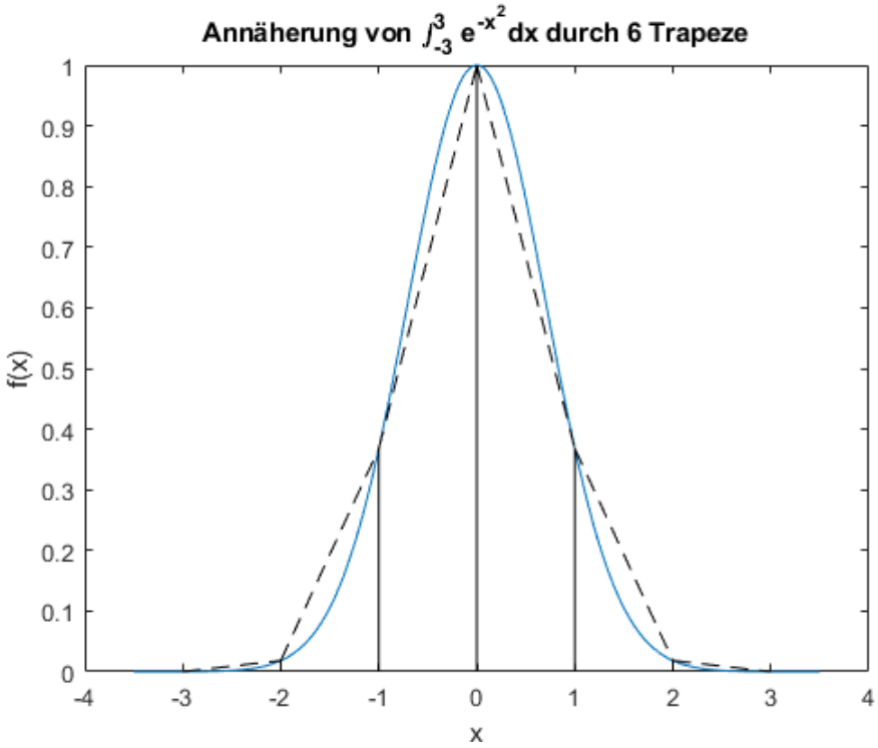


Für das gedämpfte Newton-Verfahren werden folgende Schritte angewendet:

1. Berechne δ_n mit $Df(\vec{x}_n)\delta_n = -f(\vec{x}_n)$ ausgerechnet
2. Finde das minimale $k \in \{0, 1, \dots\}$ für das gilt: $||f(\vec{x}_n + \frac{\vec{\delta}_n}{2^k})||_2 < ||f(\vec{x}_n)||_2$
3. Wenn kein k gefunden wird, soll mit $k = 0$ weiter gerechnet werden
4. Nun soll die Iterationsgleichung $\vec{x}_{n+1} = \vec{x}_n + \frac{\vec{\delta}_n}{2^k}$ verwendet werden# Numerische Itegration

Rechteck- & Trapezregel





Die folgenden formel ziehen ein Rechteck, bzw. Trapez über das ganze Integral.

Das Integral

$$\int_a^b f(x)\mathrm{d}x$$

kann folgendermassen approximiert werden

$$Rf = f\left(\frac{a+b}{2}\right) \cdot (b-a)$$

$$Tf = \frac{f(a)+f(b)}{2} \cdot (b-a)$$

(Rf = Rechtecksregel, Tf = Trapezregel)

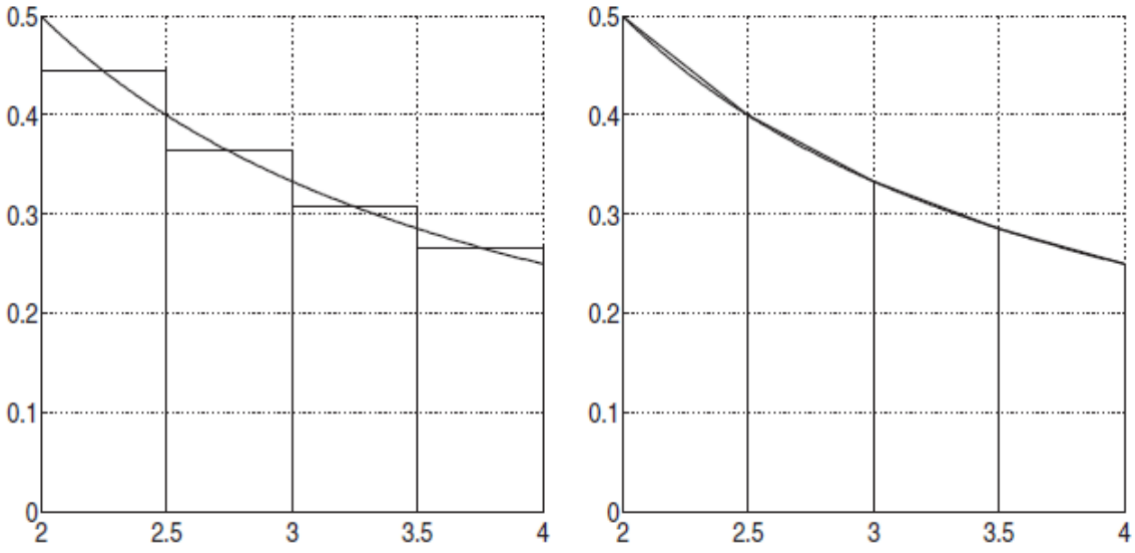
Für die summierte Rechteck- & Trapezregel wird das Integral in kleinere Schritte mit der breite h unterteilt.

$$Rf(h) = h \cdot \sum_{i=0}^{n-1} f(x_i + \frac{h}{2})$$

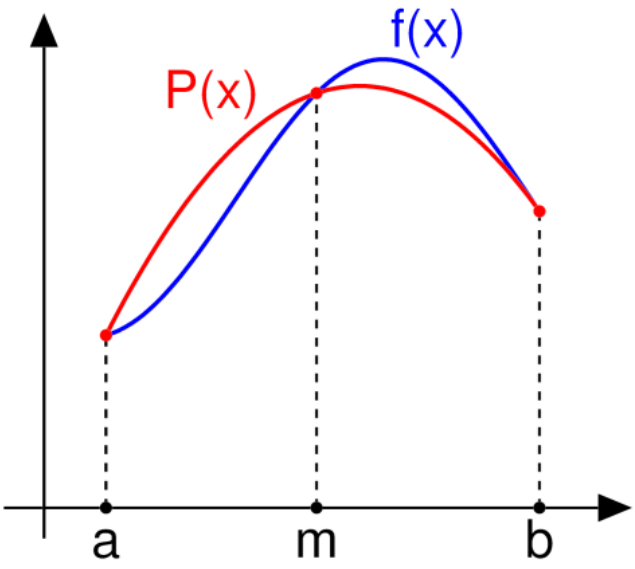
$$Tf(h) = h \cdot \left(\frac{f(a)+f(b)}{2} + \sum_{i=1}^{n-1} f(x_i)\right)$$

$$\text{wobei gilt} x_i = a + i \cdot h$$

$$h = \frac{b-a}{n}$$



Simpsonregel



Für das lösen eines Segments müssen folgende Formel ausgerechnet werden. Dabei wird das Polynom $p(x) = \alpha + \beta(x-a) + \gamma(x-a)(x-b)$ verwendet.

$$\begin{aligned}
\alpha &= f(a) \\
\beta &= \frac{f(b)-f(a)}{b-a} \\
\gamma &= \frac{f(\frac{b+a}{2}) - f(a) - \frac{f(b)-f(a)}{b-a} \cdot (\frac{b-a}{2})}{-(\frac{b-a}{2})^2} = \frac{f(a) - 2f(\frac{b+a}{2}) + f(b)}{2(\frac{b-a}{2})^2}.
\end{aligned}$$

Da $f(x) \approx p(x)$ gilt, kann das Polynom integriert werden:

$$\int_a^b f(x)dx \approx \int_a^b p(x)dx = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)$$

Die Regel oben haben nur ein Segment benutzt. Wie aber auch bei der Rechtecks- und Trapezregel, kann auch hier die summierte Simpsonregel verwendet werden.

$$Sf(h) = \frac{h}{3}\left(\frac{1}{2}f(a) + \sum_{i=1}^{n-1} f(x_i) + 2\sum_{i=1}^{n} f\left(\frac{x_{i-1}+x_i}{2}\right) + \frac{1}{2}f(b)\right)$$

Die Simpsonsregel kann auch mit dem Rechtecks- und Trapezregel berechnet werden:

$$Sf(h) = \frac{1}{3}(Tf(h) + 2Rf(h))$$

Fehlerabschätzung

$$\left| \int_a^b f(x)\,dx - Rf(h) \right| \leq \frac{h^2}{24}(b-a) \cdot \max_{x\in[a,b]} |f''(x)|$$

$$\left| \int_a^b f(x)\,dx - Tf(h) \right| \leq \frac{h^2}{12}(b-a) \cdot \max_{x\in[a,b]} |f''(x)|$$

$$\left| \int_a^b f(x)\,dx - Sf(h) \right| \leq \frac{h^4}{2880}(b-a) \cdot \max_{x\in[a,b]} |f^{(4)}(x)|$$

Gaussformel

Die folgenden Formel bestimmen das Integral zwischen a und b, wenn es n Stützpunkte gibt. Dabei müssen die Stützpunkte nicht äquidistant sein.

Satz 7.2 [1]: Gauss Formeln für n=1, 2, 3:

- Die Gauss Formeln für $n = 1, 2, 3$ für $\int_a^b f(x)dx \approx \frac{b-a}{2} \sum_{i=1}^{n} a_i f(x_i)$ lauten:
  - n = 1: $G_1 f = (b-a) \cdot f(\frac{b+a}{2})$
  - n = 2: $G_2 f = \frac{b-a}{2}\left[f\left(-\frac{1}{\sqrt{3}} \cdot \frac{b-a}{2} + \frac{b+a}{2}\right) + f\left(\frac{1}{\sqrt{3}} \cdot \frac{b-a}{2} + \frac{b+a}{2}\right)\right]$
  - n = 3: $G_3 f = \frac{b-a}{2}\left[\frac{5}{9} \cdot f\left(-\sqrt{0.6} \cdot \frac{b-a}{2} + \frac{b+a}{2}\right) + \frac{8}{9} \cdot f\left(\frac{b+a}{2}\right)\right]$

$$+ \frac{b-a}{2}\left[\frac{5}{9} \cdot f\left(\sqrt{0.6} \cdot \frac{b-a}{2} + \frac{b+a}{2}\right)\right]$$

Romberg Extrapolation

Satz 7.3 [1]: Romberg-Extrapolation

- Für die summierte Trapezregel $Tf(h)$ zur näherungsweisen Berechnung von $I(f) = \int_a^b f(x)\,dx$ gilt:
  Sei $T_{j0} = Tf\left(\frac{b-a}{2^j}\right)$ für $j = 0,1,\ldots,m$. Dann sind durch die Rekursion

$$T_{jk} = \frac{4^k \cdot T_{j+1,k-1} - T_{j,k-1}}{4^k - 1}$$

  für $k = 1,2,\ldots,m$ und $j = 0,1,\ldots,m-k$ Näherungen der Fehlerordnung $2k+2$ gegeben. Diese Methode heisst **Romberg-Extrapolation**. Die verwendete Schrittweitenfolge $h_j = \frac{b-a}{2^j}$ heisst auch Romberg-Folge.

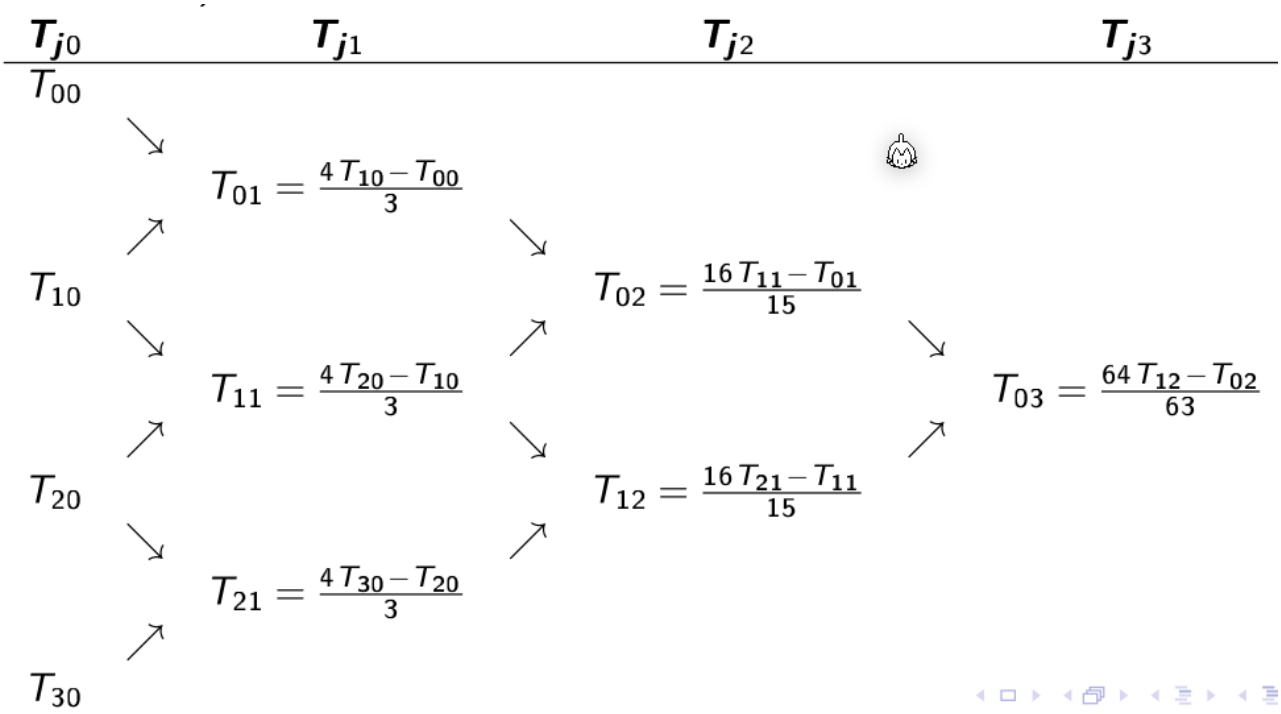
Die Rekursion wird ausgerechnet bis $k = 0$ wird, da dann die Formel $T_{j0} = Tf\left(\frac{b-a}{2^j}\right)$

Da die Werte von $f(\dots)$ immer in T_{j0} wiederverwendet werden, kann dies mit der folgenden Formel vereinfacht werden:

$$T_{j0} = \frac{1}{2}T_{j-1,0} + h_j \sum_{i=1}^{n_{j-1}} f(a + (2i-1)h_j)$$

Die zweite Spalte T_{j1} kann mit der Simpson-Regel berechnet werden:

Die folgende Graphik zeigt die oben abgebildete Rekursion:



Die folgenden zwei Graphen zeigen T_{00} und T_{10}. Wenn j um `1` höher wird, wird die X-Achse halbiert. Dasselbe gilt für T_{30} und T_{40}