

Numerik 2 - HS19
Pascal Brunner - brunnpa7

Inhaltsverzeichnis

1 numerische Differenzen	3
1.1 Lernziele	3
1.2 Problemstellung	3
1.3 Repetition Taylor-Reihe	3
1.4 Vorwärtsdifferenz	3
1.5 Diskretisierungsfehler	4
1.6 zentrale Differenz	4
1.7 Rückwärtsdifferenz	5
1.8 Vorwärtsdifferenz / zentrale Differenz / Rückwärtsdifferenz	5
1.9 Differenzformeln für höhere Ableitungen	5
1.10 Differenzformeln für partielle Ableitungen	6
1.11 Extrapolation von Differenzformeln	6
2 Numerische Integration	7
2.1 Problemstellung	7
2.2 Rechteck- und Trapezregel	7
2.3 Die Simpson-Regel	8
2.4 Der Fehler der summierten Quadraturformeln	9
2.5 Gauss-Formel	9
2.6 Romberg-Extrapolation	10
3 Einführung in gewöhnliche Differentialgleichungen	11
3.1 Lernziele	11
3.2 Problemstellung	11
3.3 Richtungsfelder für Differentialgleichungen 1. Ordnung	12
3.4 Das Euler-Verfahren	13
3.4.1 Das klassische Euler-Verfahren	13
3.4.2 Das Mittelpunkt-Verfahren	14
3.4.3 Das modifizierte Euler-Verfahren (auch Heun-Verfahren genannt)	15
3.5 Die Fehlerordnung eines Verfahrens	16
3.6 Runge-Kutta Verfahren	18
3.6.1 Das klassische vierstufige Runge-Kutta Verfahren	18
3.6.2 Das allgemeine s-stufige Runge-Kutta Verfahren	19
3.7 Mehrschrittverfahren	20
3.7.1 Adams-Basforth Methode 2. und 3. Ordnung	20
3.7.2 Adams Basforth Methoden höherer Ordnung	21
3.8 Erweiterung auf Systeme von Differentialgleichungen	21
3.9 Zurückföhre einer DGL k -ter Ordnung auf k DGL 1. Ordnung	21
3.10 Lösen eines Systems von k DGL 1. Ordnung	21
3.11 Stabilität	22
3.12 Weitere Punkte	24
3.12.1 Implizite vs. explizite Verfahren	24
3.12.2 Steife DGL	25
3.12.3 Schrittweitensteuerung	25
3.12.4 MATLAB Funktionen zur Lösungen von Anfangswertproblemen	25
4 Interpolation	25
4.1 Lernziele	25
4.2 Problemstellung	25
4.3 Polynominterpolation	26
4.4 Splineinterpolation	27

5 Ausgleichsrechnung	29
5.1 Lernziele	29
5.2 Problemstellung	29
5.3 Lineare Ausgleichsprobleme	30
5.4 Nichtlineare Ausgleichsprobleme	32
5.5 Das Gauss-Newton-Verfahren	33
6 Fourier-Reihen und Fourier-Transformation	35
6.1 Lernziele	35
6.2 Anwendungen	35
6.3 Fourier-Reihen	35
6.3.1 Allgemeine Fourier-Reihen	36
6.4 diskrete Fourier-Transformation	36

1 numerische Differenzen

Einleitender Text

1.1 Lernziele

Sie können mittels Differenzenformeln die Ableitung einer Funktion numerisch annähern und die dabei gemachten Fehler qualifizieren und mit dem Satz von Taylor auch quantifizieren

Sie können diese Differenzenformeln extrapoliere

Sie kennen die wichtigsten Verfahren der numerischen Integration und können damit bestimmte Integrale berechnen.
Sie können die dabei auftretenden Fehler bestimmen

Sie können die Romberg-Extrapolation anwenden

Sie können die hier vorgestellten Verfahren in MATLAB implementieren

1.2 Problemstellung

Liegt die Funktion in Form einer differenzierbaren Funktionsgleichung $y = f(x)$ vor, kann die Ableitung $y' = f'(x)$ analytisch mit den Verfahren der Analysis berechnet werden. Ist dies mit einem gewissen Aufwand verbunden, kann es sinnvoller sein, eine numerische Näherung für die Ableitung zu verwenden.

Liegt die Funktion als eine Tabelle diskreter Funktionswerte $(x_i, f(x_i))$ mit $i = 1, \dots, m$ vor, kann die Ableitung nicht analytisch berechnet werden. Dies ist beispielsweise bei realen Daten der Fall.

Dabei gibt es dann zwei Optionen:

1. Zuerst werden die Wertpaare durch einen sogenannten Fit angenähert. Dadurch erhält man näherungsweise die Funktionsgleichung $y = \tilde{f}(x)$. Der Vorteil dabei ist, dass der Einfluss von Fehlern und Ausreissern reduziert werden kann. Danach wird die Funktionsgleichung $y = \tilde{f}(x)$ entweder analytisch oder numerisch abgeleitet.
2. Man verzichtet auf einen Fit und wendet direkt das numerische Verfahren an. Je nach Datenqualität und dem verwendeten Verfahren kann dies zu erheblichen Fehlern führen

1.3 Repetition Taylor-Reihe

Mit den Taylor-Reihen erhält man eine Näherung einer Funktion. Ziel dabei ist es das Taylor-Polynom n-ten Grades zu bestimmen, welches mit der Ableitung der Funktion n-ten Grades übereinstimmt.

Definition 1 (Allgemeine Formel zum Taylorpolynom).

$$T_n f(x) = \sum_{k=0}^n \frac{f^{(k)}(x)}{k!} x^k \quad (1)$$

1.4 Vorwärtsdifferenz

Wir gehen davon aus, dass wir es mit einer stetig differenzierbaren Funktion $f[a,b] \rightarrow \mathbb{R}$ zu tun haben mit einem beliebigen $x_0 \in [a,b]$. Gesucht sind Näherungswerte für die erste Ableitung $f'(x_0)$ oder auch für die k-te Ableitung $f^{(k)}(x_0)$.

Dabei ist die (erste) Ableitung definiert als

$$\begin{aligned} f'(x_0) &= \lim_{h \rightarrow 0} \frac{f(x_0 + h) - f(x_0)}{h} \\ &\approx \frac{f(x_0 + h) - f(x_0)}{h} =: D_1 f(x_0, h) \end{aligned} \quad (2)$$

Man nennt $D_1 f(x_i, h)$ eine *Vorwärtsdifferenz* (oder auch Differenzformel oder finite Differenz erster Ordnung), da sie einen Näherungswert für die erste Ableitung $f'(x_0)$ liefert, sofern $h \in \mathbb{R}$ "ausreichend" klein ist.

Für eine stetige Funktion $f(x)$ kann h beliebig kleine Werte annehmen und $f(x_0 + h)$ ist berechenbar. Der Einfachheitshalber machen wir die Einschränkung, dass h konstant ist. Es gilt dabei $x_{i+1} = x_i + h$:

$$D_1 f(x_i, h) = \frac{f(x_i+h) - f(x_i)}{h} = \frac{f(x_i+h) - f(x_i)}{x_{i+1} - x_i}$$

Satz (Taylor-Entwicklung). *Die Funktion $f : [a, b] \rightarrow \mathbb{R}$ sei $(n+1)$ -mal stetig differenzierbar, $x_0 \in [a, b]$. Dann gibt es für alle $x \in [a, b]$ ein z zwischen x_0 und x , so dass*

$$f(x) = \sum_{k=0}^n \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k + R_{n+1}(x) \quad (3)$$

mit dem Taylorschen Restglied

$$R_{n+1}(x) = \frac{f^{(n+1)}(z)}{(n+1)!} (x - x_0)^{n+1} \quad (4)$$

Wir können $f(x)$ mittels $x = x_0 + h$ bzw. $h = x - x_0$ wie folgt schreiben:

$$f(x) = f(x_0 + h) = f(x_0) + f'(x_0)h + \frac{f''(x_0)}{2!}h^2 + \frac{f'''(x_0)}{3!}h^3 + \frac{f^{(4)}(x_0)}{4!}h^4 + \dots + \frac{f^{(n+1)}(z)}{(n+1)!}h^{n+1} \quad (5)$$

1.5 Diskretisierungsfehler

Für die erste Ableitung ($n=1$) folgt durch umformen für den Fehler die Differenz erster Ordnung.

$$D_1 f(x_0, h) - f'(x_0) = \frac{f''(z)}{2}h \quad (6)$$

Den Fehler einer Differenzformel lässt sich allgemein beschreiben:

Definition 2 (Diskretisierungsfehler / Fehlerordnung).

Sei $Df(x_0, h)$ eine Differenzformel für die Näherung von $f'(x_0)$. Dann bezeichnet man den absoluten Fehler

$$|Df(x_0, h) - f'(x_0)| \quad (7)$$

als **Diskretisierungsfehler**

Die Differenzformel $Df(x_0, h)$ hat die **Fehlerordnung** (oder kurz: Ordnung) k , falls es ein $C > 0$ gibt, so dass für genügend kleines h gilt:

$$|Df(x_0, h) - f'(x_0)| \leq C * h^k \quad (8)$$

Bemerkung

die obige Definition lässt sich auf Differenzformeln höhere Ableitungen anwenden

Man bezeichnet Ausdrücke, die für genügend kleines h durch Ch^k beschränkt werden, auch mit $O(h^k)$ und man schreibt

$$|Df(x_0, h) - f'(x_0)| = O(h^k) \quad (9)$$

1.6 zentrale Differenz

Die zentrale Differenz hat zum Ziele eine genauere Differenzformel als $D_1 f$ für die erste Ableitung zu berechnen. → wir möchten eine höhere Fehlerordnung erzielen.

Dabei verwenden wir $x = x_0 \pm h$ und $n = 3$ und setzen diese in die Taylorsche Formel ein:

$$\begin{aligned} f(x_0 + h) &= f(x_0) + f'(x_0)h + \frac{f''(x_0)}{2}h^2 + \frac{f'''(x_0)}{6}h^3 + \frac{f^{(4)}(z_1)}{24}h^4 \\ f(x_0 - h) &= f(x_0) - f'(x_0)h + \frac{f''(x_0)}{2}h^2 - \frac{f'''(x_0)}{6}h^3 + \frac{f^{(4)}(z_1)}{24}h^4 \end{aligned} \quad (10)$$

Durch subtrahieren der beiden Ausdrücke erhält man:

$$\begin{aligned} f(x_0 + h) - f(x_0 - h) &= 2f'(x_0)h + \frac{f'''(x_0)}{3}h^3 + \dots \\ \Rightarrow f'(x_0) &= \frac{f(x_0 + h) - f(x_0 - h)}{2h} - \frac{f'''(x_0)}{6}h^2 + \dots \end{aligned} \quad (11)$$

Also erhalten wir die zentrale Differenz $D_2 f(x_0, h)$ als Näherung für $f'(x_0)$

$$D_2 f(x_0, h) = \frac{f(x_0 + h) - f(x_0 - h)}{2h} \quad (12)$$

mit dem Diskretisierungsfehler der Fehlerordnung $O(h^2)$:

$$|D_2 f(x_0, h) - f'(x_0)| = \frac{|f'''(x_0)|}{6}h^2 + \dots \quad (13)$$

1.7 Rückwärtsdifferenz

Analog zur Vorwärtsdifferenz bestimmen wir die Rückwärtsdifferenz mit der Fehlerordnung $O(h)$

$$D_3 f(x_0, h) = \frac{f(x_0) - f(x_0 - h)}{h} \quad (14)$$

1.8 Vorwärtsdifferenz / zentrale Differenz / Rückwärtsdifferenz

$$\begin{aligned} D_1 f(x_0, h) &= \frac{f(x_0 + h) - f(x_0)}{h} \\ D_2 f(x_0, h) &= \frac{f(x_0 + h) - f(x_0 - h)}{2h} \\ D_3 f(x_0, h) &= \frac{f(x_0) - f(x_0 - h)}{h} \end{aligned} \quad (15)$$

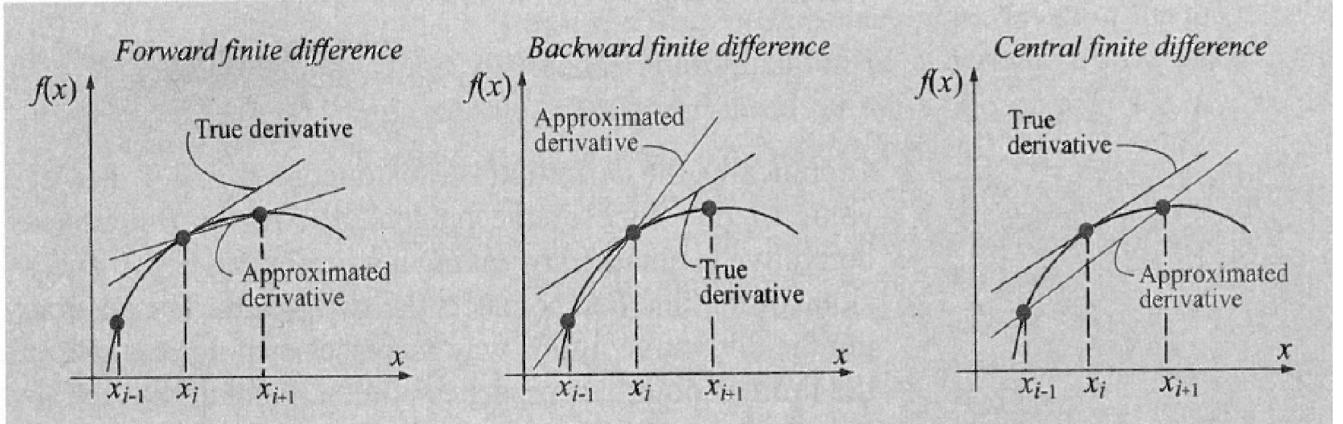


Abbildung 1: Vorwärtsdifferenz (links), Rückwärtsdifferenz (Mitte) und zentrale Differenz (rechts) als Näherung für die Ableitung $f'(x_i)$

1.9 Differenzformeln für höhere Ableitungen

Analog zur ersten Ableitung kann die Differenzform als Näherung für die zweite Ableitung konstruiert werden (Vorwärtsdifferenz, zentrale Differenz, Rückwärtsdifferenz)

$$\begin{aligned} D_4 f(x_0, h) &= \frac{f(x_0 + 2h) - 2f(x_0 + h) + f(x_0)}{h^2} \\ D_5 f(x_0, h) &= \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} \\ D_6 f(x_0, h) &= \frac{f(x_0) - 2f(x_0 - h) + f(x_0 - 2h)}{h^2} \end{aligned} \quad (16)$$

1.10 Differenzformeln für partielle Ableitungen

Wenn es eine Funktion mit mehreren Variablen (bspw. $u = u(x,y)$) ist, so verwendet man die partielle Ableitung. Dabei ergibt sich die erste Ableitung für x folgt:

$$\begin{aligned} D_1 : \frac{\partial u}{\partial x}(x_0, y_0) &\approx \frac{u(x_0 + h, y_0) - u(x_0, y_0)}{h} \\ D_2 : \frac{\partial u}{\partial x}(x_0, y_0) &\approx \frac{u(x_0 + h, y_0) - u(x_0 - h, y_0)}{2h} \\ D_3 : \frac{\partial u}{\partial x}(x_0, y_0) &\approx \frac{u(x_0, y_0) - u(x_0 - h, y_0)}{h} \end{aligned} \quad (17)$$

zweite partielle Ableitung nach x :

$$\begin{aligned} D_4 : \frac{\partial^2 u}{\partial x^2}(x_0, y_0) &\approx \frac{u(x_0 + 2h, y_0) - 2u(x_0 + h, y_0) + u(x_0, y_0)}{h^2} \\ D_5 : \frac{\partial^2 u}{\partial x^2}(x_0, y_0) &\approx \frac{u(x_0 + h, y_0) - 2u(x_0, y_0) + u(x_0 - h, y_0)}{h^2} \\ D_6 : \frac{\partial^2 u}{\partial x^2}(x_0, y_0) &\approx \frac{u(x_0, y_0) - 2u(x_0 - h, y_0) + u(x_0 - 2h, y_0)}{h^2} \end{aligned} \quad (18)$$

→ Analog geht man für die partielle Ableitung nach y vor.

1.11 Extrapolation von Differenzformeln

Erkenntnis von oben: höhere Ordnungen sind i.d.R. genauer als niedriger Ordnung. Unter dem Begriff **Extrapolation** versteht man eine rekursive Methode, wo man aus einer Formeln niedriger Ordnung Formeln höherer Ordnung gewinnen kann.

Algorithmus zur h-Extrapolation Sei $D(h)$ eine Formel zur Näherung von \bar{D} mit der Fehlerentwicklung

$$D(h) - \bar{D} = c_1 h^1 + c_2 h^2 + c_3 h^3 + \dots \quad (19)$$

sei h grösser 0 eine Ausgangsschrittweite und $D_{i0} = D(\frac{h}{2^i})$ für $i = 0, 1, \dots, m$

Dann sind durch die Rekursion

$$D_{ik} = \frac{2^k * D_{i+1,k-1} - D_{i,k-1}}{2^k - 1} \quad (20)$$

(Wobei $k=1,2,\dots,m$ und $i=0,1,\dots,m-k$) Näherungen für \bar{D} gegeben mit der Fehlerordnung $k+1$

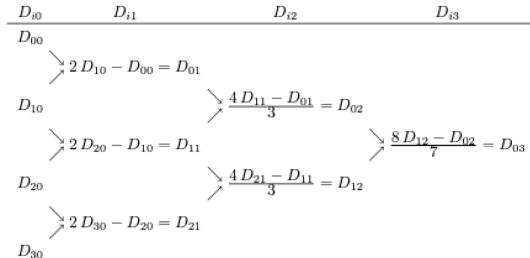


Abbildung 2: Schema h-Extrapolation

Wenn eine Differenzenformel eine Fehlerentwicklung mit nur geraden Potenzen hat, lässt sich die Extrapolation noch effizienter berechnen:

Algorithmus zur h^2 -Extrapolation Sei $D(h)$ eine Formel zur Näherung von D mit der Fehlerentwicklung

$$D(h) - D = c_1 h^2 + c_2 h^4 + c_3 h^6 + \dots \quad (21)$$

sei h grösser 0 eine Ausgangsschrittweite und $D_{i0} = D(\frac{h}{2^i})$ für $i = 0, 1, \dots, m$
Dann sind durch die Rekursion

$$D_{ik} = \frac{4^k * D_{i+1,k-1} - D_{i,k-1}}{4^k - 1} \quad (22)$$

(Wobei $k=1,2,\dots,m$ und $i=0,1,\dots,m-k$) Näherungen für \bar{D} gegeben mit der Fehlerordnung $2(k+1)$

D_{i0}	D_{i1}	D_{i2}	D_{i3}
D_{00}			
	$\nearrow \frac{4D_{10} - D_{00}}{3} = D_{01}$		
D_{10}		$\nearrow \frac{16D_{11} - D_{01}}{15} = D_{02}$	
	$\nearrow \frac{4D_{20} - D_{10}}{3} = D_{11}$		$\nearrow \frac{64D_{12} - D_{02}}{63} = D_{03}$
D_{20}		$\nearrow \frac{16D_{21} - D_{11}}{15} = D_{12}$	
	$\nearrow \frac{4D_{30} - D_{20}}{3} = D_{21}$		
D_{30}			

Abbildung 3: Schema h^2 -Extrapolation

2 Numerische Integration

Das Verfahren für die numerische Integration (auch Quadratur genannt) wird zum Lösen von nicht differenzierbaren Funktionen verwendet.

2.1 Problemstellung

Für eine Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$ soll das bestimmte Integral $I(f) = \int_b^a f(x)dx$ auf einem Intervall a, b numerisch berechnet werden.

Allgemeine Form der Quadraturverfahren

$$I(f) = \sum_{i=1}^n a_i f(x_i) \quad (23)$$

Dabei nennt man die x_i die *Stützstellen* oder *Knoten* der Quadraturformel und die a_i die *Gewicht*. Die Funktion selbst kann wieder als Funktionsgleichung $y=f(x)$ oder als Werttabelle $(x_i, f(x_i))$ mit $i = 1, \dots, m$ vorliegen

2.2 Rechteck- und Trapezregel

Definition 3 (Rechteckregel / Trapezregel). Die **Rechteckregel** (bzw. Mittelpunktregel Rf und die **Trapezregel** Tf zur Approximation von $\int_a^b f(x)dx$) sind definiert als

$$\begin{aligned} Rf &= f\left(\frac{a+b}{2}\right) * (b-a) \\ Tf &= \frac{f(a) + f(b)}{2} * (b-a) \end{aligned} \quad (24)$$

- Wir erhalten die **Rechtecksregel** wenn wir $f(x)$ in $\int_a^b dx$ durch eine Konstante (Polynom 0. Grades) ersetzen.
- Wenn wir $f(x)$ durch eine Gerade (Polynom 1. Grades) ersetzen, erhalten wir analog die **Trapezregel**

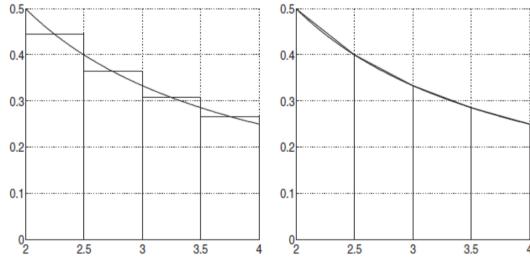


Abbildung 4: (links) summierte Rechtecksregel, (rechts) summierte Trapezregel

Um die Genauigkeit zu Steigern, lohnt sich das Intervall

$$a, b$$

in n Subintervalle mit der Schrittweite $h = \frac{b-a}{n}$ zu unterteilen und anschliessend aufzusummen. Als Resultat erhalten wir die summierten Regeln.

Definition 4 (summierte Rechteckregel / summierte Trapezregel). Sei $f : [a, b] \rightarrow \mathbb{R}$ stetig, $n \in \mathbb{N}$ die Anzahl Subintervalle

$$x_i, x_{i+1}$$

auf

$$a, b$$

mit der konstanten Breite $h = x_{i+1} - x_i = \frac{b-a}{n}$ und $x_i = a + i * h$ für $i = 0, \dots, n-1$.

Die **summierte Rechteckregel** (bzw. summierte Mittelpunktregel) $Rf(h)$ und die **summierte Trapezregel** $Tf(h)$ zur Approximation von $\int_a^b f(x)dx$ sind gegeben durch

$$\begin{aligned} Rf(h) &= h * \sum_{i=0}^{n-1} f(x_i) \\ Tf(h) &= h * \left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^{n-1} f(x_i) \right) \end{aligned} \tag{25}$$

2.3 Die Simpson-Regel

Wenn wir $f(x)$ durch ein Polynom $p(x)$ 2. Grades ersetzen erhalten wir einen neuen Ansatz:
Wir machen für $p(x)$ und $x \in [a, b]$ den Ansatz

$$p(x) = \alpha + \beta(x - a) + \gamma(x - a)(x - b) \tag{26}$$

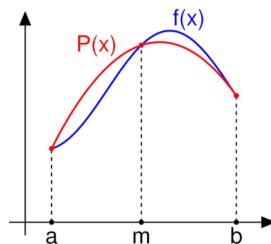


Abbildung 5: Funktion $f(x)$ wird auf dem Intervall durch ein Polynom $P(x)$ angenähert

und fordern, dass $p(x)$ an den Stellen $x = a$, $x = \frac{b+a}{2}$ und $x = b$ exakt mit $f(x)$ übereinstimmt, also:

$$\begin{aligned} p(a) &= \alpha! = f(a) \\ p\left(\frac{b+a}{2}\right) &= \alpha + \beta\left(\frac{b+a}{2} - a\right) + \gamma\left(\frac{b+a}{2} - a\right)\left(\frac{b+a}{2} - b\right) = \alpha + \beta\left(\frac{b-a}{2}\right) + \gamma\left(\frac{b-a}{2}\right)\left(\frac{a-b}{2}\right)! = f\left(\frac{b+a}{2}\right) \\ p(b) &= \alpha + \beta(b-a)! = f(b) \end{aligned} \quad (27)$$

Dabei ist die Schrittweite mittels $h = \frac{b-a}{2}$ gesetzt. Das Näherungspolynom $f(x)$ ist eindeutig bestimmt und es kann mittels der Potenzregel einfach integriert werden, da $f(x) \approx p(x)$ gilt:

Simpson-Regel

$$\int_a^b f(x)dx \approx \int_a^b p(x)dx = \frac{h}{3}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + \frac{1}{2}f(b)\right) \quad (28)$$

Analog zur Rechtecks- und Trapez-Regel kann die Genauigkeit erhöht werden. Dies geschieht in dem wir das Intervall $[a, b]$ in n Intervalle mit der Schrittweite $h = \frac{b-a}{n}$ unterteilt und aufsummiert wird.

summierte Simpson-Regel

$$Sf(h) = \frac{h}{3}\left(\frac{1}{2} + \sum_{k=1}^{n-1} f(x_k) + 2 \sum_{k=1}^n f\left(\frac{x_{k-1}+x_k}{2}\right) + \frac{1}{2}f(b)\right) \quad (29)$$

wobei wir $x_k = a + k * h$ gesetzt haben

2.4 Der Fehler der summierten Quadraturformeln

Der Fehler einer Näherung ist wie immer definiert als Betrag der Differenz zwischen dem exakten Wert und der Näherung.

Satz (Fehlerabschätzung für summierte Quadraturformeln). *Wobei $Rf(h)$ für die summierte Reckeckregel, $Tf(h)$ für die summierte Trapezregel und $Sf(h)$ für die summierte Simpsonregel gilt:*

$$\begin{aligned} \left| \int_a^b f(x)dx - Rf(h) \right| &\leq \frac{h^2}{24}(b-a) * \max_{x \in [a,b]} |f''(x)| \\ \left| \int_a^b f(x)dx - Tf(h) \right| &\leq \frac{h^2}{12}(b-a) * \max_{x \in [a,b]} |f''(x)| \\ \left| \int_a^b f(x)dx - Sf(h) \right| &\leq \frac{h^4}{2800}(b-a) * \max_{x \in [a,b]} |f^{(4)}(x)| \end{aligned} \quad (30)$$

2.5 Gauss-Formel

Bis zum aktuell Zeitpunkt haben wir die Stützstellen jeweils x_i äquidistant gewählt (\rightarrow die Schrittweite $h = \frac{b-a}{n}$ war konstant) und konnten somit die Rechtecks-, Trapez- und Simpson-Formel herleiten. Diese drei Formeln werden auch als Newton-Cotes Formeln der Ordnung $N = 0, 1$ und 2 bezeichnet. Wobei dies dem Grad des verwendeten Polynomons entspricht.

Die Stützstellen x_i können jedoch auch so gewählt werden, dass das Integral $\int_a^b f(x)dx$ 'optimal' approximieren.

Daraus erhält man die Gauss-Formel.

Dafür werden die Stützstellen x_i und die Gewichte a_i in der generellen Quadraturformel

$$I(f) = \sum_{i=1}^n a_i f(x_i) \quad (31)$$

so gewählt, dass die *Fehlerordnung* möglichst hoch bzw. der Fehler

$$\left| \int_a^b f(x)dx - I(f) \right| \quad (32)$$

möglichst klein wird

Satz (Gauss-Formel 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:

$$\begin{aligned} n = 1 : G_1 f &= (b-a) * f\left(\frac{b+a}{2}\right) \\ n = 2 : G_2 f &= \frac{b-a}{2} [f\left(-\frac{1}{\sqrt{3}} * \frac{b-a}{2} + \frac{b+a}{2}\right) + f\left(\frac{1}{\sqrt{3}} * \frac{b-a}{2} + \frac{b+a}{2}\right)] \\ n = 3 : G_3 f &= \frac{b-a}{2} [\frac{5}{9} * f\left(-\sqrt{0.6} * \frac{b-a}{2} + \frac{b+a}{2}\right) + \frac{8}{9} * f\left(\frac{b+a}{2}\right)] \\ &\quad + \frac{b-a}{2} [\frac{5}{9} * f\left(\sqrt{0.6} * \frac{b-a}{2} + \frac{b+a}{2}\right)] \end{aligned} \quad (33)$$

2.6 Romberg-Extrapolation

Auch für die Integration gibt es einen Extrapolations-Ansatz, welcher erlaubt aus einigen Anfangsnäherungen für ein bestimmtes Integral einen genaueren Wert zu extrapolieren. Es kann das exakte Schemata des h- bzw. h^2 Algorithmus verwendet werden. **Voraussetzung** ist, dass wir aus der bekannten Quadraturformel entweder eine Fehlerentwicklung in h oder h^2 haben. → h gibt jeweils die Schrittweise an. Die Trapezformel hat eine Fehlerentwicklung in geraden Potenzen der Schrittweite. → der h^2 -Algorithmus ist anwendbar.

Satz (Romberg-Extrapolation). Für die summierte Trapezregel $Tf(h)$ zur näherungsweisen Berechnung von

$$I(f) = \int_a^b f(x)dx \text{ gilt:}$$

Sei $T_{i0} = Tf\left(\frac{b-a}{2^i}\right)$ für $i = 0, 1, \dots, m$. Dann sind durch die Rekursion

$$T_{ik} = \frac{4^k * T_{i+1,k-1} - T_{i,k-1}}{4^k - 1} \quad (34)$$

für $k = 1, 2, \dots, m$ und $i=0,1,\dots,m-k$ Näherungen der Fehlerordnung $2k+2$ gegeben. Diese Methode heisst

Romberg-Extrapolation. Die verwendete Schrittweitenfolge $h_i = \frac{b-a}{2^i}$ heisst auch **Romberg-Folge**

Erkenntnis:

Man kann das Schema aus der Differation nochmals verwenden. Dafür berechnen wir die summierte Trapezformel für die Schrittweiten $h_i = \frac{b-a}{2^i}$ ($i=0,1,2,\dots,m$) → bspw. $h, \frac{h}{2}, \frac{h}{4}, \frac{h}{8}$ ($m=3$). Daraus erhalten wir $T_{00}, T_{10}, T_{20}, T_{30}$ und können anschliessend wieder extrapolieren. Dabei muss die Anzahl $n = 2^i$ der Intervalle für die Berechnung von T_{i0} angepasst werden.

D_{i0}	D_{i1}	D_{i2}	D_{i3}
D_{00}	$\frac{4D_{10} - D_{00}}{3} = D_{01}$		
D_{10}		$\frac{16D_{11} - D_{01}}{15} = D_{02}$	
D_{20}	$\frac{4D_{20} - D_{10}}{3} = D_{11}$	$\frac{64D_{12} - D_{02}}{63} = D_{03}$	
D_{30}	$\frac{4D_{30} - D_{20}}{3} = D_{21}$	$\frac{16D_{21} - D_{11}}{15} = D_{12}$	

Abbildung 6: Schema für Romberg Extrapolation

3 Einführung in gewöhnliche Differentialgleichungen

Einleitender Text

3.1 Lernziele

Sie kennen die Definitionen einer gewöhnlichen Differentialgleichung n-ter Ordnung und eines Anfangswertproblems

Sie verstehen das Prinzip eines Richtungsfeldes und können Richtungsfelder interpretieren und zeichnen

Sie können gewöhnliche Differentialgleichung lösen basierend auf dem Euler-Verfahren, dem modifizierten Euler-Verfahren, dem Mittelpunkt-Verfahren und den Runge-Kutta Verfahren

Sie können die hier vorgestellten Mehrschrittverfahren anwenden

Sie können Differentialgleichungen n-ter Ordnung in ein System von n Differentialgleichungen 1. Ordnung umwandeln und dieses System lösen

Sie kennen den Begriff Stabilität

3.2 Problemstellung

In vielen Prozessen (bspw. Natur oder Technik) spielt die zeitliche Änderungen von beobachtbaren Grösse $\rightarrow y(t)$ eine wichtige Rolle. Häufig ist die zeitliche Änderung einer solchen Grösse, also $\frac{dy}{dt}$, abhängig von der Grösse $y(t)$. D.h. wir können $\frac{dy}{dt}$ durch eine Funktion f ausdrücken, die von der Zeit t aber eben auch von der eigentlichen Grösse $y(t)$ abhängt:

$$\frac{dy}{dt} = f(t, y(t)) \quad (35)$$

→ auch 'gewöhnliche Differentialgleichung 1. Ordnung' genannt. 1. Ordnung, da die erste Ableitung auftritt. Beispiel hierfür wäre der radioaktiver Zerfall.

Definition 5 (Gewöhnliche Differentialgleichung (kurz:DGL) n-ter Ordnung). Eine Gleichung, in der Ableitungen einer unbekannten Funktion $y = y(x)$ bis zur n-ten Ordnung auftreten, heisst eine *gewöhnliche Differentialgleichung n-ter Ordnung*. Sie hat die explizite Form

$$y^{(n)}(x) = f(x, y(x), y'(x), \dots, y^{(n-1)}(x)) \quad (36)$$

Gesucht sind die Lösungen $y = y(x)$ dieser Gleichung, wobei die Lösungen y auf einem Intervall

$$a, b$$

definiert sein sollen, $y : [a, b] \rightarrow \mathbb{R}$

Definition 6 (Anfangswertproblem). Bei einem Anfangswertproblem für eine Differentialgleichung n-ter Ordnung werden die Lösungsfunktion $y = y(x)$ noch n Werte vorgeschrieben, nämlich der Funktionswert an einer bestimmten Stelle x_0 sowie die Werte der ersten n-1 Ableitungen an der gleichen Stelle.

Für die hier betrachteten Differentialgleichungen 1. und 2. Ordnung heisst das:

- *Differentialgleichung 1. Ordnung:* Gesucht ist diejenige spezifische Lösungskurve $y = y(x)$, die durch den vorgegebenen Punkt $P = (x_0, (y(x_0)))$ verläuft.
- *Differentialgleichung 2. Ordnung:* Gesucht ist diejenige spezifische Lösungskurve $y = y(x)$ die durch den vorgegebenen Punkt $P = (x_0, (y(x_0)))$ verläuft und im Punkt x_0 die vorgegebenen Steigung $y'(x_0) = m$ besitzt.

3.3 Richtungsfelder für Differentialgleichungen 1. Ordnung

Die Differentialgleichung 1. Ordnung ihr Ihre Lösungen lassen sich mit Hilfe von sogenannten Richtungsfeldern veranschaulichen.

Ausgangspunkt ist die geometrische Interpretation der Gleichung:

$$y'(x) = f(x, y(x)) \quad (37)$$

Aus dieser Gleichung ist ersichtlich, dass es einen Zusammenhang zwischen der Steigung $y'(x)$, der gesuchten Funktion $y(x)$ für einen gegebenen Punkt x und dem Punkt $(x, y(x))$.

Wenn wir uns dies in einer (x, y) -Ebene veranschaulichen hat es folgende Auswirkung:

Wenn der Graph einer Lösung y durch einen Punkt (x, y) verläuft, so muss er dort die Steigung y' haben. Dementsprechend können wir in der (x, y) -Ebene an einem belieben (x, y) -Punkt die Steigung $y'(x) = f(x, y(x))$ ausrechnen und durch einen kleinen Pfeil graphisch darstellen. Als Resultat erhalten wir mit all den Pfeilen zu jeden jeweiligen Punkten die Richtung der Lösungskurve.

→ Dies wird auch Richtungsfeld der Differentialgleichung genannt. Dabei verlaufen die Lösungskurven der Gleichung stets tangential zu den Pfeilen im Richtungsfeld.

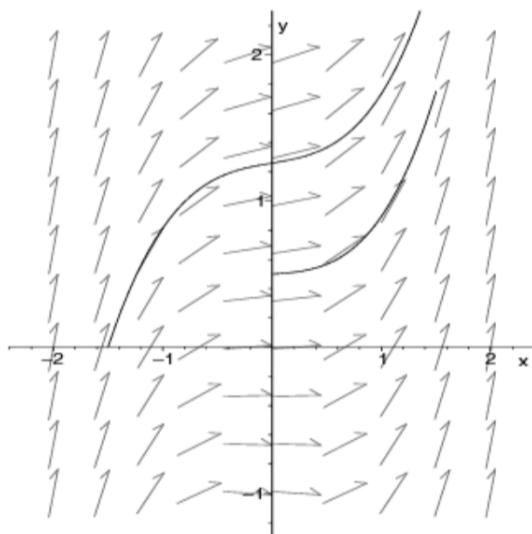


Abbildung 7: Beispiel für ein Richtungsfeld für die Gleichung $y'(x) = f(x, y(x)) = x^2 + 0.1 * y(x)$

Die Idee vieler numerischer Verfahren zur Lösung von Anfangswertproblemen ist es nun, dem Richtungsfeld möglichst genau zu folgen. Jedoch besteht, dass Problem, dass ein solches Richtungsfeld aus unendlich vielen Punkten besteht. Damit man hier trotzdem zu einer Lösung kommt, braucht man eine *Diskretisierung*.

3.4 Das Euler-Verfahren

Das einfachste numerische Einschrittsverfahren zur Lösung von Anfangswertproblem ist das klassische Euler-Verfahren. Jedoch hat es einige Einschränkungen und wird in der Praxis nicht wirklich verwendet. Verbesserte Verfahren sind das modifizierte Euler-Verfahren oder das Mittelpunkt-Verfahren, welche in den nächsten Kapitel behandelt werden.

3.4.1 Das klassische Euler-Verfahren

Wir gehen vom folgenden Anfangswertproblem aus:

$$\frac{dy}{dx} = f(x, y(x)) \text{ mit } y(a) = y_0 \quad (38)$$

Entsprechend ist $x_0 = a$ die einzige Stelle, welche wir exakt bestimmen können (da $y(x_0) = y_0$ vorgegeben ist.). Dies ist ebenfalls auch die einzige Stelle wo wir y' exakt kennen $\rightarrow y'(x_0) = f(x_0, y(x_0)) = f(x_0, y_0)$.

Dem Startpunkt $(x_0, y(x_0))$ hat entsprechend eine hohe Bedeutung, denn alle weiteren Werte werden Fehler enthalten, da wir dem Richtungsfeld nicht exakt folgen können.

Es wird angenommen, dass die Steigung in den Punkten um $(x_0, y(x_0))$ nur wenig von $y'(x_0)$ abweicht.

Wir können dann die Steigung der Sekanten durch die Punkte $(x_0, y(x_0))$ und $(x_1, y(x_1))$ mit der Steigung $y'(x_0)$ gleichsetzen und nach $y(x_1)$ auflösen:

$$y'(x_0) = f(x_0, y(x_0)) \approx \frac{y(x_1) - y(x_0)}{x_1 - x_0} = \underbrace{\frac{y(x_1) - y(x_0)}{h}}_{D_1y(x_0, h)} \Rightarrow x(x_1) \approx y(x_0) + h * f(x_0, y(x_0)) = y_1 \quad (39)$$

Wir folgen sozusagen der Tangente im Punkt $(x_0, y(x_0))$ um die Schritte weite h in der Hoffnung, dass wir dann genügend nahe an den Punkt $y(x_1)$ rankommen. Dabei verwenden wir im Prinzip nichts anderes als die Vorwärtsdifferenz D_1 . Wenn die Schrittweite genügend klein ist, sollte auch der Fehler entsprechend klein sein und umgekehrt.

Dieser Schritt lässt sich nun ausgehend von der Näherung $y_1 \approx y(x_1)$ wiederholen, wir erhalten dann $y(x_2) \approx y_1 + h * f(x_1, y_1) = y_2$ usw.

Definition 7 (Algorithmus Euler-Verfahren). Gegeben sei für $x \in$

$$a, b$$

das Anfangswertproblem:

$$y' = f(x, y) \text{ mit } y(a) = y_0 \quad (40)$$

Das Euler-Verfahren zur numerischen Lösung lautet:

$$\begin{aligned} x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + h * f(x_i, y_i) \end{aligned} \quad (41)$$

Wobei $x_0 = a, x_i = a + i * h$ für $i=0, \dots, n-1$ $n \in \mathbb{N}$ und $h = \frac{b-a}{n}$

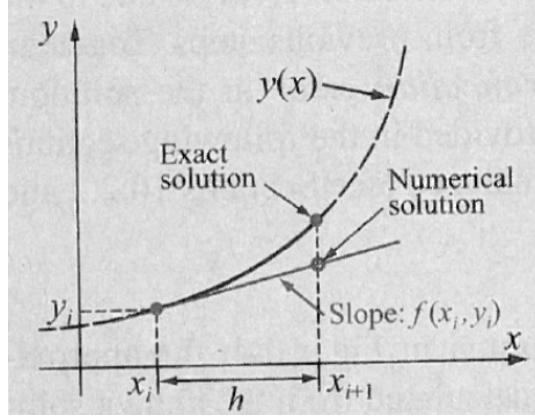


Abbildung 8: Euler-Verfahren zur Berechnung von y_{i+1} an der Stelle x_{i+1} basierend auf der Tangente mit der Steigung $f(x_i, y_i)$ an der Stelle x_i

3.4.2 Das Mittelpunkt-Verfahren

Das klassische Euler-Verfahren beruht darauf, die Steigung $y'(x_i)$ am Punkt (x_i, y_i) zu berechnen und der Tangente in diesem Punkt eine Schrittweite h zu folgen. Beim Mittelpunktverfahren folgt man der Tangente nur die halbe Schrittweite $\frac{h}{2}$ und berechnet beim Mittelpunkt des Intervalls $x_{\frac{h}{2}} = x_i + \frac{h}{2}$ die neue Steigung $y_{\frac{h}{2}}$:

$$y_{\frac{h}{2}} = y_i + \frac{h}{2} * f(x_i, y_i) \quad (42)$$

Anschliessend geht man zurück an den Ausgangspunkt (x_i, y_i) und benutzt die berechnete Steigung $y_{\frac{h}{2}}$ für einen ganzen Schritt der Schrittweite h .

Definition 8 (Algorithmus: Mittelpunkt-Verfahren). Gegeben sei für $x \in [a, b]$ das Anfangswertproblem

$$y' = f(x, y) \text{ mit } y(a) = y_0 \quad (43)$$

Das Mittelpunktverfahren zur numerischen Lösung lautet:

$$\begin{aligned} x_{\frac{h}{2}} &= x_i + \frac{h}{2} \\ y_{\frac{h}{2}} &= y_i + \frac{h}{2} * f(x_i, y_i) \\ x_{x+1} &= x_i + h \\ y_{x+1} &= y_i + h * f(x_{\frac{h}{2}}, y_{\frac{h}{2}}) \end{aligned} \quad (44)$$

Wobei $x_0 = a$, $x_i = a + i * h$ für $i = 0, \dots, n-1$ ($n \in \mathbb{N}$) und $h = \frac{b-a}{n}$

Beispiel Mittelpunkt-Verfahren

Wir berechnen die numerische Lösung des Anfangsproblems mit dem Mittelpunkt-Verfahren:

$$y'(t) = f(t, y(t)) = t^2 + 0.1 * y(t) \quad (45)$$

mit $y(-1.5) = 0$ auf dem Intervall $[a, b] = [-1.5, 1.5]$ und $n = 5$

$$\begin{aligned} n = 5 \Rightarrow h &= \frac{b-a}{n} = \frac{1.5 - (-1.5)}{5} = 0.6 \\ t_i &= a + ih = -1.5 + i * 0.6 (i = 0, 1, \dots, 5) \\ y_{i+1} &= y_i + h f(t_{\frac{h}{2}}, y_{\frac{h}{2}}) = y_i + h (t_{\frac{h}{2}}^2 + 0.1 y_{\frac{h}{2}}) \end{aligned} \quad (46)$$

i	t_i	y_i	$t_{h/2} = t_i + \frac{h}{2}$	$y_{h/2} = y_i + \frac{h}{2} \cdot (t_i^2 + 0.1y_i)$	$t_{i+1} = t_i + h$	$y_{i+1} = y_i + h \cdot (t_{h/2}^2 + 0.1y_{h/2})$
0	-1.5	0	-1.2	0.6750	-0.9	0.9045
1	-0.9	0.9045	-0.6	1.1746	-0.3	1.1910
2	-0.3	1.1910	0.0	1.2537	0.3	1.2662
3	0.3	1.2662	0.6	1.3312	0.9	1.5621
4	0.9	1.5621	1.2	1.8519	1.5	2.5372
5	1.5	2.5372	-	-	-	-

Abbildung 9: Tabelle für das Mittelpunkt-Verfahren

3.4.3 Das modifizierte Euler-Verfahren (auch Heun-Verfahren genannt)

Beim modifizierten Euler-Verfahren verwendet man den Durchschnitt zweier Steigungen.

Zuerst folgt man wie beim klassischen Euler-Verfahren ausgehend vom Punkt (x_i, y_i) der Tangente der Steigung $f(x_i, y_i)$ einen ganzen Schritt und erhält den neuen Punkt $(x_{i+1}, y_{i+1}^{Euler})$ wobei $y_{i+1}^{Euler} = y_i + h * f(x_i, y_i)$. Anschliessend berechnet man im Punkt $(x_{i+1}, y_{i+1}^{Euler})$ die nächste Steigung $f(x_{i+1}, y_{i+1}^{Euler})$. Danach wird der Durchschnitt der beiden Steigungen genommen und vom Ausgangspunkt (x_i, y_i) ein Schritt mit Schrittänge h gemacht zum neuen Punkt (x_{i+1}, y_{i+1}) .

Definition 9 (Algorithmus: Modifiziertes Euler-Verfahren). Gegeben sei für $x \in [a, b]$ das Anfangswertproblem

$$y' = f(x, y) \text{ mit } y(a) = y_0 \quad (47)$$

Das modifizierte Euler-Verfahren zur numerischen Lösung lautet:

- a) Führe das klassische Euler-Verfahren durch und speichere die erste Tangentensteigung in der Variable k_1

$$\begin{aligned} x_{i+1} &= x_i + h \\ y_{i+1}^{Euler} &= y_i + h * f(x_i, y_i) \\ k_1 &= f(x_i, y_i) \end{aligned} \quad (48)$$

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

$$k_2 = f(x_{i+1}, y_{i+1}^{Euler}) \quad (49)$$

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

$$\begin{aligned} x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + h * \frac{(k_1 + k_2)}{2} \end{aligned} \quad (50)$$

- d) Wiederhole diese Schritte ausgehend von $x_0 = a$ für $x_i = a + ih$ mit $i=0, \dots, n-1$ ($n \in \mathbb{N}$) und $h = \frac{b-a}{n}$

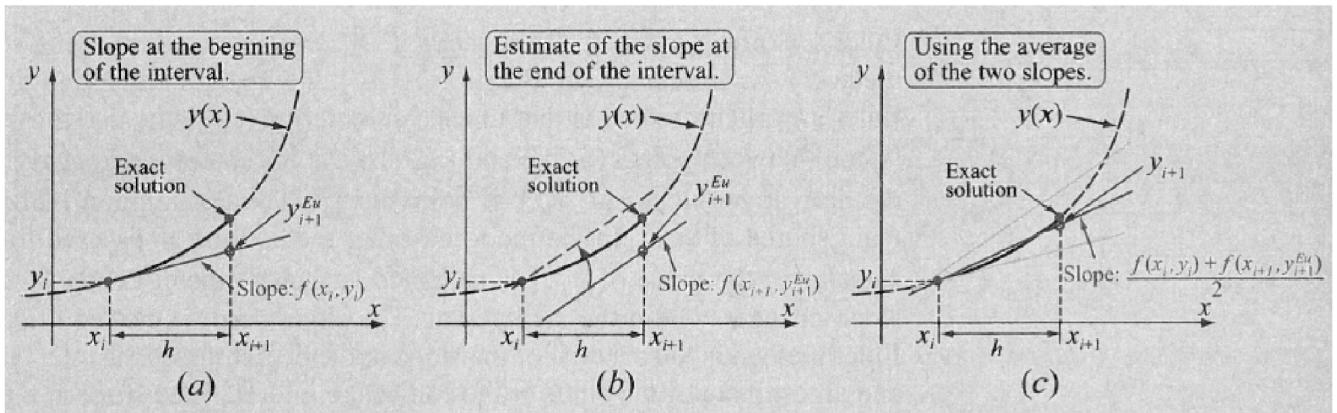


Abbildung 10: Überblick der eulerischen Verfahren

Beispiel modifiziertes Euler-Verfahren

Wie berechnen die numerische Lösung des Anfangsproblems $y'(t) = t(t, y(t)) = t^2 + 0.1 * y(t)$ mit $y(-1.5) = 0$ auf dem Intervall $[a, b] = [-1.5, 1.5]$ und $n = 5$, mit dem modifizierten Euler-Verfahren

$$\begin{aligned} n = 5 \Rightarrow h &= \frac{b - a}{n} = \frac{1.5 - (-1.5)}{5} = 0.6 \\ t_i &= a + ih = -1.5 + i * 0.6 \quad (i = 0, 1, \dots, 5) \\ y_{i+1} &= y_i + h * \frac{(k_1 + k_2)}{2} \end{aligned} \tag{51}$$

i	t	y_{exakt}	y_{euler}	$y_{\text{mittelpunkt}}$	$y_{\text{mod. euler}}$
0	-1.5	0	0	0	0
1	-0.9	0.9135	1.3500	0.9045	0.9585
2	-0.3	1.2133	1.9170	1.1910	1.3023
3	0.3	1.3069	2.0860	1.2662	1.4384
4	0.9	1.6267	2.2652	1.5621	1.7989
5	1.5	2.6318	2.8871	2.5372	2.8426

Abbildung 11: Vergleich der kennengelernten Verfahren

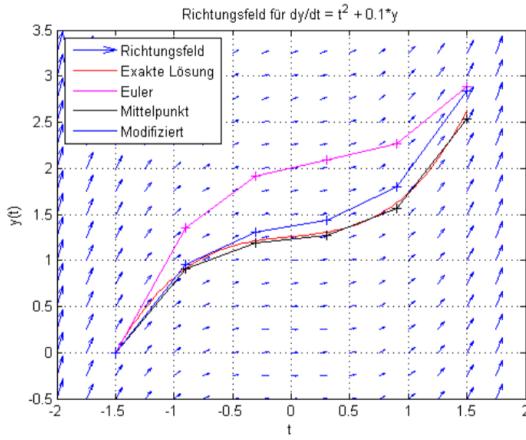


Abbildung 12: Vergleich der kennengelernten Verfahren anhand des Richtungsfeldes

3.5 Die Fehlerordnung eines Verfahrens

Wie in der obigen Abbildung ersichtlich, kann sich der numerische Wert stark von der exakten Lösung unterscheiden. Damit wir dies genauer untersuchen können, führen wir die Begriffe *lokalen Fehler* und *globalen Fehler* sowie die jeweilige *Fehlerordnung* ein.

Definition 10 (lokaler Fehler / Konsistenzordnung). Sei $y'(x) = f(x, y(x))$ eine Differentialgleichung mit der Anfangsbedingung $y(x_i) = y_i$ und der exakten Lösung $y(x)$.

Sei y_{i+1} der mit einem numerischen Näherungsverfahren mit der Schrittweite h berechnete Näherungswert $y(x_{i+1})$, wobei $x_{i+1} = x_i + h$. Dann ist der **lokale Fehler** (also nach einer Iteration) definiert als die Differenz zwischen dem exakten Wert und der Näherung:

$$\varphi(x_i, h) := y(x_{i+1} - y_{i+1}) \quad (52)$$

Ein numerisches Verfahren hat die **Konsistenzordnung p** falls gilt:

$$|\varphi(x, h)| \leq C * h^{p+1} \quad (53)$$

für genügend kleine h und einer Konstante $C > 0$, die von der Differentialgleichung abhängt.

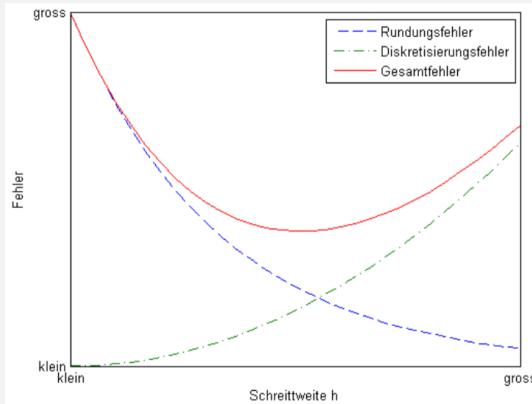


Abbildung 13: Verhalten des Rundungsfehlers und des Diskretisierungsfehlers

Definition 11 (globaler Fehler / Konvergenzordnung). Sei $y'(x) = f(x, y(x))$ eine Differentialgleichung mit der Anfangsbedingung $y(x_0) = y_0$ und der exakten Lösung $y(x)$. Sei y_n der mit einem numerischen Näherungsverfahren mit der Schrittweite h berechnete Näherungswert für $y(x_n)$, wobei $x_n = x_0 + nh$. Dann ist der Gesamtfehler (also nach n Iterationen) bzw. der **globale Fehler** definiert als die Differenz zwischen dem exakten Wert und der Näherung:

$$y(x_n) - y_n \quad (54)$$

Ein numerisches Verfahren hat die **Konvergenzordnung p** falls gilt:

$$|y(x_n) - y_n| \leq C * h^P \quad (55)$$

für genügend kleine h und einer Konstante $C > 0$, die von der Differentialgleichung abhängt.

Bemerkungen

- Der lokale Fehler ist ein Mass für die Abweichung von der exakten Lösung nach einer Iteration, während der globale Fehler die Abweichung von der exakten Lösung nach n Iterationen misst.
- Die hier gemachten Definitionen beziehen sich nur auf den Diskretisierungsfehler. Es kommt noch der Rundungsfehler hinzu.
- Für die hier betrachteten Verfahren ist die Konsistenz- und Konvergenzordnung jeweils identisch
- Verwendbar sind nur Verfahren mit der Konvergenzordnung $p \leq 1$, da dann der globale Fehler gegen Null

strebt:

$$\lim_{h \rightarrow 0} |y(x_n) - y_n| \leq \lim_{h \rightarrow 0} C * h^P = 0 \quad (56)$$

Das bedeutet, der Diskretisierungsfehler wird theoretisch beliebig klein, wenn die Schrittweite h beliebig klein wird. In der Praxis verhindern natürlich Rundungsfehler, dass eine beliebige Genauigkeit erzielt werden kann.

3.6 Runge-Kutta Verfahren

Im Einzelschrittverfahren wird die Steigung im Richtungsfeld verwendet gemäss:

$$\begin{aligned} x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + \text{Steigung} * h \end{aligned} \quad (57)$$

Das klassische Euler-Verfahren verwendet einen Punkt, um die Steigung zu berechnen und hat die Konsistenz- und Konvergenzordnung $p = 1$. Das Mittelpunkt-Verfahren und das modifizierte Euler-Verfahren verwenden dazu zwei Punkte und haben die Konsistenz- und Konvergenzordnung $p = 2$.

Offenbar kann durch die Hinzunahme von zusätzlichen Punkten zur Berechnung eines Durchschnitt-Werts für die Steigung die Genauigkeit eines Einschrittverfahrens verbessert werden.

3.6.1 Das klassische vierstufige Runge-Kutta Verfahren

Definition 12 (Algorithmus: klassisches vierstufige Runge-Kutta Verfahren). Gegeben sei für $x \in [a, b]$ das Anfangswertproblem

$$y' = f(x, y) \text{ mit } y(a) = y_0 \quad (58)$$

Das klassische Runge-Kutta zur numerischen Lösung lautet

$$\begin{aligned} k_1 &= f(x_i, y_i) \\ k_2 &= f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right) \\ k_3 &= f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_2\right) \\ k_4 &= f(x_i + h, y_i + hk_3) \\ x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + h * \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \quad (59)$$

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

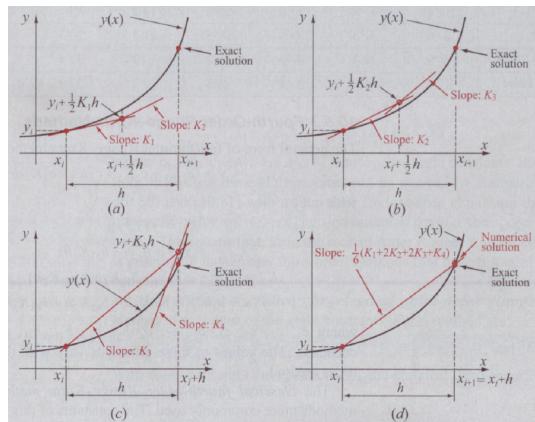


Abbildung 14: Klassisches vierstufiges Runge-Kutta-Verfahren

Das klassische vierstufige Runge-Kutta Verfahren hat die Konsistenz- und Konvergenzordnung $p = 4$

Beispiel Runge-Kutta Verfahren

Wir berechnen wieder die numerische Lösung des Anfangswertproblems

$$y'(t) = f(t, y(t)) = t^2 + 0.1 * y(t) \quad (60)$$

mit $y(-1.5) = 0$ auf dem Intervall $[a, b] = [-1.5, 1.5]$ und $n = 5$, diesmal aber mit klassischen Runge-Kutta Verfahren.

Klassisches Runge-Kutta Verfahren

i	t_i	y_{exakt}	y_i	k_1	k_2	k_3	k_4	y_{i+1}
0	-1.5	0	0	2.2500	1.5075	1.485	0.8991	0.9135
1	-0.9	0.9135	0.9135	0.9013	0.4784	0.4657	0.2093	1.2133
2	-0.3	1.2133	1.2133	0.2113	0.1277	0.1252	0.2188	1.3069
3	0.3	1.3069	1.3069	0.2207	0.4973	0.5056	0.9710	1.6267
4	0.9	1.6267	1.6267	0.9727	1.6318	0.1651	2.512	2.6318
5	1.5	2.6318	2.6318	-	-	-	-	-

Abbildung 15: Klassisches Runge-Kutta-Verfahren

Der Vergleich mit den Werten der exakten Lösung y_{exakt} bei dieser Genauigkeit keinen Unterschied mehr

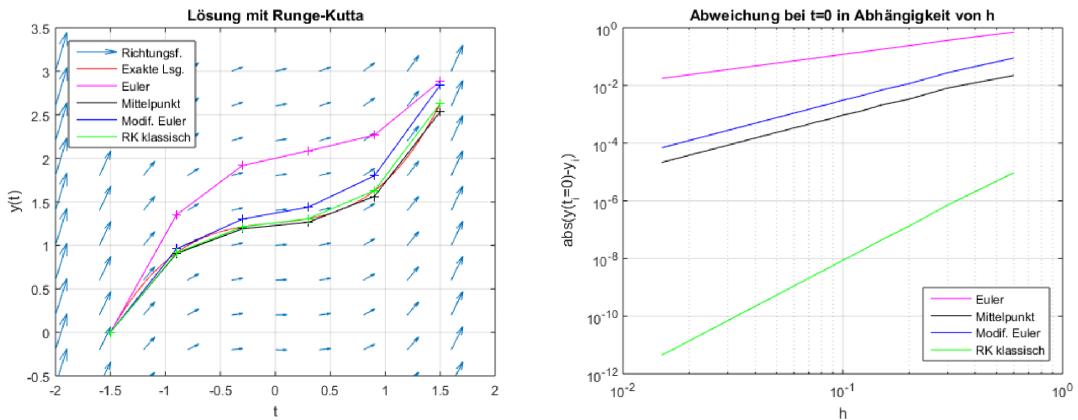


Abbildung 16: Vergleich von verschiedenen Verfahren inkl. Runge-Kutta (links)

3.6.2 Das allgemeine s-stufige Runge-Kutta Verfahren

Definition 13 (Allgemeines s-stufiges Runge-Kutta Verfahren). Ein allgemeines (explizites) s-stufiges Runge-Kutta Verfahren ist gegeben durch die Formeln

$$\begin{aligned} k_n &= f(x_i + c_n h, y_i + h \sum_{m=1}^{n-1} a_{nm} k_m) \quad \text{für } n = 1, \dots, s \\ y_{i+1} &= y_i + h \sum_{n=1}^s b_n k_n \end{aligned} \quad (61)$$

Hierbei ist $s \in \mathbb{N}$ die Stufenzahl und a_{bm}, b_n, c_n sind Konstanten. Die Konsistenz- und Konvergenzordnung hängt von der Wahl dieser Konstanten ab

Bemerkung: Man notiert die Koeffizienten meist in der Form

c_1					
c_2	a_{21}				
c_3	a_{31}	a_{32}			
\vdots					
c_n	a_{n1}	a_{n2}	\dots	$a_{n,n-1}$	
\vdots					
c_s	a_{s1}	a_{s2}	\dots	$a_{s,s-1}$	
	b_1	b_2	\dots	b_{s-1}	b_s

Abbildung 17: Koeffizienten Formen

3.7 Mehrschrittverfahren

Bei Mehrschrittverfahren im Gegensatz zu den Einschrittverfahren für die Berechnung von y_{i+1} nicht nur der vorhergehende Punkt (x_i, y_i) benötigt, sondern mindestens zwei (also z.B. (x_i, y_i) und (x_{i-1}, y_{i-1})) oder noch mehr vorhergehende Punkte.

Wir beschreiben hier die Familie der (expliziten) Adams Bashforth Methoden.

Die Lösung der DGL

$$y' = f(x, y) \quad (62)$$

wird dabei durch Integration über das Intervall $[x_i, x_{i+1}]$ erreicht:

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(x, y) dx \quad (63)$$

Dabei wird $f(x,y)$ durch ein Polynom angenähert, welches die Werte von $f(x,y)$ bei (x_i, y_i) und vorhergehenden Punkten interpoliert. Je nach Ordnung des verwendeten Polynoms erhält man eine andere Iterationsvorschrift.

3.7.1 Adams-Bashforth Methode 2. und 3. Ordnung

Verwendet man die Punkte (x_i, y_i) und (x_{i-1}, y_{i-1}) , dann ist das interpolierende Polynom vom Grad 1 (eine Gerade) und es gilt:

$$f(x, y) = f(x_i, y_i) + \frac{f(x_i, y_i) - f(x_{i-1}, y_{i-1})}{h} * (x - x_i) \quad (64)$$

mit $h = x_{i+1} - x_i$. Durch die Integration erhalten wir dann:

$$\begin{aligned} y_{i+1} &= y_i + \int_{x_i}^{x_{i+1}} f(x, y) dx \\ &= y_i + \int_{x_i}^{x_{i+1}} f(x, y) dx + \int_{x_i}^{x_{i+1}} \frac{f(x_i, y_i) - f(x_{i-1}, y_{i-1})}{h} * (x - x_i) dx \\ &= y_i + f(x_i, y_i) \int_{x_i}^{x_{i+1}} dx + \frac{f(x_i, y_i) - f(x_{i-1}, y_{i-1})}{h} \int_{x_i}^{x_{i+1}} (x - x_i) dx \\ &= y_i + f(x_i, y_i) * (x_{i+1} - x_i) + \frac{f(x_i, y_i) - f(x_{i-1}, y_{i-1})}{h} * \frac{1}{2} (x_{i+1} - x_i)^2 \\ &= y_i + f(x_i, y_i) * h + (f(x_i, y_i) - f(x_{i-1}, y_{i-1})) * \frac{1}{2} h \\ &= y_i + \frac{h}{2} (3f(x_i, y_i) - f(x_{i-1}, y_{i-1})) \end{aligned} \quad (65)$$

Die Iteration

$$y_{i+1} = y_i + \frac{h}{2}(3f(x_i, y_i) - f(x_{i-1}, y_{i-1})) \quad (66)$$

ist die Adams-Bashforth Methode 2. Ordnung. Denn diese verwendet zwei Punkte. Diese Methode kann erst ab dem zweiten Punkt (x_2, y_2) verwendet werden, da die Punkte (x_0, y_0) und (x_1, y_1) für die Iteration bekannt sein müssen. Dementsprechend muss für die Berechnung von (x_1, y_1) ein Einschrittverfahren verwendet werden.

Durch die Verwendung eines Polynoms vom Grad 3 (Parabel) kann $f(x, y)$ in den Punkten (x_i, y_i) , (x_{i-1}, y_{i-1}) und (x_{i-2}, y_{i-2}) interpoliert werden und wir erhalten die Adams-Bashforth Methode 3. Ordnung, welche ab (x_3, y_3) einsetzbar ist:

$$y_{i+1} = y_i + \frac{h}{12}(23f(x_i, y_i) - 16f(x_{i-1}, y_{i-1}) + 5f(x_{i-2}, y_{i-2})) \quad (67)$$

3.7.2 Adams Bashforth Methoden höherer Ordnung

Durch die Verwendung eines Interpolationspolynoms vom Grad s erhält man die Adams Bashforth Methode der Ordnung $s+1$ über die generelle Formel:

$$y_{i+} = y_1 + \sum_{j=0}^s b_j f(x_{i-j}, y_{i-j}) \quad (68)$$

mit den Koeffizienten

$$b_j = \frac{(-1)^j}{j!(s-j)!} \int_0^1 \prod_{k=0, k \neq j}^s (u+k) du \quad j = 0, 1, \dots, s \quad (69)$$

3.8 Erweiterung auf Systeme von Differentialgleichungen

Die bisherigen Systeme sind nur für Differentialgleichungen 1. Ordnung anwendbar.

Wir zeigen nun:

- i) Wie aus einer Differentialgleichung k -ter Ordnung ein System von k Differentialgleichungen 1. Ordnung gemacht wird
- ii) wie dieses System anschliessend mit den uns bereits bekannten Verfahren gelöst werden kann

3.9 Zurückführen einer DGL k -ter Ordnung auf k DGL 1. Ordnung

evtl Beispiel 7.9 S.129 einfügen

Definition 14 (Rezept für das Zurückführen auf ein System erster Ordnung).

1. Die Differentialgleichung nach den höchsten vorkommenden Ableitungen der unbekannten Funktion auflösen
2. Neue Funktionen für die unbekannten Funktionen und deren Ableitung bis Ordnung der höchsten Ableitung minus 1 einführen
3. Das System erster Ordnung durch Ersetzen der höheren Ableitungen durch die neuen Funktionen aufstellen
4. Das entsprechende Anfangswertproblem in vektorieller Form aufschreiben

3.10 Lösen eines Systems von k DGL 1. Ordnung

evtl Beispiel 7.9 S.130 weiterführen

Definition 15 (Rezept für das Lösen eines Systems von k DGL 1. Ordnung). Ist ein Lösungsverfahren

$$\begin{aligned} x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + \text{Steigung} * h \end{aligned} \quad (70)$$

für die eindimensionale Gleichung

$$x'(x) = f(x, y(x)), y(x_0) = y_0 \quad (71)$$

definiert, so kann es völlig analog erweitert werden als

$$\begin{aligned} x_i &= x_i + h \\ \mathbf{y}^{(i+1)} &= \mathbf{y}^{(i+1)} + \text{Steigung} * h \end{aligned} \quad (72)$$

für ein System

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}(x)) \text{ mit } \mathbf{y}(x_0) = \mathbf{y}^{(0)} \quad (73)$$

(wobei wie üblich ein hochgestellter Index in Klammern $y^{(i)}$ einen Vektor aus \mathbb{R}^n nach der i-ten Iteration bezeichnet).

Dabei werden ersetzt:

- \mathbf{y}' durch den Vektor \mathbf{y}' der Ableitungen der einzelnen Komponenten
- $\mathbf{f}(x, \mathbf{y}(x))$ durch die vektorwertige Funktion $\mathbf{f}(x, \mathbf{y}(x))$ und
- die skalare Anfangsbedingung $y(x_0) = y_0$ durch die Anfangsbedingung $\mathbf{y}(x_0) = \mathbf{y}^{(0)}$

Es ist dann also:

$$\mathbf{y}(x) = \begin{pmatrix} y_1(x) \\ y_2(x) \\ \vdots \\ y_n(x) \end{pmatrix}, \mathbf{y}' = \begin{pmatrix} y'_1(x) \\ y'_2(x) \\ \vdots \\ y'_n(x) \end{pmatrix}, \mathbf{f}(x, \mathbf{y}(x)) = \begin{pmatrix} f_1(x, \mathbf{y}(x)) \\ f_2(x, \mathbf{y}(x)) \\ \vdots \\ f_n(x, \mathbf{y}(x)) \end{pmatrix}, \mathbf{y}(x_0) = \mathbf{y}^{(0)} = \begin{pmatrix} y_1(x_0) \\ y_2(x_0) \\ \vdots \\ y_n(x_0) \end{pmatrix} \quad (74)$$

Bemerkung: Wurd das System erstellt, um eine DGL k-ter Ordnung zu lösen, so finden sich die Lösung $y(x)$ in der ersten Komponente des Vektors $\mathbf{y}(x)$, als $y(x) \approx y_1(x)$

3.11 Stabilität

Der Fehler bei der Lösung einer DGL setzt sich aus dem vom benutzten Verfahren abhängigen Diskretisierungsfehler und dem vom Rechner abhängigen Rundungsfehler zusammen. In gewissen Situation kann der numerische Fehler im Verlauf der Iteration unbeschränkt gross werden (unabhängig der Schrittweite h). Dabei spricht man von *Instabilität* bzw. von einer instabilen Lösung. Die Stabilität einer Lösung hängt von drei Faktoren ab:

- i.) dem benutzen Verfahren
- ii.) der Schrittweite h
- iii.) dem spezifischen Anfangswertproblem

Dabei gibt es verschiedene Methoden, um das Problem zu untersuchen, z.B.:

- Durchrechnen der Lösung einmal mit single- und einmal mit double-precision mit anschliessendem Vergleich.
→ Dies kann Aussagen möglich machen zur Akkumulation von Rundungsfehlern
- Variation der Schrittweite h . → dies erlaubt Aussagen zum Diskretisierungsfehler

- Vergleich der Resultate erzielt mit einem Lösungsverfahren höherer Ordnung und einem Lösungsverfahren niedriger Ordnung
- Vergleich der Resultate eines numerischen Lösungsverfahrens mit der analytischen Lösung, sofern diese bekannt ist.

Den letzten Punkt schauen wir genauer an:

Das Anfangsproblem

$$y' = -\alpha y, y(0) = y_0 = 1 (\alpha > 0) \quad (75)$$

für welches wir die exakte Lösung kennen:

$$y(x) = e^{-\alpha x} \quad (76)$$

Das Euler-Verfahren liefert die numerische Lösung

$$y_{i+1} = y_i - h * \alpha y_i = <_i (1 - h\alpha) \quad (77)$$

Durch rekursives Einsetzen von $y_i = y_{i-1}(1 - h\alpha)$ erhalten wir:

$$y_{i+1} = y_i * (1 - h\alpha) = y_{i-1}(1 - h\alpha)^2 = y_{i-2}(1 - h\alpha)^3 = \dots = \underbrace{y_0}_{=1} (1 - h\alpha)^{i+1} \quad (78)$$

Da die exakte Lösung $y = e^{-\alpha x}$ streng monoton fallend ist, sollte auch die Näherungslösung streng monoton fallend sein, d.h. in diesem Fall

$$|1 - h\alpha| < 1 \quad (79)$$

und es folgt

$$0 < h\alpha < 2 \Rightarrow 0 < h < \frac{2}{\alpha} \quad (80)$$

Wir haben also eine obere Schranke für h hergeleitet bzw. eine Schrittweitenobergrenze für die die numerische Lösung stabil bleibt.

Definition 16 (Stabilitätsfunktion / Stabilitätsintervall). Kann bei der Anwendung eines Verfahrens auf die DGL $y' = -\alpha y$ die numerische Lösung in der Form

$$y_{i+1} = g(h\alpha) * y_i \quad (81)$$

geschrieben werden, so nennt man $g(z)$ die Stabilitätsfunktion des Verfahrens (mit $z = h\alpha$)

Das offene Intervall $z \in (0, \alpha)$, in dem $|g(z)| < 1$ gilt, bezeichnet man als das Stabilitätsintervall des Verfahrens

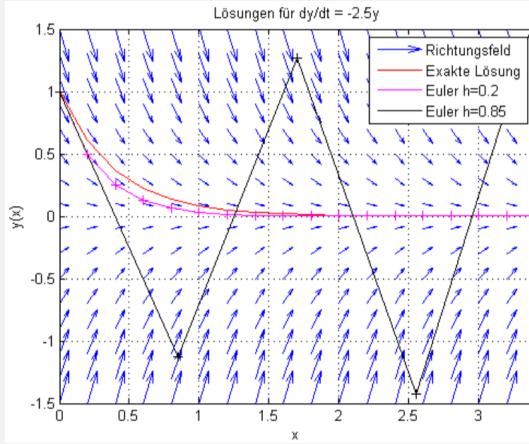


Abbildung 18: Euler-Verfahren für die Lösung der DGL $y' = -2.5y$ mit den Schrittweiten $h_1 = 0.2$ (stabil) und $h_2 = 0.85$ (instabil)

Bemerkungen:

- Die Stabilitätsfunktion für das Euler-Verfahren ist $g(z) = 1 + z$
- Man kann zeigen, dass die Stabilitätsfunktion eines s-stufigen (expliziten) Runge-Kutta Verfahrens ein Polynom vom Grad s ist.

3.12 Weitere Punkte

Zur Vollständigkeit werden weitere Punkte zur Differentialgleichungen betrachtet.

3.12.1 Implizite vs. explizite Verfahren

Bis anhin haben wir uns auf die Lösung von expliziten DGL der Form

$$y^{(n)}(x) = f(x, y(x), y'(x), \dots, y^{n-1}(x)) \quad (82)$$

beschränkt, d.h. die DGL ist nach der höchsten Ableitung aufgelöst. Wir haben demnach auch nur explizite Verfahren angewendet. Im Gegensatz dazu gibt es die impliziten DGL der Form:

$$F(x, y(x), y'(x), \dots, y^{(n-1)}(x), y^{(n)}(x)) = 0 \quad (83)$$

die nicht nach der höchsten Ableitung aufgelöst sind. Bei den zugehörigen impliziten Lösungsverfahren muss jeweils noch ein Nullstellenproblem gelöst werden (bspw. mit Newton-Verfahren), wie beim impliziten Euler-Verfahren

$$\begin{aligned} x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + h * f(x_{i+1}, y_{i+1}) \end{aligned} \quad (84)$$

da hier die gesuchte Lösung y_{i+1} auf beiden Seiten der Gleichung auftaucht. Diese Verfahren sind deshalb aufwendiger, aber auch stabiler als die expliziten Verfahren.

3.12.2 Steife DGL

Im Zusammenhang mit der Stabilitätsüberlegungen, redet man auch von sogenannten *steifen DGL*. Diese sind DGL mit stark schwankenden Zeitskalen t (oder Längeskalen x) der Lösung y . Zur Lösung solcher steifen DGL benötigen explizite Lösungsverfahren eine verschwindend kleine Schrittweite h , so dass die Lösung wegen langer Laufzeiten unpraktikabel oder wegen zunehmenden Rundungsfehler instabil wird. Steife DGL werden deshalb mit impliziten Verfahren gelöst.

3.12.3 Schrittweitensteuerung

Für DGL mit stark schwankenden Lösungsfunktionen $y(x)$ ist es effizienter, statt einer konstanten Schrittweite $h = \frac{b-a}{n} = const$ eine Variable Schrittweite h , anzuwenden, die klein ist in Bereichen grosser Krümmung der Lösung $y(x)$ und gross in Bereichen mit kleiner Krümmung. Hierzu sind Fehlerschätzungen für den lokalen Fehler nötig.

3.12.4 MATLAB Funktionen zur Lösung von Anfangswertproblemen

- ode45 → Für nicht steife DGL. Einzelschritt-Verfahren basierend auf Runge-Kutta Verfahren der vierten und fünften Stufe. Gut als erster Versuch für viele Anfangswertprobleme.
- ode23 → Für nicht steife DGL. Einzelschritt-Verfahren basierend auf Runge-Kutta Verfahren der zweiten und dritten Stufe. Häufig schneller aber weniger genau als ode45
- ode113 → Für nicht steife DGL. Mehrschrittverfahren basierend auf Adams-Bashforth(-Moulton) Methoden.
- ode15s → Für steife DGL. Benutzt ein Mehrschritt-Verfahren. Tiefe bis mittlere Genauigkeit
- ode23s → Für steife DGL. Einzelschrittverfahren. Anwendbar auf einige Fälle die ode15s nicht lösen kann. Tiefe Genauigkeit
- ode23t → Für moderat steife DGL. Tiefe Genauigkeit
- ode23tb → Für steife DGL. Basiert auf einem impliziten Runge-Kutta Verfahren. Oft effizienter als ode15s.

4 Interpolation

4.1 Lernziele

- Sie können mittels der Lagrange - Interpolationsformel und dem Aitken-Neville Schema eine Anzahl Messpunkte durch ein Polynom interpolieren und dieses Schema in MATLAB implementieren
- Sie können für vorgegebene Stützpunkte die natürliche kubische Splinefunktion berechnen.

4.2 Problemstellung

Bei der Interpolation geht es darum bei einer Wertetabelle, die Lücken aufweist, die fehlenden Funktionswerte anzunähern. Es sei also eine Wertetabelle einer Funktion f mit $y_i = f(x_i)$ der Art

x	x_0	x_1	x_2	...	x_{j-1}	x_j	x_{j+1}	...	x_n
y	y_0	y_1	y_2	...	y_{j-1}	?	y_{j+1}	...	y_n

Abbildung 19: Allgemeine Wertetabelle Interpolation

gegeben. Die Wertepaare (x_i, y_i) heißen **Stützpunkte**, die x_i **Stützstelle** und die y_i **Stützwerte**. Gesucht ist nun eine möglichstgute Näherung des fehlenden Wertes y_j . Dementsprechend sind wir auf der Suche nach einer (stetigen) Funktion, welche die eigentliche Funktion $f(x)$ an der Stelle $f(x_j)$ möglichst gut annähert und exakt durch die bekannten Stützpunkte in der Umgebung x_j geht. Eine solche Funktion nennt man **Interpolierende**. Interpolatio kommt vor allen zur Anwendung, wenn die Funktion $f(x)$ nicht genügend bekannt oder nur schwer exakt zu berechnen ist. (Bspw. die digitale Bildbearbeitung)

Definition 17 (Interpolationsproblem). Gegeben sind $n+1$ Wertepaare (x_i, y_i) , $i = 0, \dots, n$ mit $x_i \neq x_j$ für $i \neq j$. Gesucht ist eine stetige Funktion g mit der Eigenschaft $g(x_i) = y_i$ für alle $i = 0, \dots, n$.

4.3 Polynominterpolation

Gegeben sind $n+1$ Stützpunkte

x	x_0	x_1	x_2	...	x_n
y	y_0	y_1	y_2	...	y_n

Abbildung 20: gegebene Stützpunkte

und gesucht ist ein Polynom $P_n(x)$, welches diese Punkte interpoliert. Wenn wir uns das Polynom in der üblichen Form nach Potenzen von x entwickelt denken, dann ist

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n \quad (85)$$

⇒ Jeder Stützpunkt gibt eine lineare Gleichung für die Bestimmung des Koeffizienten a_k . → wir erhalten gleich viele Gleichungen wie unbekannte Koeffizienten:

$$\begin{aligned} a_0 + a_1 x_0 + a_2 x_0^2 + \dots + a_n x_0^n &= y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 + \dots + a_n x_1^n &= y_1 \\ &\dots = \dots \\ a_0 + a_1 x_n + a_2 x_n^2 + \dots + a_n x_n^n &= y_n \end{aligned} \quad (86)$$

$$\left(\begin{array}{cccc} 1 & x_0 & \cdots & x_0^n \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^n \end{array} \right) \left(\begin{array}{c} a_0 \\ \vdots \\ a_n \end{array} \right) = \left(\begin{array}{c} y_0 \\ \vdots \\ y_n \end{array} \right)$$

Abbildung 21: Vandermonde-Matrix

Die Matrix wird Vandermonde-Matrix genannt. Im Prinzip könnten wir nun das lineare Gleichungssystem auflösen. Dies ist jedoch nicht effizient, da die Matrix typischerweise schlecht konditioniert ist. Durch eine andere Darstellung, lässt sich dies durchaus einfacher berechnen.

Satz (Lagrange Interpolationsformel). Durch $n+1$ Stützpunkte mit verschiedenen Stützstellen (d.h. $x_i \neq x_j$, für $i \neq j$) gibt es genau ein Polynom $P_n(x)$ vom Grade $\geq n$, welches alle Stützpunkte interpoliert, d.h. wo gilt

$$P_n(x_i) = y_i, i = 0, 1, \dots, n \quad (87)$$

$P_n(x)$ lautet in der Lagrangeform

$$P_n(x) = \sum_{i=0}^n l_i(x) y_i \quad (88)$$

dabei sind die $l_i(x)$ die Lagrange-Polynome vom Grad n definiert durch

$$l_i(x) = \prod_{j=0, j \neq i}^{n-1} \frac{x - x_j}{x_i - x_j} \quad (i = 0, 1, \dots, n) \quad (89)$$

Für viele Aufgabenstellungen ist es einfacher, nicht ein Interpolationspolynom für alle gegebenen Stützpunkte zu berechnen, sondern nur die Punkte in unmittelbarer Umgebung des gesuchten Wertes zu berücksichtigen. Im einfachsten Fall nimmt man jeweils die zwei benachbarten Stützpunkte und verbindet sie mit einer Geraden. Das

Interpolationspolynom ist dann also erster Ordnung und man spricht deshalb auch von (*stückweise*) *linearer Interpolation*. Dazu gibt es wiederum eine Fehlerabschätzung:

Satz (Fehlerabschätzung). *Sind die y_i Funktionswerte einer genügend oft stetig differenzierbaren Funktion f als $y_i = f(x_i)$, dann ist der Interpolationsfehler an einer Stelle x gegeben durch:*

$$|f(x) - P_n(x)| \leq \frac{|(x - x_0)(x - x_1)\dots(x - x_n)|}{(n+1)!} \max_{x_0 \leq \zeta \leq x_n} |f^{(n+1)}(\zeta)| \quad (90)$$

Bemerkung: Wir müssen das Maximum der $(n+1)$ -ten Ableitung der Funktion $f(x)$ auf dem Intervall $[x_0, x_n]$ kennen. Entspricht ist die Fehlerabschätzung nur dann anwendbar, wenn wir die Funktion $f(x)$ und ihre Ableitung kennen.

Um die Lagrange-Polynome zu berechnen kann sehr aufwendig sein, aus diesem Grund führen wir nun eine Rekursionsformel ein. Diese macht die Berechnung der Interpolationspolynome $P_n(x)$ effizienter → *Aitken-Neville Schema*. Es bezeichne dafür $p_{ij}(x)$ das Interpolationspolynom vom Grad $\leq j$ durch die Stützpunkte

x	x_{i-j}	x_{i-j+1}	...	x_i
y	y_{i-j}	y_{i-j+1}	...	y_i

Abbildung 22: Stützpunkte für das Aitken-Neville-Schema

Satz (Aitken-Neville Schema). *Die Polynome $p_{ij}(x)$ lassen sich folgendermassen rekursiv berechnen: für $i = 1, 2, 3, \dots$ und $j = 1, 2, \dots, i$ gilt*

$$p_{i0} = y_i \\ p_{ij} = \frac{(x_i - x)p_{i-1,j-1} + (x - x_{i-j})p_{i,j-1}}{x_i - x_{i-j}} \quad (91)$$

x	y							
x_0	$y_0 = p_{00}$							
x_1	$y_1 = p_{10}$	p_{11}						
x_2	$y_2 = p_{20}$	p_{21}	p_{22}					
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
x_i	$y_i = p_{i0}$	p_{i1}	p_{i2}	...	p_{ii}			
⋮	⋮	⋮	⋮	...	⋮	⋮	⋮	⋮
x_n	$y_n = p_{n0}$	p_{n1}	p_{n2}	...	p_{ni}	...	p_{nn}	

Abbildung 23: Schematische Darstellung für das Aitken-Neville-Schema

Dabei müssen die x_i in der ersten Spalte nicht unbedingt der Grösse nach sortiert sein. Ein Polynom p_{ij} entspricht also dem Interpolationspolynom, welches die Punkte x_0, x_1, \dots, x_i interpoliert, aus diesem Grund gilt: $P_i(x) = p_{ij}(x)$ bzw. $P_n(x) = p_{nn}(x)$

4.4 Splineinterpolation

Polynome mit einem hohen Grad oszillieren (= schwingen). Dies führt dazu, dass bei einer grossen Anzahl Stützpunkten das Interpolationspolynom meist keine gute Näherung mehr für die zu interpolierende Funktion $f(x)$ darstellt. Die Idee der Spline-Interpolation besteht nun darin, durch Polynome niederen Grades zu interpolieren und damit die Schwingungen zu unterdrücken, wobei man gleichzeitig sicherstellt, dass keine Knicke entstehen. Um diese zu erreichen, müssen die Polynome an den Anschlussstellen nicht nur denselber Funktionswert, sondern auch **dieselbe Ableitung** haben → Die Steigung der Tangente in diesen Punkten stimmt überein.

Wir wollen jetzt als Beispiel einen Algorithmus zur Berechnung der Koeffizienten für die natürliche kubische Splinefunktion kennen lernen:

Definition 18 (Algorithmus: natürliche kubische Splinefunktion). Gegeben seien $n + 1$ Stützpunkte (x_i, y_i) mit monoton aufsteigenden Stützstellen (Knoten) $x_0 < x_1 < \dots < x_n$ ($n \geq 2$).

Gesucht ist die natürliche kubische Splinefunktion $S(x)$, welche in jedem Intervall $[x_i, x_{i+1}]$ mit $i = 0, 1, \dots, n-1$ durch ein kubisches Polynom

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \quad (92)$$

dargestellt wird als $S(x) = S_i(x)$ mit $x \in [x_i, x_{i+1}]$

Die Koeffizienten a_i, b_i, c_i, d_i der Polynome $S_i(x)$ für $i = 0, 1, \dots, n-1$ berechnen sich wie folgt:

1. $a_i = y_1$
2. $h_i = x_{i+1} - x_i$
3. $c_0 = 0, c_n = 0$
4. Berechnung der Koeffizienten c_1, c_2, \dots, c_{n-1} aus dem Gleichungssystem

(a) $i = 1$:

$$2(h_0 + h_1)c_1 + h_1c_2 = 3\frac{y_2 - y_1}{h_1} - 3\frac{y_1 - y_0}{h_0}$$

(b) falls $n \geq 4$ gilt für $i = 2, \dots, n-2$:

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_ic_{i+1} = 3\frac{y_{i+1} - y_i}{h_i} - 3\frac{y_i - y_{i-1}}{h_{i-1}}$$

(c) $i = n - 1$:

$$h_{n-2}c_{n-2} + 2(h_{n-2} + h_{n-1})c_{n-1} = 3\frac{y_n - y_{n-1}}{h_{n-1}} - 3\frac{y_{n-1} - y_{n-2}}{h_{n-2}}$$

$$5. b_i = \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{3}(c_{i+1} + 2c_i)$$

$$6. d_i = \frac{1}{3h_i}(c_{i+1} - c_i)$$

Das Gleichungssystem unter Punkt 4 im Algorithmus hat die Form: $\mathbf{Ac} = \mathbf{z}$ mit:

$$\mathbf{A} = \begin{pmatrix} \frac{2(h_0 + h_1)}{h_1} & \frac{h_1}{h_2} & & & \\ & \frac{2(h_1 + h_2)}{h_2} & \frac{h_2}{h_3} & & \\ & & \ddots & \ddots & \\ & & & \frac{h_{n-3}}{h_{n-2}} & \frac{2(h_{n-3} + h_{n-2})}{h_{n-2}} \\ & & & & \frac{h_{n-2}}{2(h_{n-2} + h_{n-1})} \end{pmatrix}$$

Abbildung 24: A-Form für Algorithmus

und

$$\mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix}, \quad \mathbf{z} = \begin{pmatrix} \frac{3\frac{y_2 - y_1}{h_1} - 3\frac{y_1 - y_0}{h_0}}{h_1} \\ \frac{3\frac{y_3 - y_2}{h_2} - 3\frac{y_2 - y_1}{h_1}}{h_2} \\ \vdots \\ \frac{3\frac{y_n - y_{n-1}}{h_{n-1}} - 3\frac{y_{n-1} - y_{n-2}}{h_{n-2}}}{h_{n-1}} \end{pmatrix}$$

Abbildung 25: c-Form für Algorithmus

Bemerkung: Die Matrix A ist immer invertierbar. Zur numerischen Lösung sollte man den Gausschen-Algorithmus bzw. den Cholesky-Algorithmus verwenden. Das System ist gut konditioniert, Pivotsuche ist nicht erforderlich. Natürlich bietet MATLAB bereits fertige Funktionen bzw. eine ganze Toolbox für Splineinterpolation

5 Ausgleichsrechnung

Bei der Auswertung von Daten, will man häufig Datenpunkte mit einer gewissen Streuung durch eine relativ einfache Funktion annähern. Dabei sucht man eine Funktion, die möglichst nahe bei den Datenpunkten durchläuft, ohne dies exakt zu interpolieren.

5.1 Lernziele

- Sie können die Begriffe Ausgleichsproblem, Ansatzfunktion, Ausgleichsfunktion und Fehlerfunktional definieren
- Sie kennen die Optimierung des Fehlerfunktional im Sinne der kleinsten Fehlerquadrate (least squares)
- Sie können das lineare sowie das allgemeine Ausgleichsproblem definieren und für spezifische Beispiele lösen
- Sie können das gedämpfte Gauss-Newton Verfahren anwenden und in Matlab implementieren

5.2 Problemstellung

Im Unterschied zur Interpolation versuchen wir bei der Ausgleichsrechnung nicht, eine Funktion f zu finden, die exakt durch sämtliche Wertepaare geht, sondern diese nur möglichst gut approximiert. Dies ist insbesondere dann sinnvoll, wenn es eine grosse Anzahl von Datenpunkten gibt (die meist zusätzlich noch fehlerbehaftet sind) und die durch eine Funktion mit nur wenigen Paramtern beschrieben werden sollen.

Definition 19 (Ausgleichsproblem). Gegeben sind n Wertpaare (x_i, y_i) , $i = 1, \dots, n$ mit $x_i \neq x_j$ für $i \neq j$. Gesucht ist eine stetige Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$, die die Wertpaare in einem gewissen Sinn bestmöglich annähert, d.h. dass möglichst genau gilt

$$f(x_i) \approx y_i \tag{93}$$

für alle $i = 1, \dots, n$

Definition 20 (Ansatzfunktionen / Ausgleichsfunktion / Fehlerfunktional / kleinste Fehlerquadrate). Gegeben sei eine Menge F von stetigen **Ansatzfunktionen** f auf dem Intervall $[a, b]$ sowie n Wertepaare $(x_i, y_i), i = 1, \dots, n$

Ein Element $f \in F$ heisst **Ausgleichsfunktion** von F zu den gegebenen Wertpaaren, falls das **Fehlerfunktional**

$$E(f) := \|\mathbf{y} - f(\mathbf{x})\|_2^2 = \sum_{i=1}^n (y_i - f(x_i))^2 \quad (94)$$

für f minimal wird, d.h. $E(f) = \min\{E(g)|f \in F\}$

Man nennt das so gefundene f dann optimal im Sinne der **kleinsten Fehlerquadrate** (least squares fit)

Bemerkungen:

- Diese Forderung der kleinsten Fehlerquadrate bedeutet nichts anderes, als dass das Quadrat der 2-Norm des Fehlervektors

$$\begin{pmatrix} y_1 - f(x_1) \\ \vdots \\ y_n - f(x_n) \end{pmatrix} \quad (95)$$

minimal sein soll. Andere Normen sind auch möglich, was zu anderen Ergebnissen führen kann. Die in der Praxis am häufigsten verwendeten Norm ist allerdings die 2-Norm

- Es ist möglich, die Fehler noch mit Gewichten $w_i > 0$ zu versehen, d.h. man minimiert

$$\sum_{i=1}^n w_i * (y_i - f(x_i))^2 \quad (96)$$

Wenn z.B. gewisse Datenpunkten (x_i, y_i) grosse Messgenauigkeiten aufweisen, können diese mit einem kleineren Gewicht versehen werden, als Datenpunkte mit höheren Messgenauigkeit. Die Summe der Gewichte sollte dabei normiert sein, also $\sum_{i=1}^n w_i = 1$

- Wählt man F als Menge aller Geraden, dann nennt man das so gefundene f *Ausgleichsgerade* oder in statischen Zusammenhängen auch *Regressionsgerade*

5.3 Lineare Ausgleichsprobleme

Ausgehend von der Wertetabelle

x_i	x_1	x_2	\dots	x_n
y_i	y_1	y_2	\dots	y_n

Abbildung 26: Ausgehende Wertetabelle von linearen Ausgleichsprobleme

suchen wir die Ausgleichsgerade der Form $f(x) = ax + b$, also $F := \{a_1 f_1 + a_2 f_2 | a_1, a_2 \in \mathbb{R}\}$ mit den Ansatzfunktionen $f_1(x) = x$ und $f_2(x) = 1$

Das Fehlerfunktional hat dann die Form

$$E(f)(a, b) := E(f) = \sum_{i=1}^n (y_i - f(x_i))^2 = \sum_{i=1}^n (y_i - (ax_i + b))^2 \quad (97)$$

Dieses soll minimal werden, d.h. die partiellen Ableitungen nach den Parametern a und b müssen verschwinden (in Analogie zum eindimensionalen Fall):

$$\begin{aligned}
0 &\stackrel{!}{=} \frac{\partial E}{\partial a} = \sum_{i=1}^n \frac{\partial}{\partial a} (y_i - (ax_i + b))^2 = \sum_{i=1}^n 2 \cdot (y_i - (ax_i + b)) \cdot (-x_i) = -2 \sum_{i=1}^n (y_i - (ax_i + b)) \cdot (x_i) \\
0 &\stackrel{!}{=} \frac{\partial E}{\partial b} = \sum_{i=1}^n \frac{\partial}{\partial b} (y_i - (ax_i + b))^2 = \sum_{i=1}^n 2 \cdot (y_i - (ax_i + b)) \cdot (-1) = -2 \sum_{i=1}^n (y_i - (ax_i + b))
\end{aligned}$$

Abbildung 27: partielle Ableitung von a und b

Wir haben also zwei Gleichungen für die Unbekannten a, b . Durch die Umformung erhalten wir

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

Abbildung 28: Umformung der partiellen Ableitung von a und b

Daraus folgt:

$$\begin{aligned}
a \sum_{i=1}^n x_i^2 + b \sum_{i=1}^n x_i &= \sum_{i=1}^n x_i y_i \\
a \sum_{i=1}^n x_i + b \underbrace{\sum_{i=1}^n 1}_n &= \sum_{i=1}^n y_i
\end{aligned} \tag{98}$$

oder in der Matrixschreibweise erhalten wir das lineare Gleichungssystem für die Unbekannten (a, b) und können dieses nach a, b auflösen

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

Abbildung 29: Matrixschreibweise von linearen Ausgleichsproblemen

Definition 21 (lineares Ausgleichsproblem). Gegeben seien n Wertepaare $(x_i, y_i), i = 1, \dots, n$ und m Basisfunktionen f_1, \dots, f_m auf einem Intervall a, b

. Wir wählen F als die Menge der Ansatzfunktionen $f := \lambda_1 f_1 + \dots + \lambda_m f_m$ mit $\lambda_j \in \mathbb{R}$ also $F = \{f = \lambda_1 f_1 + \dots + \lambda_m f_m \mid \lambda_j \in \mathbb{R}, j = 1, \dots, m\}$

Es liegt dann ein **lineares Ausgleichsproblem** vor mit dem Fehlerfunktional

$$\begin{aligned} E(f) &= \|y - f(x)\|_2^2 \\ &= \sum_{i=1}^n (y_i - f(x_i))^2 \\ &= \sum_{i=1}^n (y_i - \sum_{j=1}^m \lambda_j f_j(x_i))^2 \\ &= \|y - A\lambda\|_2^2 \end{aligned} \tag{99}$$

vor, wobei

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

Abbildung 30: Matrixschreibweise

→ Das System $A\lambda = y$ heisst **Fehlergleichungssystem**

Bemerkungen:

- Das Fehlergleichungssystem besitzt n Gleichungen mit m Unbekannte. In der Regel ist $n > m$, dement sprechend haben wir mehr Gleichungen als Unbekannt → das LGS ist überbestimmt. Man kann daher nicht damit rechnen, dass es eine Lösung gibt
- Für $m = n$ haben wir eine eindeutige Lösung und $E(f) = 0$. Dies ist gleichbedeutend damit, dass die Ausgleichsfunktion f exakt durch sämtliche Wertepaare geht. Dies ist ein Spezialfall der Interpolation

Definition 22 (Normalgleichungen / Normalgleichungssystem). Die Gleichungen

$$\frac{\varphi E(f)(\lambda_1, \dots, \lambda_m)}{\varphi \lambda_j} = 0 \quad j = 1, \dots, m \tag{100}$$

heissen **Normalgleichungen** des linearen Ausgleichsproblems

Das System sämtlicher Normalgleichungen heisst **Normalgleichungssystem** und lässt sich als lineares Gleichungssystem schreiben

$$A^T A \lambda = A^T y \tag{101}$$

Bemerkungen:

Die Lösungen des Normalgleichungssystems sind die gesuchten Parameter des linearen Ausgleichproblems. Die $m \times m$ Matrix $A^T A$ ist symmetrisch und positiv definit, das Normalgleichungssystem kann deshalb mit der Cholesky-Zerlegung gelöst werden. Numerisch stabiler ist die sogenannte QR-Zerlegung.

5.4 Nichtlineare Ausgleichsprobleme

Wir haben gesehen, dass es die Möglichkeit gibt ein nichtlineares Ausgleichsproblem in ein lineares Ausgleichsproblem umzuwandeln. Jedoch soll man in der Lage sein ein nichtlineares Ausgleichsproblem entsprechend lösen zu können.

Definition 23 (Allgemeines Ausgleichsproblem). Gegeben seien n Wertepaare (x_i, y_i) , $i = 1, \dots, n$, und die Menge F der Ansatzfunktionen $f_p = f_p(\lambda_1, \lambda_2, \dots, \lambda_m, x)$ mit m Parametern $\lambda_j \in \mathbb{R}$, $j = 1, \dots, m$ also $F = \{f_p(\lambda_1, \lambda_2, \dots, \lambda_m, x) | \lambda_j \in \mathbb{R}, j = 1, \dots, m\}$

Das **allgemeine Ausgleichproblem** besteht darin, die m Parameter $\lambda_1, \dots, \lambda_m$ zu bestimmen, so dass das Fehlerfunktional E

$$\begin{aligned} E(f) &= \sum_{i=1}^n (y_i - f_p(\lambda_1, \lambda_2, \dots, \lambda_m, x_i))^2 \\ &= \left\| \begin{pmatrix} y_1 - f_p(\lambda_1, \lambda_2, \dots, \lambda_m, x_1) \\ y_2 - f_p(\lambda_1, \lambda_2, \dots, \lambda_m, x_2) \\ \vdots \\ y_n - f_p(\lambda_1, \lambda_2, \dots, \lambda_m, x_n) \end{pmatrix} \right\|_2^2 \\ &= \|y - f(\lambda)\|_2^2 \end{aligned} \quad (102)$$

minimal wird unter allen zulässigen Parameterbelastungen, wobei

$$f(\lambda) := f(\lambda_1, \lambda_2, \dots, \lambda_m) := \begin{pmatrix} f_1(\lambda_1, \lambda_2, \dots, \lambda_m) \\ f_2(\lambda_1, \lambda_2, \dots, \lambda_m) \\ \vdots \\ f_n(\lambda_1, \lambda_2, \dots, \lambda_m) \end{pmatrix} := \begin{pmatrix} f_p(\lambda_1, \lambda_2, \dots, \lambda_m, x_1) \\ f_p(\lambda_1, \lambda_2, \dots, \lambda_m, x_2) \\ \vdots \\ f_p(\lambda_1, \lambda_2, \dots, \lambda_m, x_n) \end{pmatrix}, \quad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad \lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{pmatrix}$$

Abbildung 31: Bedingungen bei allgemeinen Ausgleichsproblem

Bemerkungen:

- Falls die Ansatzfunktion f_p linear in den Parametern sind, haben wir den Spezialfall des linearen Ausgleichsproblems mit $f(\lambda) = A\lambda$
- Das allg. Ausgleichproblem ist also äquivalent zur Bestimmung des Minimums einer Funktion

$$E : \mathbb{R}^m \rightarrow \mathbb{R} \quad (103)$$

und wir könnten wieder die Normalgleichungen aufstellen, indem wir die partiellen Ableitungen von f nach den Parametern λ_i gleich Null setzen und das etnsthende, diesmal nicht lineare Gleichungssystem lösen.

5.5 Das Gauss-Newton-Verfahren

Definition 24 (Quadratmittelpproblem). Gegeben ist eine Funktion $g : \mathbb{R}^m \rightarrow \mathbb{R}^n$ und das zugehörige Fehlerfunktional $E : \mathbb{R}^m \rightarrow \mathbb{R}$, definiert durch

$$E(\mathbf{x}) := \|g(\mathbf{x})\|_2^2 \quad (104)$$

Das Problem, einen Vektor $\mathbf{x} \in \mathbb{R}^m$ zu finden, für den $E(\mathbf{x})$ minimal wird, nennt man **Quadratmittelpproblem**

evtl. Seite 153 ergänzen?!

Definition 25 (Gauss-Newton-Verfahren). Sei $\lambda^{(0)}$ ein Startvektor in der Nähe des Minimums von E. Das Gauss-Newton-Verfahren zur näherungsweisen Bestimmung des Minimums lautet für $k = 0, 1, \dots$:

1. Berechne $\delta^{(k)}$ als Lösungen des linearen Ausgleichsproblems

$$\min \|g(\lambda^{(k)}) + Dg(\lambda^{(k)}) * \delta^{(k)}\|_2^2 \quad (105)$$

d.h. löse konkret:

$$Dg(\lambda^{(k)})^T Dg(\lambda^{(k)}) \delta^{(k)} = -Dg(\lambda^{(k)})^T g(\lambda^{(k)}) \quad (106)$$

nach $\delta^{(k)}$ auf

2. Setze

$$\lambda^{(k+1)} = \lambda^{(k)} + \delta^{(k)} \quad (107)$$

Definition 26 (gedämpftes Gauss-Newton-Verfahren). Sei $\lambda^{(0)}$ ein Startvektor in der Nähe des Minimums von E. Das gedämpfte Gauss-Newton-Verfahren zur näherungsweisen Bestimmung des Minimums für $k = 0, 1, \dots$ lautet:

1. Berechne $\sigma^{(k)}$ als Lösung des linearen Ausgleichsproblems

$$\min \|g(\lambda^{(k)}) + Dg(\lambda^{(k)}) * \delta^{(k)}\|_2^2 \quad (108)$$

d.h. als Lösung des linearen Gleichungssystems

$$Dg(\lambda^{(k)})^T Dg(\lambda^{(k)}) \delta^{(k)} = -Dg(\lambda^{(k)})^T g(\lambda^{(k)}) \quad (109)$$

nach $\lambda^{(k)}$ auf

2. Finde das minimale $p \in \{0, 1, \dots, p_{max}\}$ mit

$$\underbrace{\|g(\lambda^{(k)} + \frac{\delta^{(k)}}{2^p})\|_2^2}_{\lambda^{k+1}} < \|g(\lambda^{(k)})\|_2^2 \quad (110)$$

3. Falls kein minimales p gefunden werden kann, rechne mit $p = 0$ weiter

4. Setze

$$\lambda^{(k+1)} = \lambda^{(k)} + \frac{\delta^{(k)}}{2^P} \quad (111)$$

Bemerkungen:

- Ein Vorteil ist, dass es in der Regel für einen grösseren Bereich von Startvektoren konvergiert als das ungedämpfte Gaus-Newton-Verfahren. Durch das fortlaufende Anpassen der "Korrekturrichtung" kann es sozusagen noch mit Startvektoren umgehen, die weiter entfernt sind vom Optimum. Die Dämpfung ist aber keine Garantie für Konvergenz. Man benötigt noch weitere Bedingungen.
- Als Abbruchkriterium des Algorithmus kann z.B.

$$\|\frac{\delta^{(k)}}{2^P}\|_2 TOL \quad (112)$$

verwendet werden. Wiederum ist das aber keine Garantie, dass die berechnete Näherung einen maximalen Abstand von TOL zum gesuchten Minimum besitzt

6 Fourier-Reihen und Fourier-Transformation

Die Theorie der Fourier-Reihen und Fourier-Transformation ermöglicht die Zerlegung eines periodischen Signals in eine unendliche Summe (d.h. Reihe) von harmonischen Schwingungen der Form $A\cos(\omega t + \phi)$ oder $B\sin(\omega t + \phi)$

6.1 Lernziele

- Sie kennen wichtige Anwendungsfälle der Fourier-Theorie
- Sie kennen die Definition einer Fourier-Reihe und können deren Koeffizienten berechnen
- Sie können die diskrete Fourier-Transformation auf konkrete Problemstellung anwenden und in MATLAB implementieren

6.2 Anwendungen

Es macht möglich, ein beliebiges Signal (z.B. in Form einer Schall- oder elektromagnetischen Welle) in seine Frequenzen zu zerlegen bzw. aus einem Frequenzspektrum des ursprünglichen Signals zu rekonstruieren. Was die Fourier-Analyse so einzigartig macht, ist die Möglichkeit, sie unmittelbar physikalisch zu erfahren.

Mögliche Anwendungsgebiete:

- Detektion von Frequenzen
- Entrauschen von verrauschten Signalen
- MP3-Verfahren zur Komprimierung von Audiodateien
- Bildverarbeitung bspw. JPEG

6.3 Fourier-Reihen

Evtl. Seiten 162-164 ergänzen

6.3.1 Allgemeine Fourier-Reihen

Satz (Fourier-Reihen / Fourier-Koeffizienten). Sei $f : \mathbb{R} \rightarrow \mathbb{R}$ eine periodische Funktion mit der Kreisfrequenz ω_0 und Periode $T = \frac{2\pi}{\omega_0}$. Des Weiteren lasse sich das Periodenintervall in endlich viele Teilintervalle zerlegen, in denen $f(x)$ stetig und monoton ist, und in den Unstetigkeitsstellen existiere sowohl der links- als auch der rechtsseitige Grenzwert (Dirichletsche Bedingungen).

Dann kann $f(x)$ in eine **Fourier-Reihe** der Form

$$f(x) = \frac{A_0}{2} + \sum_{k=1}^{\infty} [A_k \cos(k * \omega_0 * x) + B_k \sin(k * \omega_0 * x)] \quad (113)$$

entwickelt werden. Dabei bedeuten:

- $\omega_0 = \frac{2\pi}{T}$: Kreisfrequenz der Grundschwingung
- $k * \omega_0$: Kreisfrequenz der sogenannten k -ten harmonischen Oberschwingung

Die **Fourierkoeffizienten** von f werden dabei aus den Integralformeln

$$\begin{aligned} A_0 &= \frac{2}{T} \int_{(T)} f(x) dx \\ A_k &= \frac{2}{T} \int_{(T)} f(x) \cos(k\omega_0 x) dx \\ B_k &= \frac{2}{T} \int_{(T)} f(x) \sin(k\omega_0 x) dx \end{aligned} \quad (114)$$

berechnet. Das Symbol (T) unter dem Integral bedeutet, dass die Integration über ein beliebiges Intervall der Länge T zu erstrecken ist.

Die **Amplitude** des k -ten Fourierkoeffizienten berechnet sich zu

$$\begin{aligned} amp_{k=0} &:= \frac{A_0}{2} \\ amp_k &:= \sqrt{A_k^2 + B_k^2} \end{aligned} \quad (115)$$

6.4 diskrete Fourier-Transformation

Bisher war die Funktion $f(x)$ in analytischer Form gegeben. Tatsächlich findet man in vielen Anwendungen mit einer endlichen Anzahl von diskreten Datenpunkten (bspw. Messungen, Bilder, Tonaufnahme etc.). Hier befassen wir uns mit FOurier-Reihen für Funktionen mit **äquidistanten** Punkten (sprich mit gleichem Abstand zwischen den Datenpunkten)

Satz (Diskrete Fourier-Reihen / Diskrete Fourier-Koeffizienten (DFT)). Sei $f : [0, T] \rightarrow \mathbb{R}$ definiert durch $2n+1$ äquidistante Punkte $(t_i, f(t_i))$ mit $i = 1, \dots, 2n+1$ und $f(0) = f(T)$. Der Abstand zwischen den Punkten sei konstant $\Delta t = \frac{T}{2n}$ d.h. $t_i = (i - 1) * \Delta t$. Dann kann $f(t_i)$ in eine **diskrete Fourier-Reihe** der Form

$$f(t_i) = \sum_{k=0}^n [A_k \cos(k * \omega_0 * t_i) + B_k \sin(k * \omega_0 * t_i)] \quad (116)$$

entwickelt werden, auch bekannt als **inverse reelle diskrete Fourier-Transformation**. Der letzte Punkt mit $t_{2n+1} = T$ wird nicht berücksichtigt, da $f(0) = f(T)$. Dabei bedeuten:

- $\omega_0 = \frac{2\pi}{T}$: Kreisfrequenz der Grundschwingung
- $k * \omega_0$: Kreisfrequenz der sogenannten k -ten harmonischen Oberschwingung

Die **diskreten Fourierkoeffizienten** von f werden dabei aus

$$\begin{aligned} A_0 &= \frac{1}{2n} \sum_{i=1}^{2n} f(t_i) \\ A_k &= \frac{1}{n} \sum_{i=1}^{2n} f(t_i) \cos(k * \omega_0 * t_i) \quad k = 1, 2, \dots, n-1 \\ A_n &= \frac{1}{2n} \sum_{i=1}^{2n} f(t_i) \cos(n * \omega_0 * t_i) \end{aligned} \quad (117)$$

und

$$\begin{aligned} B_0 &= 0 \\ B_k &= \frac{1}{n} \sum_{i=1}^{2n} f(t_i) \sin(k * \omega_0 * t_i) \quad k = 1, 2, \dots, n-1 \\ B_n &= 0 \end{aligned} \quad (118)$$

berechnet. Die Koeffizienten A_k und B_k werden auch **reelle diskrete Fourier-Transformation** genannt.
Bemerkungen:

- Die Koeffizienten A_k und B_k beschreiben die Funktion $f(t)$ im Frequenzbereich. Sie erlauben, die Funktion quasi aus der Zeit- (oder Orts-) Domäne in die Frequenzdomäne zu transformieren, deshalb der Begriff "Diskrete Fourier-Transformation"
- Die Rücktransformation aus der Frequenzdomäne in die Zeit- (oder Orts-) Domäne erfolgt über $f(t_i) = \sum_{k=0}^n [A_k \cos(k * \omega_0 * t_i) + B_k \sin(k * \omega_0 * t_i)]$ deshalb der Begriff "inverse diskrete Fourier-Transformation"

$$\{f(t_1), f(t_2), \dots, f(t_{2n})\} \xrightarrow{DFT} \{A_0, A_1, \dots, A_n, B_0, B_1, \dots, B_n\} \xrightarrow{\text{inverse DFT}} \{f(t_1), f(t_2), \dots, f(t_{2n})\} \quad (119)$$

- Falls $f(0) \neq f(T)$, ersetzt man üblicherweise $f(0)$ und $f(T)$ mit $\frac{f(0)+f(T)}{2}$
- die reelle DFT ist von der Ordnung $O(n^2)$. Für grosse n gibt es den schnelleren Algorithmus der "Fast Fourier Transform" welcher die Ordnung $O(n \log n)$ hat. In MATLAB gibt es dazu die Funktion "fft.m"

Eine einfache Art, das Frequenzspektrum grafisch darzustellen, ist das sogenannte Leistungsdichte-Spektrum. Handelt es sich beim Signal $f(t)$ um eine elektromagnetische Welle, entspricht das Quadrat der Fourierkoeffizien-

ten der Energie dieser Welle bei der k-ten Frequenz. Je mehr Energie (d.h. je grösser das Quadrat der Fourier-Koeffizienten) bei einer spezifischen Frequenz, umso dominanter ist die entsprechende harmonische Schwingung.

Definition 27 (Leistungsdichte-Spektrum / Power-Spektrum). Wir definieren das Leistungsdichte-Spektrum / Power-Spektrum für die diskrete Fourier-Transformation als

$$P_k = \frac{1}{4} \frac{(A_k^2 + B_k^2)}{n} \quad k = 0, \dots, n \quad (120)$$

für die k-te Frequenz $v_k = \frac{k*\omega_0}{2\pi} = \frac{k}{T}$