School of Engineering
Winterthur
Zürcher Hochschule für Angewandte Wissenschaften

# Numerik für Ingenieure

Skript
(R. Axthelm)


WORK IS ALWAYS IN PROGRESS!!!

# Inhaltsverzeichnis

–1–
# Funktionen ertasten mit dem Taylorpolynom

Problemstellung:

Wir befinden uns in der Situation, mit einer zu "komplizierten" Funktion $f$ rechnen zu müssen. Die Idee ist nun eine alternative Funktion zu finden, die $f$ gut approximiert und von einfacher Gestalt ist.

Vielleicht verhält es sich unglücklicherweise gerade so, dass wir das Integral

$$\int_I \cos\left(\frac{\pi}{2}\, x^2\right)\, dx$$

benötigen, wobei der Integrant keine Stammfunktion in geschlossener Form besitzt. Wir ersetzen die Funktion $\cos\left(\frac{\pi}{2}\, x^2\right)$ durch ein sogenantes Taylorpolynom der Gestalt

$$\sum_{k=0}^{n} a_n x^n\,,$$

machen damit zwar einen Fehler, können aber dieses Polynom problemlos integrieren und auch differenzieren. Dieses Mittel ist ein sehr wertvolles, da man sich das Leben kurzerhand leicht machen kann. Es findet daher sehr oft Anwendung in der Praxis.

Eine komplizierte Funktion durch ein einfaches Polynom ersetzen ist natürlich etwas, was man nicht geschenkt kriegt. Zunächst einmal werden wir feststellen, dass das nicht immer funktioniert und wenn doch werden wir einen Preis dafür bezahlen: Wir machen einen Fehler.

Konzentrieren wir uns zunächst darauf wie der Trick funktioniert, dann betrachten wir Beispiele des Misserfolgs und anschließend schauen wir uns an, was wir für den Spaß bezahlen müssen.

### Die Idee

Es sei $f : I \subset \mathbb{R} \to \mathbb{R}$ eine gegebene Funktion. Wir suchen ein Polynom $T_{f,n}(x, x_0) \in \mathbb{P}_n$, welches in der Nähe von $x_0 \in I$ die Funktion $f$ näherungsweise gut beschreibt.

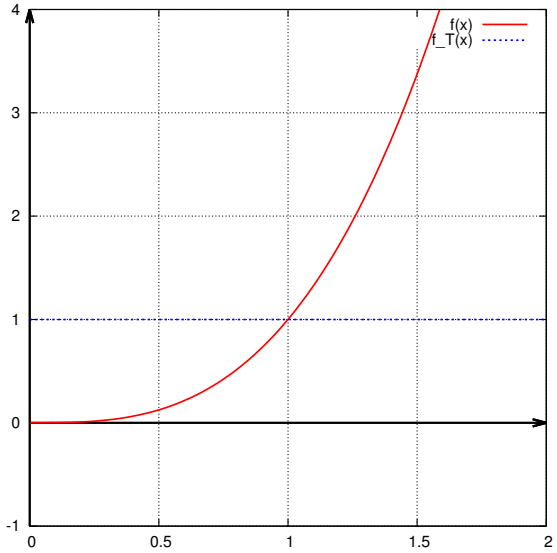



Abbildung 1: $T_{f,0}(x,1)$ für $f(x) = x^3$

Am Neispiel $f(x) = x^3$ etwa wäre bei $x_0 = 1$

$$T_{T,0}(x,1) = 1$$

eine mögliche Approximation. Es ist

$$T_{f,0}(x,1) = 1 = f(1)\,.$$

Nahe bei $x_0 = 1$ ist der Fehler klein (s. Abb. 1).

Die Approximation ist augenscheinlich nicht gut genug. Schön wäre noch, wenn nicht nur der Wert bei $x_0$ übereinstimmt sondern auch die erste Ableitung. Wir bilden den Differenzenquotient für dir erste Ableitung bei $x = x_0$:

$$f'(x_0) \approx \frac{f(x) - f(x_0)}{x - x_0}$$

Dann erhalten wir eine Näherung für $f$ durch

$$f(x) \approx f(x_0) + f'(x_0)\,(x - x_0)\,.$$

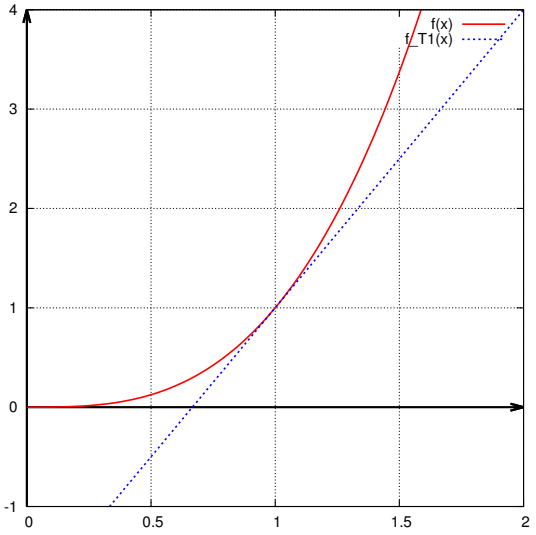



Abbildung 2: $T_{f,1}(x,1)$ für $f(x) = x^3$

Wir definieren für das Polynom vom Grad 1:

$$T_{f,1}(x,x_0) = f(x_0) + f'(x_0)\,(x - x_0) \in \mathbb{P}_1$$

Dafür gilt, dass Funktionswert und erste Ableitung bei $x_0$ von $f$ und $T_{f,1}$ übereinstimmen:

$$T_{f,1}(x_0,x_0) = f(x_0) \qquad \text{und} \qquad T'_{f,1}(x_0,x_0) = f'(x_0)$$

Beispiel 1 An unserem Beispiel, dargestellt in Abbildung 2 mit $x_0 = 1$ sieht das so aus:

$$
\begin{aligned}
f(1) &= 1 \\
f'(1) &= 3 \\
T_{f,1}(x,1) &= 1 + 3\,(x-1) = 3\,x - 2
\end{aligned}
$$

Es gilt für $T_{f,1}(x,1)$

$$
\begin{aligned}
T_{f,1}(1,1) &= 3 - 2 = 1 = f(1) \\
T'_{f,1}(1,1) &= 3 = f'(1)
\end{aligned}
$$

Das ist schon gar nicht schlecht aber wir kriegen den Hals nicht voll und wollen noch mehr. Es sollte sich die Approximation noch besser an die Kurve bei $(x_0, f(x_0))$ anschmiegen. Wir fordern daher noch, dass

$$T''_{f,2}(x_0,x_0) = f''(x_0)$$

gilt. Wir setzen für ein beliebiges $a_2 \in \mathbb{R}$ die Funktion

$$T_{f,2}(x,x_0) = f(x_0) + f'(x_0)\,(x-x_0) + a_2\,(x-x_0)^2$$

an und bstimmen die Konstante $a_2$ so, dass unser Wunsch erfüllt wird. Damit erhalten wir

$$T''_{f,2}(x_0,x_0) = f''(x_0) \qquad \Leftrightarrow \qquad a_2 = \frac{1}{2}\,f''(x_0)\,.$$

Mit

$$T_{f,2}(x,x_0) = f(x_0) + f'(x_0)(x-x_0) + \frac{f''(x_0)}{2}(x-x_0)^2$$

gilt nun:

$$
\begin{aligned}
T_{f,2}(x_0,x_0) &= f(x_0) \\
T'_{f,2}(x_0,x_0) &= f'(x_0) \\
T''_{f,2}(x_0,x_0) &= f''(x_0)
\end{aligned}
$$

Beispiel 2 Für unser Beispiel, dargestellt in Abb. 3 sieht das so aus:

$$
\begin{aligned}
T_{f,2}(x,1) &= 1 + 3\,(x-1) + \frac{1}{2}\,6\,(x-1)^2 \\
&= 3\,x^2 - 3\,x + 1
\end{aligned}
$$

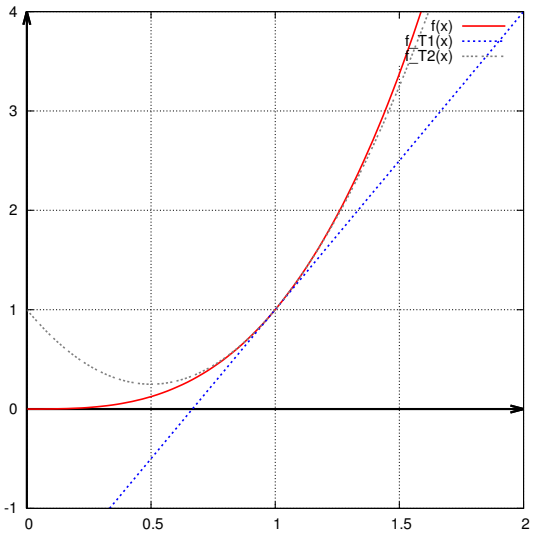



Abbildung 3: $T_{f,2}(x,1)$ für $f(x) = x^3$

Wir wollen nun diese Vorgehensweise auf beliebige Ableitungsordnungen erweitern und setzen dazu die Funktion

$$T_{f,n}(x,x_0) = f(x_0)\,\underbrace{(x-x_0)^0}_{\in\mathbb{P}_0} + f'(x_0)\,\underbrace{(x-x_0)^1}_{\in\mathbb{P}_1} + \cdots + a_n\,\underbrace{(x-x_0)^n}_{\in\mathbb{P}_n}\,.$$

Die $n$-te Ableitung dieser Funktion ist gerade die $n$-te Ableitung des letzten Terms in $\mathbb{P}_n$, da alle anderen Terme verschwinden. Es gilt also

$$\begin{aligned}
T_{f,n}^{(n)}(x,x_0) &= \left(a_n\,(x-x_0)^n\right)^{(n)} \\
&= \left(a_n\cdot n\cdot(x-x_0)^{n-1}\right)^{(n-1)} \\
&= \left(a_n\cdot n\cdot(n-1)\cdot(x-x_0)^{n-2}\right)^{(n-2)} \\
&\;\;\vdots \\
&= a_n\cdot n\cdot(n-1)\cdots 1\cdot(x-x_0)^0 \\
&= n!\,a_n\,.
\end{aligned}$$

Damit können wir schlussendlich die Konstante $a_n$ bestimmen:

$$T_{f,n}^{(n)}(x,x_0) = f^{(n)}(x_0) \qquad \Leftrightarrow \qquad a_n = \frac{1}{n!}\,f^{(n)}(x_0)$$

Diesen Prozess können wir an allen Funktionen $f$ für beliebig viele Ableitungen $n$ durchführen, sofern die Funktion an der Stelle $x_0$ auch $n$-mal differenzierbar ist.

Wir halten dieses wertvolle Ergebnis fest:

**Definition 1.1** (Taylorpolynom ). *Für eine in $x_0 \in I$ $n$–mal differenzierbare Funktion $f: I \to \mathbb{R}$ heißt das Polynom*

$$T_{f,n}(x,x_0) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x-x_0)^k$$

$$= f(x_0) + f'(x_0)(x-x_0) + \frac{f^{(2)}(x_0)}{2!}(x-x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n$$

*das Taylorpolynom der Ordnung $n$ von $f$ an der Stelle $x_0$.*

Ist die Funktion $f$ gar $\infty$-oft differenzierbar, so kann man auch die entsprechende Reihe bilden:

**Definition 1.2** (Taylorreihe ). *Sei $f : I \to \mathbb{R}$ eine unendlich oft differenzierbare Funktion. Dann heißt die Potenzreihe*

$$T_f(x, x_0) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k$$

*die Taylorreihe der Funktion $f$ im Entwicklungspunkt $x_0$. Wir sagen die Funktion ist taylorentwickelbar, wenn $f(x) = T_f(x, x_0)$ gilt.*

Beispiel 3

1.

$$f(x) = \sin x$$

entwickelt um $x_0 = 0$: Wie lautet die $n$–te Ableitung der Sinusfunktion bei $x = 0$?

$$\begin{aligned}
\sin^{(1)}(0) &= \cos(0) = 1 & \sin^{(2)}(0) &= -\sin(0) = 0 \\
\sin^{(3)}(0) &= -\cos(0) = -1 & \sin^{(4)}(0) &= \sin(0) = 0 \\
\sin^{(5)}(0) &= \cos(0) = 1 & \sin^{(6)}(0) &= -\sin(0) = 0 \\
\sin^{(7)}(0) &= -\cos(0) = -1 & \sin^{(8)}(0) &= \sin(0) = 0 \\
&\vdots \\
\sin^{(4n-3)}(0) &= 1 & \sin^{(4n)}(0) &= 0 \\
\sin^{(4n-1)}(0) &= -1 & \sin^{(4n-2)}(0) &= 0 \\
&\vdots \\
\sin^{(2n+1)}(0) &= (-1)^n & \sin^{(2n)}(0) &= 0
\end{aligned}$$

Daraus ergibt sich die Taylorreihe (siehe Abbildung 4)

$$T_{\sin}(x, 0) = \sum_{n=0}^{\infty} \frac{\sin^{(n)}(0)}{n!} x^n = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$

2. Wir entwickeln die Funtkion

$$f(x) = \frac{1}{1-x}$$

mit dem Definitionsbereich $\mathbb{D}_f = \mathbb{R} \setminus \{1\}$ um $x_0 \neq 1$:

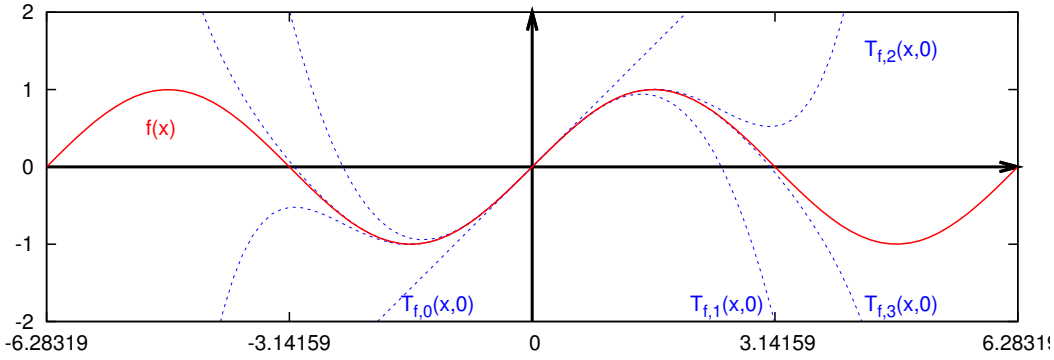



Abbildung 4: Taylorpolynome für $n = 0, 1, 2, 3$, $x_0 = 0$ und $f(x) = \sin x$

$$
\begin{aligned}
f^{(0)}(x_0) &= \frac{1}{1-x_0} & f^{(3)}(x_0) &= \frac{1\cdot 2\cdot 3}{(1-x_0)^4} \\
f^{(1)}(x_0) &= \frac{1}{(1-x_0)^2} & &\vdots \\
f^{(2)}(x_0) &= \frac{1\cdot 2}{(1-x_0)^3} & f^{(k)}(x_0) &= \frac{k!}{(1-x_0)^{k+1}}
\end{aligned}
$$

Daraus ergibt sich zunächst die Reihe

$$T_f(x, x_0) = \sum_{k=0}^{\infty} \frac{(x-x_0)^k}{(1-x_0)^{k+1}} \,.$$

Für den Entwicklungspunkt $x_0 = 0$ erhalten wir die Geometrische Reihe

$$T_f(x, 0) = \sum_{k=0}^{\infty} x^k \,,$$

von der wir ja wissen, dass sie nur auf dem Bereich $(-1, 1)$ konvergiert (s. Abb. 5).

3.

$$f(x) = \mathrm{e}^{-\frac{1}{x^2}} \,.$$

Für eine Taylorentwicklung mit Mittelpunkt $x_0 = 0$ benötigen wir alle Ableitungen an

diesem Punkt:

$$
\begin{aligned}
f(0) &= \lim_{x\to 0} \mathrm{e}^{-\frac{1}{x^2}} = 0 \\
f'(0) &= \lim_{x\to 0} \left(\mathrm{e}^{-\frac{1}{x^2}}\right)' = \lim_{x\to 0} \frac{2}{x^3}\mathrm{e}^{-\frac{1}{x^2}} = 0 \\
f''(0) &= \lim_{x\to 0} \left(-\frac{6}{x^4} + \frac{4}{x^6}\right) \mathrm{e}^{-\frac{1}{x^2}} = 0 \\
&\vdots \\
f^{(n)}(0) &= 0
\end{aligned}
$$

$T_f(x, 0) = 0$ ist die Taylorreihe von $f$. Ohne darüber gesprochen zu haben sind wir uns sicher einig, dass es sich hierbei nicht um eine gute Approximation handelt.

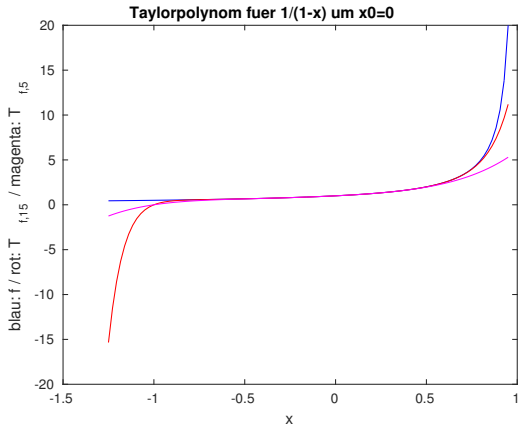



Abbildung 5: $T_{f,n}(x, x_0)$ für $f(x) = \frac{1}{1-x}$ um $x_0 = 0$ mit $n \in \{5, 15\}$

Jede in $x_0$ unendlich oft differenzierbare Funktion kann um $x_0$ einer Taylorreihe $T_f(x, x_0)$ zugeordnet werden. Diese Potenzreihe ist der einzig mögliche Kandidat, welcher die Funktion $f$ in einer Umgebung von $x_0$ darstellen könnte. Die Taylorreihe stellt jedoch nicht notwendigerweise die Funktion $f$ dar. (s. Beispiel 3: 2.,3.)

**Definition 1.3** (Konvergenzradius ). *Falls für fast alle Koeffizienten $a_n$ einer Potenzreihe*

$$\sum_{k=0}^{\infty} a_n\, x^n$$

*$a_n \neq 0$ gilt und der Grenzwert $\lim_{n\to\infty} \left|\frac{a_{n+1}}{a_n}\right|$ existiert, so ist der Konvergenzradius der Potenzreihe durch*

$$\rho = \frac{1}{\lim_{n\to\infty} \left|\frac{a_{n+1}}{a_n}\right|}$$

*gegeben.*

*Eine Potenzreihe*

$$\sum_{n=0}^{\infty} a_n (x - x_0)^n$$

*mit Konvergenzradius $\rho$ ist für alle $x \in (x_0 - \rho, x_0 + \rho)$ konvergent und divergiert für $|x - x_0| > \rho$. Das Intervall*

$$K = (x_0 - \rho, x_0 + \rho)$$

*heißt* Konvergenzintervall *oder auch* Konvergenzbereich *der Potenzreihe.*

*Das Konvergenzverhalten an den beiden Randpunkten $x_0 \pm \rho$ muss separat untersucht werden. Dazu setzen wir $x = x_0 \pm \rho$ und untersuchen die entstehende Reihe wie gewohnt auf Konvergenzkriterien.*

**Beispiel 4** Wir untersuchen eben die Konvergenzbereiche der Taylorreihen aus Beispiel 3:

1.

$$\begin{aligned}
a_n &= \frac{(-1)^n}{(2\,n+1)!} \\
a_{n+1} &= \frac{(-1)^{n+1}}{(2\,n+3)!} \\
\Rightarrow \qquad \left|\frac{a_{n+1}}{a_n}\right| &= \frac{-1}{(2\,n+2)\,(2\,n+3)} \overset{n \to \infty}{\longrightarrow} 0 \\
\Rightarrow \qquad \rho &\overset{n \to \infty}{\longrightarrow} \infty \\
\Rightarrow \qquad K &= (x_0 - \infty, x_0 + \infty) = \mathbb{R}
\end{aligned}$$

Die Taylorreihe von $\sin x$ konvergiert auf ganz $\mathbb{R}$.

2.

$$\begin{aligned}
a_n &= 1 \\
a_{n+1} &= 1 \\
\Rightarrow \qquad \left|\frac{a_{n+1}}{a_n}\right| &= 1 \\
\Rightarrow \qquad \rho &\overset{n \to \infty}{\longrightarrow} 1 \\
\Rightarrow \qquad K &= (-1, 1)
\end{aligned}$$

Die Taylorreihe von $\frac{1}{1-x}$ konvergiert auf $(-1,1)$, also für $|x|<1$. Da sowohl für $x=-1$ die Reihe

$$\sum_{k=0}^{\infty}(-1)^k$$

als auch für $x=1$ die Reihe

$$\sum_{k=0}^{\infty}1^k$$

die jeweilige Reihe divergiert, gehören die Randpunkte des offenen Intervalls auch nicht zum Konvergenzbereich.

3. Formal erhalten wir für bestimme $c_k \in \mathbb{R}$ und $n \geq 1$

$$a_n = \frac{f^{(n)}(x_0)}{n!} = \frac{1}{n!}\left(\sum_{k=1}^{n}\frac{c_k}{x_0^{n+2\,k}}\right)\,\mathrm{e}^{-\frac{1}{x_0^2}}$$

$$a_{n+1} = \frac{f^{(n+1)}(x_0)}{(n+1)!} = \frac{1}{(n+1)!}\left(\sum_{k=1}^{n+1}\frac{c_k}{x_0^{n+1+2\,k}}\right)\,\mathrm{e}^{-\frac{1}{x_0^2}}$$

$$\Rightarrow \qquad \frac{a_{n+1}}{a_n} = \frac{1}{(n+1)}\,\frac{\frac{1}{x_0}\,\sum_{k=1}^{n}\frac{c_k}{x_0^{n+2\,k}}+\frac{c_{n+1}}{x_0^{3\,n+3}}}{\sum_{k=1}^{n}\frac{c_k}{x_0^{n+2\,k}}}$$

$$\leq \frac{c_{\max}}{c_{\min}}\frac{1}{(n+1)}\,\frac{\frac{1}{x_0}\,\sum_{k=1}^{n}\frac{1}{x_0^{n+2\,k}}+\frac{1}{x_0^{3\,n+3}}}{\sum_{k=1}^{n}\frac{1}{x_0^{n+2\,k}}} \qquad C := \frac{c_{\max}}{c_{\min}}$$

$$= C\frac{1}{(n+1)}\left(\frac{1}{x_0}+\frac{1}{\sum_{k=1}^{n}\frac{x_0^{3\,n+3}}{x_0^{n+2\,k}}}\right)$$

$$= C\frac{1}{(n+1)}\left(\frac{1}{x_0}+\frac{1}{x_0^{2\,n+1}+x_0^{2\,n-1}+\cdots+x_0^3}\right)$$

$$\overset{x_0\,\to\,0}{\longrightarrow}\ \infty$$

$$\Rightarrow \qquad \left|\frac{a_{n+1}}{a_n}\right| \overset{n\,\to\,\infty}{\longrightarrow} = \infty$$

$$\Rightarrow \qquad \rho = 0$$

Die Reihe konvergiert also nur im Punkt $0$, womit sich der Konvergenzbereich auf ein mageres

$$K = \{0\}$$

beschränkt. So spielt das Leben eben manchmal....

Auf ihrem Konvergenzbereich $K$ ist eine Potenzreihe eine Funktion. Selbstredend ist eine Funktionsapproximation mittels Taylorpolynom auch nur auf dem Konvergenzbereich der entsprechenden Taylorreihe sinnvoll.

**Satz 1.4** (Restfunktion, Fehler). *Sei $f$ $(n+1)$-mal differenzierbar. Dann gilt für die Restfunktion $R_{f,n}$ der Taylorapproximation*

$$f(x) = T_{f,n}(x, x_0) + R_{f,n}(x, x_0)\,,$$

*dass $R_{f,n}(x_0, x_0) = 0$ gilt und für $x \neq x_0$ erhalten wir die Darstellung*

$$R_{f,n}(x, x_0) = \frac{f^{(n+1)}(\zeta)}{(n+1)!}\,(x - x_0)^{n+1}$$

*mit einer Stelle $\zeta$ zwischen $x$ und $x_0$. Falls zusätzlich die $(n+1)$-te Ableitung $f^{(n+1)}$ beschränkt ist auf $I$, also eine Konstante $C > 0$ existiert mit*

$$\left|f^{(n+1)}(\zeta)\right| \leq C \qquad \forall \zeta \in I\,,$$

*dann gilt*

$$|R_{f,n}(x, x_0)| \leq \frac{C}{(n+1)!}\,|x - x_0|^{n+1} \qquad \forall x \in I\,.$$

Beispiel 5 Restgliedabschätzung

Auf $I = \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ schätzen wir das Restglied

$$R_{\sin,3}(x, 0)$$

ab:

$$|R_{\sin,3}(x,0)| = \left|\frac{\sin^{(4)}(\zeta)}{4!}\,x^4\right| = \frac{|\sin(\zeta)|}{4!}\,x^4 \leq \frac{|x|^5}{4!} \leq \frac{\pi^5}{2^5\,4!} < 0.4$$

**Definition 1.5** (taylorentwickelbar ). *Wir sagen $f : I \to \mathbb{R}$ ist taylorentwickelbar wenn auf $I \subset \mathbb{R}$*

$$f(x) = T_{f,n}(x, x_0) + R_{f,n}(x, x_0)\,,$$

*und*

$$\lim_{n\to\infty} R_{f,x_0}(x, x_0) = 0$$

*gilt.*

Wir können nun das Integral oder auch die Ableitung einer Funktion $f$ näherungsweise durch Integration oder Differentiation des zugehörigen Taylorpolynoms berechnen:

Integration:

$$\int f(x)\,dx \approx \int T_{f,n}(x,x_0)\,dx = \int \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}\,(x-x_0)^k\,dx$$
$$= \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{(k+1)!}\,(x-x_0)^{k+1} + C$$

Differentiation:

$$\frac{d}{dx}\,f(x) \approx \frac{d}{dx} T_{f,n}(x,x_0) = \frac{d}{dx} \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}\,(x-x_0)^k = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{(k-1)!}\,(x-x_0)^{k-1}$$

# –2–
# Lösen linearer Gleichungssysteme

Problemstellung:

Zu einer gegebenen Matrix $a \in \mathbb{R}^{n\times n}$ und einem gegebenen Vektor $b \in \mathbb{R}^n$ ist der Vektor $x \in \mathbb{R}^n$ gesucht, der das lineare Gleichungssystem (kurz *LGS*) der Form

$$A\,x = b$$

löst.

Wir wissen aus der linearen Algebra, dass so ein Problem nicht immer und wenn nicht zwingend eindeutig lösbar ist. In unserem Fall soll das aber so sein, dass heißt wir gehen getrost von Matrizen mit $\det A \neq 0$ aus. Im Folgenden wollen wir uns damit beschäftigen, wie sich die Lösung eines solchen LGS numerisch, also rechnergestützt berechnen lässt. Wir werden uns dabei mit zweierlei grundsätzlicher Methoden beschäftigen, nämlich der direkten Methode zum einen und einer iterativen Methode zum anderen. Als Beispiel einer direkten Methode haben Sie sicher schon den Gauß-Algorithmus kennengelernt. So was Ähnliches machen wir hier auch. Abgesehen von RUndungsfehlern, die beim Rechnen mit Computern unumgänglich sind erhalten wir bei direkten Methoden genau die gesuchte Lösung, während die iterative Methode in der praktischen Anwendung lediglich Näherungslösungen anbietet; zumindest darf man nicht mehr erwarten. Wenn man mit einer Näherungslösung zufrieden ist hat man den Vorteil, dass solche Näherungsverfahren in der Regel schneller sind.

## 2.1 Ein direktes Verfahren: Das LR-Verfahren

### 2.1.1 Verfahrensidee

Die Idee der LR-Zerlegung besteht darin, die Matrix $A$ in ein Produkt aus einer linken unteren Dreiecksmatrix $L$ und einer rechten oberen Dreiecksmatrix $R$ zu zerlegen der Gestalt, dass

$$\begin{aligned} A &= L \cdot R \\ &= \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ l_{21} & 1 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & 1 & 0 \\ l_{11} & \cdots & \cdots & l_{n,n-1} & 1 \end{pmatrix} \cdot \begin{pmatrix} r_{11} & r_{12} & \cdots & \cdots & r_{1n} \\ 0 & r_{22} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & r_{n-1,n-1} & r_{n-1,n} \\ 0 & \cdots & \cdots & 0 & r_{nn} \end{pmatrix} \end{aligned}$$

gilt. Wir können dann aus einem LGS zwei machen, was zunächst nicht nach einem Fortschritt sondern nach einem Rückschritt aussieht, aber bei genauerem Hinschauen stellen wir fest, dass sich die Lösung $x$ aus den beiden neuen LGS' geradewegs herauslesen lässt. Die Situation verhält sich dann nämlich so: Gegeben sind $A, b$, gesucht ist $x$ mit

$$\begin{aligned} & & A\,x &= b \\ \Leftrightarrow & & L \cdot \underbrace{R\,x}_{=:y} &= b \\ \Leftrightarrow & & L\,y = b \;\wedge\; R\,x &= y \end{aligned}$$

Die beiden Systeme $L\,y = b$ und $R\,x = y$ kann man sukkzessive, bzw. rekursiv auflösen, da die Matrizen jeweils Dreiecksmatrizen sind. Die eigentliche Arbeit steckt also darin, die Matrizen $L$ und $R$ überhaupt aufzustellen. Wie kann man diese bestimmen?

Die Idee besteht darin, genau wie beim Gauß-Algorithmus, sukkzessive die Spaltenanteile unter der Diagonalen in der Matrix $A$ zu "verlieren". Kontruiere im ersten Schritt eine Matrix $L_1$, so dass

$$L_1 \cdot A = \begin{pmatrix} \star & \star & \cdots & \star \\ 0 & \star & & \star \\ \vdots & \vdots & & \star \\ 0 & \star & \cdots & \star \end{pmatrix}$$

und dann eine Matrix $L_2$ so dass

$$L_2 \cdot L_1 \cdot A = \begin{pmatrix} \star & \star & \cdots & \star \\ 0 & \star & & \star \\ \vdots & 0 & \ddots & \star \\ 0 & 0 & \cdots & \star \end{pmatrix}$$

bis

$$L_{n-1} \cdots L_1 \cdot A = \underbrace{\begin{pmatrix} \star & \star & \cdots & \star \\ 0 & \ddots & & \star \\ \vdots & 0 & \ddots & \star \\ 0 & \cdots & 0 & \star \end{pmatrix}}_{=:R}$$

erreicht ist. Damit haben wir dann bereits die obere Dreieckmatrix $R$ berechnet und es gilt insgesamt schon mal

$$L_{n-1} \cdots L_1 \cdot A = R$$

bzw.

$$A = L_1^{-1} \cdots L_n^{-1}\, R\,.$$

Die Matrizen $L_i$ sind von der Form

$$L_i = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ \star & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ \star & & \cdots & 1 \end{pmatrix}$$

und heißen *Frobeniusmatrizen*. Matrizen dieser Form lassen sich sehr einfach invertieren. Schauen wir uns das an einem Beispiel an:

Beispiel 6

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 2 & 3 \\ 3 & 5 & 4 \end{pmatrix}$$

$$L_1 A = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 2 & 3 \\ 3 & 5 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 1 \\ 0 & -2 & 1 \\ 0 & -1 & 1 \end{pmatrix}$$

$$L_2 L_1 A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 0 & -2 & 1 \\ 0 & -1 & 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 2 & 1 \\ 0 & -2 & 1 \\ 0 & 0 & \frac{1}{2} \end{pmatrix}}_{=:R}$$

$$L = L_1^{-1} L_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & \frac{1}{2} & 1 \end{pmatrix}$$

Insgesamt haben wir jetzt folgende Zerlegung berechnet:

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 2 & 3 \\ 3 & 5 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & \frac{1}{2} & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 1 \\ 0 & -2 & 1 \\ 0 & 0 & \frac{1}{2} \end{pmatrix} = L \cdot R$$

Wollen wir etwa das LGS

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 2 & 3 \\ 3 & 5 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

lösen, so formulieren wir zunächst um zu

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & \frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$

erhalten darüber

$$y = \begin{pmatrix} 1 \\ -1 \\ -\frac{3}{2} \end{pmatrix},$$

setzen das in

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & -2 & 1 \\ 0 & 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ -\frac{3}{2} \end{pmatrix}$$

ein und erhalten schlussendlich

$$x = \begin{pmatrix} 6 \\ -1 \\ -3 \end{pmatrix}.$$

Hübsch, nicht war?

Wir schauen uns einmal an wie das Berechnungsprinzip der Matrizen $L_i$ aussieht. Das wird schon klar, wenn wir uns den ersten Schritt vor Augen führen: Es sei $A^0 := A$

$$L_1 \cdot A^0 = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{a_{21}^0}{a_{11}^0} & 1 & 0 \\ -\frac{a_{31}^0}{a_{11}^0} & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} a_{11}^0 & a_{12}^0 & a_{13}^0 \\ a_{21}^0 & a_{22}^0 & a_{23}^0 \\ a_{31}^0 & a_{32}^0 & a_{33}^0 \end{pmatrix} = \begin{pmatrix} a_{11}^0 & a_{12}^0 & a_{13}^0 \\ 0 & a_{22}^1 & a_{23}^1 \\ 0 & a_{32}^1 & a_{33}^1 \end{pmatrix} =: A^1$$

Das ist so was ähnliches wie der Gauß-Algorithmus und lässt sich ebenso ähnlich sukkzessive auf alle Spalten fortführen, in denen zu eliminierende Koeffizienten auftauchen.

$$L_2 \cdot L_1 \cdot A^0 = L_2 \cdot A^1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{a_{32}^1}{a_{22}^1} & 1 \end{pmatrix} \cdot \begin{pmatrix} a_{11}^0 & a_{12}^0 & a_{13}^0 \\ 0 & a_{22}^1 & a_{23}^1 \\ 0 & a_{32}^1 & a_{33}^1 \end{pmatrix} = \begin{pmatrix} a_{11}^0 & a_{12}^0 & a_{13}^0 \\ 0 & a_{22}^1 & a_{23}^1 \\ 0 & 0 & a_{33}^2 \end{pmatrix}$$

Bei einer $n \times n$-Matrix erhalten wir also im $k$-ten Schritt den Eintrag in der $k$-ten Spalte von $L_k$ gemäß

$$L_k = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & 0 & 1 & \ddots & & \vdots \\ \vdots & & \vdots & -\frac{a_{k+1,k}^{k-1}}{a_{kk}^{k-1}} & \ddots & \ddots & \vdots \\ \vdots & & \vdots & \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & -\frac{a_{n,k}^{k-1}}{a_{kk}^{k-1}} & 0 & 0 & 1 \end{pmatrix}$$

**LGS/MyLR.m**:

```
function x=MyLR(A,y)

% A=LR
[L R]=LRDecomp(A);

% Ax=y <=> LRx=y <=> Lb=y AND Rx=b
b = LSolve(L,y);
x = RSolve(R,b);

end
```

Subfunktion **(L R)=LRDecomp(A)**:

```
L = eye(n);
R=A;

for k=1:n-1

  for i=k+1:n

    L(i,k) = R(i,k)/R(k,k);
    R(i,k) =0;

    for j=k+1:n
      R(i,j) = R(i,j) - L(i,k)*R(k,j);
    end

  end
end
```

Subfunktion **b=LSolve(L,y)**:

```
n = length(y);

for k=1:n
   b(k)=y(k);
  for i=1:k-1
     b(k) = b(k) - L(k,i)*b(i);
   end
end

end
```

Subfunktion **y=RSolve(R,b)**:

```
n = length(b);

for k=n:-1:1
   x(k) = b(k);
   for i=k+1:n
     x(k) = x(k) - x(i)*R(k,i);
   end
   x(k) = x(k)/R(k,k);
end

end
```

### 2.1.2 Pivotisierung

Da bei der Berechnung der $L_i$ mit Diagonalelementen von $A^i$ dividiert wird, muss der Möglichkeit Rechnung getragen werden, dass diese auch unter Umständen einmal Null sind. Genau wie beim Gauß-Verfahren führt man dann eine einfache Pivotisierung, also geschicktes Vertauschen von Zeilen durch, wobei dann auch die entsprechenden Komponenten in $y$ vertauscht werden muss. Wir betrachten das Verfahren mit Pivotisierung anhand eines kleinen Beispiels:

Beispiel 7 Pivotisierung

Bei der Pivotisierung geht man so vor, dass man in jedem Schritt $i$ zur Berechnung von $L_i$ zunächst die Zeilen der Matrix so vertauscht, dass man das betragsmäßig größte Element als Pivotelement erhält.

$$A = \begin{pmatrix} -1 & 2 & 3 \\ -2 & 7 & 4 \\ 1 & 4 & -2 \end{pmatrix} \quad \text{vertausche Zeile 2 mit 1, denn } |-2| > |-1|$$

$$\Rightarrow \qquad P_1 A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1 & 2 & 3 \\ -2 & 7 & 4 \\ 1 & 4 & -2 \end{pmatrix} = \begin{pmatrix} -2 & 7 & 4 \\ -1 & 2 & 3 \\ 1 & 4 & -2 \end{pmatrix}$$

$$\Rightarrow \qquad L_1 = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ \frac{1}{2} & 0 & 1 \end{pmatrix} \quad \text{und} \quad A_1 = L_1 P_1 A = \begin{pmatrix} -2 & 7 & 4 \\ 0 & -\frac{3}{2} & 1 \\ 0 & \frac{15}{2} & 0 \end{pmatrix}$$

$$\Rightarrow \qquad P_2 A_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} -2 & 7 & 4 \\ 0 & -\frac{3}{2} & 1 \\ 0 & \frac{15}{2} & 0 \end{pmatrix} = \begin{pmatrix} -2 & 7 & 4 \\ 0 & \frac{15}{2} & 0 \\ 0 & -\frac{3}{2} & 1 \end{pmatrix}$$

$$\Rightarrow \qquad L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{5} & 1 \end{pmatrix}$$

$$\Rightarrow \qquad L_2\, P_2\, L_1\, P_1\, A = \begin{pmatrix} -2 & 7 & 4 \\ 0 & \frac{15}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} = R$$

Wie sieht jetzt die untere Dreicksmatrix $L$ aus? Es ist ja

$$\begin{aligned} & L_2\, P_2\, L_1\, P_1\, A = R \\ \Leftrightarrow \qquad & L_2 \underbrace{P_2\, L_1\, P_2^{-1}}_{=:\tilde{L}_1} \underbrace{P_2\, P_1}_{=:P} A = R \\ \Leftrightarrow \qquad & L_2\, \tilde{L}_1\, P\, A = R \\ \Leftrightarrow \qquad & P\, A = \underbrace{\tilde{L}_1^{-1}\, L_2^{-1}}_{=:L} R \\ \Leftrightarrow \qquad & P\, A = L\, R \end{aligned}$$

Dabei ist

$$\tilde{L}_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ \frac{1}{2} & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ -\frac{1}{2} & 0 & 1 \end{pmatrix},$$

$$L = \tilde{L}_1^{-1}\, L_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ \frac{1}{2} & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{5} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ \frac{1}{2} & -\frac{1}{5} & 1 \end{pmatrix}$$

und

$$P_2\, P_1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

Wir können dann unser Gleichungssystem umformulieren zu:

$$
\begin{aligned}
& A\,x = y \\
\Leftrightarrow \quad & P\,A\,x = P\,y \\
\Leftrightarrow \quad & L\,R\,x = P\,y
\end{aligned}
$$

ÜA: Lösen Sie das LGS mit $y = (4, 9, 3)^T$. Es sollte als Lösung $x = (1, 1, 1)^T$ herauskommen.

Wenn ein LGS durch eine reguläre Matrix beschrieben wird, dann liefert die LR-Zerlegung (mit Pivotisierung) sicher eine Lösung.

Man kann zusätzlich Speicher sparen, wenn man nicht zwei Matrizen $L, R \in \mathbb{R}^{n\times n}$ belegt, sondern alles zusammen in eine:

$$
LR = \begin{pmatrix}
r_{11} & \cdots & \cdots & r_{1n} \\
l_{21} & r_{22} & \cdots & r_{2n} \\
\vdots & \ddots & \ddots & \vdots \\
l_{n1} & \cdots & l_{n,n-1} & r_{nn}
\end{pmatrix}
$$

## 2.2 Ein iteratives Verfahren: Das SOR-Verfahren

Im Gegensatz zu direkten Verfahren steht das iterative Verfahren. Hier starten wir mit einem beliebigen Startwert $x^0$ und erhalten in jedem Iterationsschritt Zwischenwerte $x^i$, die sich sukkzessive an die gesuchte Lösung $x = x^\star$, mit $A\,x^\star = y$ annähern. Der Vorteil bei iterativen Verfahren besteht darin, dass, wenn man nur an Näherungswerten interessiert ist, der Rechenaufwand zur Berechnung einer Näherungslösung deutlich geringer ist als der bei direkten Verfahren. Die Iterationsvorschrift muss natürlich so gewählt oder konstruiert sein, dass jeder weitere Iterationsschritt die Zwischenlösung $x^i$ der gesuchten auch näher bringt. Hinter der Idee steckt die sogenannte Fixpunktiteration, die dafür sorgt, dass

$$\|x^\star - x^{i+1}\| \leq \|x^\star - x^i\|$$

erfüllt ist.

### 2.2.1 Fixpunktiteration

Gegeben ist ein LGS der Form

$$A\,x = b\,. \tag{1}$$

Die grundlegende Idee besteht darin, zu einem gegebenen Startwert $x^0$ ein Iterationsverfahren

$$x^{k+1} = F(x^k)$$

aufzustellen mit den Eigenschaften

$$x^\star = \lim_{k\to\infty} x^k = \lim_{k\to\infty} F(x^k) = F(x^\star)$$

$$\text{für das gilt} \qquad A\,x^\star = y\,.$$

Dieses $x^\star$ heißt *Fixpunkt* und eine solche Iteration nennen wir *Fixpunktiteration*. Schauen wir uns doch zunächst mal an wie so eine Abbildung aussehen könnte. Das Grundkonzept der Konstruktion einer solchen Iteration sieht so aus: Wir zerlegen die Matrix $A$ in eine Differenz zweier Matrizen $M$ und $N$ gemäß:

$$A = M - N$$

Dann gilt

$$\begin{aligned} & (M-N)\,x = y \\ \Leftrightarrow \quad & M\,x = N\,x + y \\ \Leftrightarrow \quad & x = M^{-1}\,N\,x + M^{-1}\,y \\ \Leftrightarrow \quad & =: T\,x + z \end{aligned}$$

Wenn nun, umgekehrt gedacht, die implizite Folge

$$\begin{aligned} x^0 \quad & \text{gegebener Startwert} \\ x^{i+1} &= T\,x^i + z\,, \quad i = 0, 1, 2, \ldots \end{aligned} \tag{2}$$

konvergiert, dann ist der Grenzwert $x$ die Lösung von (1). Es stellt also die alles entscheidende Frage: "Unter welcher Bedingung konvergiert die Iteration?" Um diese Frage zu beantworten, benötigen wir ein Mittel, um Matrizen zu "messen". Eine Norm, wie wir sie aus der Linearen Algebra kennen, um Längen von Vektoren zu messen.

**Definition 2.1** (induzierte Matrixnorm ). *Eine Matrixnorm $\|\cdot\|$ heißt von einer Vektornorm $\|\cdot\|_V$ induzierte oder natürliche Matrixnorm, wenn sie von ihr als Operatornorm abgeleitet ist. Die natürliche Matrixnorm einer reellen oder komplexen Matrix $A \in \mathbb{C}^{m\times n}$ ist damit definiert als*

$$\|A\| := \max_{x\neq 0} \frac{\|Ax\|_V}{\|x\|_V},$$

*wobei die Norm im Zähler als Argument einen Vektor $Ax \in \mathbb{C}^m$ und die Norm im Nenner als Argument einen Vektor $x \in \mathbb{C}^n$ besitzt. Da es zu jedem Vektor $x \neq 0$ mit $\bar{x} := \frac{x}{\|x\|_V}$ einen auf Eins normierten Vektor gibt, hat jede natürliche Matrixnorm auch die Darstellung*

$$\|A\| = \max_{\|x\|_V=1} \|Ax\|_V,$$

*es reicht also aus, das Maximum über alle Einheitsvektoren zu betrachten.*

Eine induzierte Matrixnorm ist eine von einer Vektornorm als Operatornorm abgeleitete Matrixnorm. Sie entspricht anschaulich dem größtmöglichen Streckungsfaktor, der durch die Anwendung der Matrix auf einen Vektor entsteht.

**Beispiel 8** Spaltensummennorm

$$\|A\|_1 = \max_j \sum_{k=1}^{n} |A_{kj}|$$

$$\left\| \begin{pmatrix} -1 & 2 \\ -3 & 1 \end{pmatrix} \right\|_1 = \max_j \{|-1|+|-3|, |2|+|1|\} = \max_j \{4, 3\} = 4$$

**Beispiel 9** Zeilensummennorm

$$\|A\|_\infty = \max_k \sum_{j=1}^{n} |A_{kj}|$$

$$\left\| \begin{pmatrix} -2 & 2 \\ -3 & 6 \end{pmatrix} \right\|_\infty = \max_j \{|-2|+|2|, |-3|+|6|\} = \max_j \{4, 9\} = 9$$

**Beispiel 10** Spektralnorm

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)}$$

$$\left\| \begin{pmatrix} -2 & 2 \\ -3 & 6 \end{pmatrix} \right\|_1 = \max_j \{|-2|+|2|, |-3|+|6|\} = \max_j \{4, 9\} = 9$$

Ist $A$ symmetrisch, so gilt

$$\|A\|_2 = |\lambda_{\max}|\,,$$

wobei $\lambda_{\max}$ den batragsmäßig größten Eigenwert beschreibt.

Ist $A$ symmetrisch und positiv definit, so gilt

$$\|A\|_2 = \lambda_{\max}\,,$$

wobei $\lambda_{\max}$ den größten Eigenwert beschreibt. Das liegt daran, dass positiv definite Matrizen nur positive Eigenwerte besitzen:

$$0 \; < < A\,v, v > = < \lambda\,v, v > = \lambda\,\|v\|^2 \quad \Rightarrow \quad \lambda > 0$$

Das Analoge gilt für positiv, semidefiniten Matrizen. Hieru ein Beispiel.

**Beispiel 11**

Für die Matrix

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}$$

erhalten wir die Eigenwerte

$$\lambda_1 = 0 \quad \text{und} \quad \lambda_2 = 5$$

mit zugehörigen Eigenvektoren

$$v_1 = \begin{pmatrix} -0.8944 \\ 0.4472 \end{pmatrix} \quad \text{und} \quad v_2 = \begin{pmatrix} 0.4472 \\ 0.8944 \end{pmatrix}.$$

Der entsprechende Befehl in Matlab lautet

```
>> [V,D]=eig([[1 2];[2 4]])
```

Die Spektralnorm ist gegeben durch

$$\|A\|_2 = 5\,,$$

was Sie schnell in Matlab mit

```
>> norm([[1 2];[2 4]],2)
```

nachrechnen können. Die Matrix ist positiv semidefinit, denn es gilt:

$$\begin{aligned} < A\,x, x > &= \left\langle \begin{pmatrix} x_1 + 2\,x_2 \\ 2\,x_1 + 4\,x_2 \end{pmatrix}, x \right\rangle \\ &= x_1^2 + 4\,x_1\,x_2 + 4\,x_2^2 \\ &= \underbrace{(x_1 + 2\,x_2)^2}_{\geq 0} \\ &= 0 \\ \Leftrightarrow \qquad x_1 &= -2\,x_2 \end{aligned}$$

was für den Eigenvektor $v_1$ zum Eigenwert $\lambda_1 = 0$ ja gerade erfüllt ist.

**Hilfssatz 2.2.** *Für jede induzierte Matrixnorm*

$$\|A\| = \max_{\|x\|_V = 1} \|A\,x\|_V$$

*gilt stets die Abschätzung*

$$\|A\,x\|_V \leq \|A\|\,\|x\|_V\,.$$

**Beweis Hilfssatz 2.2:**

$$\|A\| = \max_{\|x\|} \frac{\|A\,x\|_V}{\|x\|_V} \geq \frac{\|A\,v\|_V}{\|v\|_V} \qquad \Leftrightarrow \qquad \|A\,v\|_V \leq \|A\|\|v\|_V \quad \forall\, v \in \mathbb{R}^n$$

□

**Satz 2.3.** *Die Iteration (2) konvergiert, wenn die Matrix $T$ kontrahierende Eigenschaft hat, d.h. wenn*

$$\|T\| \leq L < 1$$

*gilt.*

**Beweis Satz 2.3:**

*Mit Hilfssatz 2.2 können wir zeigen, dass*

$$\|T\,x^k - x^k\| \leq L^k\,\|x^1 - x^0\|$$

*gilt:*

$$\begin{aligned}
\|T\,x^k - x^k\| &= \|T\,x^k - T\,x^{k-1}\| \\
&\leq \|T\|\,\|x^k - x^{k-1}\| \\
&\leq L\,\|T\,x^{k-1} - T\,x^{k-2}\| \\
&\leq L^2\,\|x^{k-1} - x^{k-2}\| \\
&\vdots \\
&\leq L^k\,\|x^1 - x^0\|
\end{aligned}$$

*Und damit stellen wir sicher, dass $x^k$ eine Cauchy-Folge ist. Es gilt nämlich*

$$\begin{aligned}
\|x^m - x^n\| &\leq \|x^m - T\,x^m\| + \|T\,x^m - T\,x^n\| + \|T\,x^n - x^n\| \\
&\leq L^m\,\|x^1 - x^0\| + L\,\|x^m - x^n\| + L^n\,\|x^1 - x^0\| \\
\Leftrightarrow \qquad \|x^m - x^n\| &\leq \frac{L^m + L^n}{1 - L}\,\|x^1 - x^0\|
\end{aligned}$$

*Falls $L < 1$ gilt nun*

$$\lim_{n,m\to\infty} \|x^m - x^n\| = 0\,.$$

*Damit ist $x^k$ eine Cauchy-Folge in $\mathbb{R}$ und besitzt dort einen Grenzwert*

$$x = \lim_{n\to\infty} x^n\,.$$

□

Die Iteration konvergiert, wenn die Abbildung kontrahierend ist. Das ist eine hinreichende Bedingung.

### 2.2.2 Jacobi und Gauß-Seidel

Beim *Jacobi-Verfahren* wählen wir für $M$ eine die Diagonalmatrix mit den Diagonalelementen von $A$:

$$M_{ij} = A_{ij}\delta_{ij} = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{pmatrix}$$

und in folge dessen natürlich

$$N = M - A = \begin{pmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & \ddots & & \vdots \\ \vdots & & \ddots & -a_{n-1,n} \\ -a_{n,1} & \cdots & -a_{n,n-1} & 0 \end{pmatrix},$$

denn es soll ja auf jeden Fall $A = M - N$ gelten. Wir können damit folgende Umrechnung durchführen.

$$\begin{aligned} & A\,x = y \\ \Leftrightarrow \quad & x = M^{-1}\,N\,x + M^{-1}\,y \\ & = M^{-1}\,(M - A)\,x + M^{-1}\,y \\ & = (I - M^{-1}\,A)\,x + M^{-1}\,y \\ & = T\,x + z \end{aligned}$$

Die Iteration lautet dann: Berechne $x^k$ zu einem gegebenen Startwert $x^0$ gemäß

$$\begin{aligned} x^{k+1} &= T\,x^k + z & T &= (I - D^{-1}\,A)\,,\ z = D^{-1}\,y \\ &= F(x^k) & F(x) &= Tx + z \end{aligned}$$

**J-Verfahren (Gesamtschrittverfahren ):**

Gegeben ist ein Startvektor $x^0$, Berechne $x^{k+1}$ aus $x^k$ für $k = 0, \ldots$

```
for i=1,n
```

$$x_i^{k+1} = \left( y_i - \sum_{j=1,\, j\neq i}^{n} a_{ij}\, x_j^k \right) \frac{1}{a_{ii}}$$

```
end
```

function **x=MyJOR(A,y,K)**: % Version 1

```
N=length(y);
xalt=zeros(1,N);
x=xalt;
for k=1:K
  for i=1:N
    summ=0;
    for j=1:N
      if j~=i
        summ = summ + A(i,j)*xalt(j);
      end
    end
    x(i) = (y(i)-summ)/A(i,i);
  end
  xalt = x;
end

end
```

Anstatt die genaue Anzahl der Iterationen vorzugeben, ist es oft sinnvoller eine Toleranz `TOL` für das Residuum die fordern. Das Residuum beschreibt, wie gut die Näherung schon an der gesuchten Lösung heranreicht. Da wir die gesuchte Lösung ja nun nicht kennen untersuchen wir statt dessen die Abweichung

$$\mathrm{Res} = \|y - A\, x^k\|_V$$

in der gewünschten Norm, was meistens die $\|\cdot\|_2$-Norm ist. Der Algorithmus hat dann diese Form:

function **(x iter)=MyJOR(A,y,TOL)**: % Version 2

```
Res=10;
itermax=1000;
N=length(y);
xalt=zeros(1,N);
x=xalt;
while Res>TOL & iter<itermax
  for i=1:N
    summ=0;
    for j=1:N
      if j~=i
        summ = summ + A(i,j)*xalt(j);
      end
    end
    x(i) = (y(i)-summ)/A(i,i);
  end
  xalt = x;
  Res = norm(A*x'-y,2);
  iter = iter + 1;
end

end
```

Wenn man sich das Jacobi-Verfahren genau ansieht, stell man fest, dass man jeweils in der $i$-ten Zeile, bereits aktualisierte Werte

$$x_1^{k+1}, x_2^{k+1}, \ldots, x_{i-1}^{k+1}$$

statt

$$x_1^k, x_2^k, \ldots, x_{i-1}^k$$

verwenden kann. Das beschleunigt die Konvergenz und führt auf den neuen Algorithmus:

**Gauß-Seidel-Verfahren (Einzelschrittverfahren )**

Gegeben ist ein Startvektor $x^0$, Berechne $x^{k+1}$ aus $x^k$ für $k = 0, \ldots$

```
for i=1,n
```

$$x_i^{k+1} = \left( y_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}\, x_j^k \right) \frac{1}{a_{ii}}$$

```
end
```

Die Zerlegung der Matrix beim Einzelschrittverfahren war dann

$$A = L + D + R\,,$$

wobei $L$ die linke untere und $R$ die rechte obere Dreiecksmatrix aus $A$ beschreibt:

$$\begin{aligned} A &= L + D + R \\ &= \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ a_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{pmatrix} + \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{pmatrix} + \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & a_{n-1,n} \\ 0 & \cdots & \cdots & 0 \end{pmatrix} \end{aligned}$$

Damit formulieren wir das LGS um gemäß

$$\begin{aligned} & A\,x = y \\ \Leftrightarrow \quad & (L + D + R)\,x = y \\ \Leftrightarrow \quad & D\,x = y - L\,x - R\,x \\ \Leftrightarrow \quad & x = D^{-1}(y - L\,x - R\,x) \end{aligned}$$

Damit stellen wir die Iteration zu beliebigem Startwert $x^0 \in \mathbb{R}^n$

$$x^{k+1} = D^{-1}(y - L\,x^{k+1} - R\,x^k)$$

bzw. für jede Komponente $i \in \{1, \ldots, n\}$:

$$\begin{aligned} x_i^{k+1} &= (y_i - (L\,x^{k+1})_i - (R\,x^k)_i)\frac{1}{a_{ii}} \\ &= \left(y_i - \sum_{j=1}^{i-1} L_{ij} x_j^{k+1} - \sum_{j=i+1}^{n} R_{ij} x_j^k\right) \frac{1}{a_{ii}} \end{aligned}$$

Die Iterationsmatrix $T$ der Fixpunktiteration der Form

$$x^{k+1} = T\,x^k + z$$

ist gegeben durch

$$T = -(L + D)^{-1} R\,. \quad \text{(ÜA)}$$

function **(x iter)=MySOR(A,y,TOL)**: % Version 1

```
Res=10;
itermax=1000;
N=length(y);
while Res>TOL & iter<itermax
  for i=1:N
    summ=0;
    for j=1:N
      if j~=i
        summ = summ + A(i,j)*x(j);
      end
    end
    x(i) = (y(i)-summ)/A(i,i);
  end
  Res = norm(A*x'-y,2);
  iter = iter + 1;
end

end
```

**Definition 2.4** (vollständig konsistent ). *Ein Gleichungssystem $A\,x = y$ heißt vollständig konsistent mit einer Fixpunktgleichung $x = T\,x + z$, wenn jede Lösung der einen Gleichung auch Lösung der jeweils anderen ist.*

An der hinreichenden Konvergenzaussage in Satz 2.3 ist unbefriedigend, dass die Kontraktionsbedingung von der verwendeten Matrixnorm abhängig ist.

Beispiel 12

Für die symmetrische Matrix

$$T = \begin{pmatrix} 0.1 & -0.4 \\ -0.4 & 0.8 \end{pmatrix}$$

erhalten wir folgende Werte in den verschiedenen Matrixnormen

$$\begin{aligned} \|T\|_1 &= 1.2 > 1 \\ \|T\|_2 &= 0.9815 < 1 \\ \|T\|_\infty &= 1.2 > 1 \end{aligned}$$

Da der *Spektralradius*

$$\sigma(T) = \max_i |\lambda_i(t)|$$

für jede Matrixnorm eine untere Schranke bildet, ist er - und damit eben die $\|\cdot\|_2$-Norm für Matrizen - die entscheidende Größe für die Konvergenz der Fixpunktiteration. Mit ihm ist auch die notwendige Konvergenzordnung zu formulieren.

**Satz 2.5.** *Eine Fixpunktiteration $x^{k+1} = T\,x^k + z$, welche zum Gleichungssystem $A\,x = y$ vollständig konsistent ist, erzeugt genau dann für jeden beliebigen Startvektor $x^0$ eine gegen $x$ konvergente Folge, falls $\sigma(T) < 1$ ist.*

### 2.2.3 Konvergenzordnung

Neben der Frage ob ein Verfahren überhaupt konvergiert steht natürlich gleich als nächstes die Frage wie schnell das passiert. Wir haben im vorherigen Kapitel - bei den entsprechenden Tests im Praktikum - gesehen, dass das SOR weniger Iterationen benötigt als das JOR, um die gleiche Toleranz zu unterschreiten. Und wir haben auhc experimentell festgestellt, dass die Spektralnorm zur jeweiligen Iterationsmatrix, meist $T$ genannt, beim SOR kleiner ist als beim JOR. Wenn man sich den Beweis zu Satz 2.3 genau ansieht wird auch klar warum das so ist. Wenn die Kontraktionseigenschaft von $T$ stärker ist, konvergiert die zugehörige Folge eben schneller. Jetzt mal so salopp formuliert, denn es gilt für $L < P$ und $0 < L, P < 1$:

$$\frac{L^k}{1-L} < \frac{P^k}{1-P}$$

**Definition 2.6** (Konvergenzordnung ). *Die Folge $(x^k)_k$ mit Grenzwert $s$ hat mindestens die Konvergenzordnung $p \geq 1$, wenn es eine von $k$ unabhängige Konstante $K > 0$ gibt, so dass*

$$\|x^{k+1} - s\| \leq K\,\|x^k - s\|^p \qquad \forall k \geq 0$$

*mit $K < 1$, falls $p = 1$ ist. $K$ wird mit Konvergenzrate oder Fehlerkonstante bezeichnet.*

Die Konvergenz heißt *linear* wenn $p = 1$, *quadratisch* wenn $p = 2$ und *kubisch* wenn $p = 3$ ist. Eine höhere als kubische Konvergenz kommt nur selten vor. Ist die Konstante $K$ darüberhinaus eine Nullfolge $K_n \to 0$ so sagen wir, das Verfahren konvergiert *superlinear* (im Fall $p = 1$). $K$ und $p$ bestimmen die *Konvergenzgeschwindigkeit* der Folge. Je kleiner $K$ und größer $p$ ist, desto schneller konvergiert das Verfahren. Dabei bestimmt die Ordnung das Verhalten wesentlich stärker als die Konstante $K$. Diese ist wichtig beim Vergleich von linear konvergenten Verfahren wie es bei JOR und SOR gegeben ist.

Unsere Konvergenzkriterien richteten sich bisher immer auf die Iterationsmatrix $T$. Wie können wir aber der Matrix $A$ ansehen, dass das Gesamtschritt- oder Einzelschritt-Verfahren konvergieren wird?

**Satz 2.7.** *Gesamt- und Einzelschrittverfahren konvergieren für reguläre Matrizen $A$, die das starke Zeilen- oder das starke Spaltensummenkriterium erfüllen:*

*starkes Zeilensummenkriterium:*

$$\sum_{j=1,\,j\neq i}^{n} |a_{ij}| < |a_{ii}|\,, \quad \text{für jede Zeile } i$$

*starkes Spaltensummenkriterium:*

$$\sum_{i=1,\, j\neq i}^{n} |a_{ij}| < |a_{jj}|\,, \quad \textit{für jede Spalte } j$$

(ohne Beweis)

**Satz 2.8.** *Gesamt- und Einzelschrittverfahren konvergieren für reguläre Matrizen $A$, die das schwache Zeilensummenkriterium*

$$\sum_{j=1,\, j\neq i}^{n} |a_{ij}| \leq |a_{ii}|\,, \quad \textit{für jede Zeile } i$$

*und für wenigstens eine Zeile das starke Zeilensummenkriterium erfüllen.*
*Oder es ist das Analoge für das Spaltensummenkriterium erfüllt.*

(ohne Beweis)

Sehr wahrscheinlich ist die Erfüllung dieser Kriterien bei diagonal dominanten Matrizen. Das sind solche, die große Werte auf der Diagonalen haben. Anschaulich kann man sich klar machen, dass durch die Division mit Diagonalelementen die große Werte haben in der Fixpunktmatrix $T$, diese dann ihre benötigte kontrahierende Eigenschaft erhalten kann.

Beispiel 13

1. $$\begin{pmatrix} 10 & -4 & -2 \\ -4 & 10 & -4 \\ -6 & -2 & 12 \end{pmatrix}$$
   erfüllt das starke Zeilen- und das schwache Spaltensummenkriterium. Eins davon würde genügen. Der Algorithmus konvergiert.

2. $$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \\ 1 & 2 & 1 \end{pmatrix}$$
   erfüllt keins der Kriterien. Der Algorithmus konvergiert nicht.

3. $$\begin{pmatrix} 4 & 1 & 2 \\ 1 & 3 & 2 \\ 1 & 1 & 2 \end{pmatrix}$$
   erfüllt das schwache Zeilensummenkriterium aber kein Spaltensummenkriterium. Das Erste genügt. Der Algorithmus konvergiert.

### 2.2.4 Relaxation

Die konvergenzgeschwindigkeit kann oft wesentlich durch einen Relaxationparameter $\omega \in (0, 2)$ verbessert werden.

Es sei

$$\Delta x_i^{k+1} = x_i^{k+1} - x_i^k$$

Wir ersetzen den neu berechneten Wert $x_i^{k+1}$ durch eine Linearkombination dieses Wertes und den vorangegangenen $x_i^k$ gemäß

$$\tilde{x}_i^{k+1} = x_i^k + \omega\, \Delta x_i^{k+1} = x_i^k + \omega\, (x_i^{k+1} - x_i^k) = (1 - \omega)\, x_i^k + \omega\, x_i^{k+1}$$

| | |
|---|---|
| $\omega = 0$ | $\tilde{x}^{k+1} = x^k$ , stillstand, nicht sinnvoll! |
| $\omega = 1$ | $\tilde{x}^{k+1} = x^{k+1}$ , Jacobi/Gauß-Seidel |
| $\omega < 1$ | Dämpfung, Unterrelaxation |
| $\omega > 1$ | Überrelaxation |

**JOR (gedämpftes Jacobi-/Einzelschritt-Verfahren)**

Es sei ein startwert $x^0 \in \mathbb{R}^n$ gegeben. Berechnen Sie $x^k \in \mathbb{R}^n$ für $k = 1, \ldots$ gemäss:

fork=0,....
  fori=1,..,n
    $x_i^{(k+1)} = (1 - \omega)\, x_i^{(k)} - \frac{\omega}{a_{ii}} \left( y_i - \sum_{j=1, j \neq i} a_{ij} x_j^{(k)} \right)$

**SOR="Successive Over-Relaxation" (gedämpftes Gauß-Seidel-/Einzelschritt-Verfahren)**

Es sei ein startwert $x^0 \in \mathbb{R}^n$ gegeben. Berechnen Sie $x^k \in \mathbb{R}^n$ für $k = 1, \ldots$ gemäss:

fork=0,....
  fori=1,..,n
    $x_i^{(k+1)} = (1 - \omega)\, x_i^{(k)} - \frac{\omega}{a_{ii}} \left( y_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} + \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right)$

**Bemerkung 2.9.** *Für symmetrische, positiv definite Matrizen $A$ kann man einige nützliche Aussagen treffen:*

1. *Die Verfahren konvergieren für $\omega \in (0, 2)$.*
2. *Um die Konvergenz gegenüber dem Gauß-Seidel-Verfahren zu beschleunigen, verwendet man heuristische Werte zwischen 1,5 und 2,0. Die optimale Wahl hängt von der Koeffizientenmatrix $A$ ab. Werte $\omega < 1$ können gewählt werden, um eine Lösung zu stabilisieren, die ansonsten leicht divergiert.*
3. *Das Theorem von Kahan (1958) zeigt, dass für $\omega$ außerhalb des Intervalls $(0, 2)$ keine Konvergenz mehr vorliegt.*
4. *Ein optimales $\omega$ für das Gauß-Seidel-Verfahren erhalten wir durch*

$$\omega = \frac{2}{1 + \sqrt{1 - \varrho^2}},$$

*wobei $\varrho$ der Spektralradius der Iterationsmatrix der Jacobi-Iteration ist.*

# –3–
# Lösen nichtlinearer Gleichungen und Gleichungssysteme

## 3.1 Newton-Verfahren

Gesucht ist eine Stelle $x \in \mathbb{R}$, so dass zu einer gegebenen Funktion $f : \mathbb{R} \to \mathbb{R}$ gilt

$$f(x) = 0\,.$$

Da das Taylorpolynom $T_f := T_{f,1}(x, x_0)$ von $f$ eine Approximation an $f$ ist, ist die Nullstelle von $T_f$ ein Kandidat einer Approximation der Nullstelle von $f$.

$$\begin{aligned} & T_f = f(x_0) + f'(x_0)\,(x - x_0) = 0 \\ \Leftrightarrow \quad & f'(x_0)\,(x - x_0) = -f(x_0) \\ \Leftrightarrow \quad & x = x_0 - \frac{f(x_0)}{f'(x_0)} \end{aligned}$$

$x_0$ ist eine bel. Wahl des Entwicklungspunktes und $x$ die Nullstelle von $T_f$.

**Motivation der Fixpunktiteration**

$$x = x - \frac{f(x)}{f'(x)} \quad \Leftrightarrow \quad f(x) = 0$$

Iteration:

$$\begin{aligned} & \text{Starte mit } x_0 \\ & x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \end{aligned}$$

**Veranschaulichung**

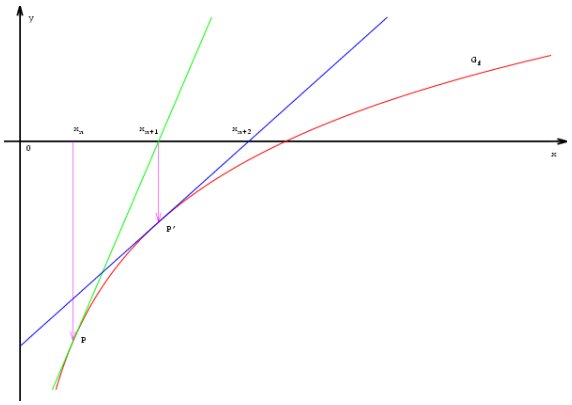
Starte bei $x_0 = 0$.

Lege eine Tangentengerade $T_{f,x_0}(x)$ am Punkt $(x_0, f(x_0))$ an:

$$T_{f,x_0}(x) = f'(x_0)\,(x - x_0) + f(x_0)$$

Der neue Punkt $x_1$ ist der Schnittpunkt der Tangente mit der $x$-Achse:

$$T_{f,x_0}(x_1) = 0 \quad \Leftrightarrow \quad x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$



usw.

Beispiel 14

$$f(x) = 1 - \mathrm{e}^x\,, \quad f'(x) = -\mathrm{e}^x$$

$$\begin{aligned}
x_0 &= 1 \\
x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)} = x_0 + \frac{1 - \mathrm{e}^{x_0}}{\mathrm{e}^{x_0}} = x_0 - 1 + \mathrm{e}^{-x_0} = \mathrm{e}^{-1} \approx 0.3679 \\
x_2 &= x_1 - 1 + \mathrm{e}^{-x_1} = \mathrm{e}^{-1} - 1 + \mathrm{e}^{-\mathrm{e}^{-1}} \approx 0.0601 \\
x_3 &= x_2 - 1 + \mathrm{e}^{-x_2} \approx 0.0018
\end{aligned}$$

usw.

function **(x k)=NewtonIteration(x0,f,df,TOL,ItMax):**

```
x = x0;
Res = abs(f(x));
k = 0;
while Res>TOL & k<ItMax

  x = x - f(x)/df(x);
  Res = abs(f(x));
  k = k+1;
  fprintf('x_%d = %.8f mit Res = %.2e\n',k,x,Res);

end
end
```

Der Aufruf erfolgt von einem Hauptprogramm das vielleicht - wer hätte das gedacht? - `NLGS/MyNewton.m` heißt. Der Inhalt könnte für Beispiel 14 ungefär so aussehen:

**NLGS/MyNewton.m:**

```
f = @(x) 1-exp(x);
df = @(x) -exp(x);

x0=1;
[x k] = NewtonIteration(x0,f,df,1.0e-08,1000);
```

und liefert die Ausgabe:

```
x_1 = 0.36787944 mit Res = 4.45e-01
x_2 = 0.06008007 mit Res = 6.19e-02
x_3 = 0.00176920 mit Res = 1.77e-03
x_4 = 0.00000156 mit Res = 1.56e-06
x_5 = 0.00000000 mit Res = 1.22e-12
```

**Problemsituationen**

- Es gibt keine Nullstelle
  - $f(x) = x^2 + 2, x^0 = 1$ divergiert.
- Es gibt mehr als eine Nullstelle
  - $f(x) = \cos x, x^0 = \frac{\pi}{4}$ konvergiert zu $\frac{\pi}{2}$,
    aber $x^0 = \operatorname{acot}(2\,\pi)$ divergiert (theoretisch), denn
    $x^1 = x^0 + 2\,\pi$ führt auf $x^{k+1} = x^0 + (k+1)\,2\,\pi \to \infty$.
    Numericshe Ungenauigkeiten führen dazu, dass die Folge irgendwann doch konvergiert. Der Algorithmus findet, dann eine Nullstelle, die vom Startwert sehr weit entfernt ist.
- Ein ungeünstig gewählter Startwert führt auf ein Divergenz.
  - $f(x) = x^3 - 2\,x + 2, x^0 = 0$ oszilliert

## 3.2 Konvergenz

**Satz 3.1** (Konvergenzordnung des Newton-Verfahrens). *Es sei $f \in C^2$ und $a \in \mathbb{R}$ eine Nullstelle von $f$ mit $f'(a) \neq 0$. Dann gibt es eine Umgebung $U(a)$, so dass das Newtonverfahren für jeden Startwert $x^0 \in U(a)$ quadratisch gegen die Nullstelle $a$ konvergiert, d.h. es gibt eine Konstante $C \in \mathbb{R}$ mit*

$$|x^{k+1} - a| \leq C\,|x^k - a|^2\,.$$

**Beweis Satz: 3.1:**

$$\begin{aligned}
|x^{k+1} - a| &= \left| x^k - \frac{f(x^k)}{f'(x^k)} - a \right| = \left| (x^k - a) - \frac{f(x^k) - f(a)}{f'(x^k)} \right|, \quad f(a) = 0! \\
&= \left| (x^k - a)\left(1 - \frac{1}{f'(x^k)}\,\frac{f(x^k) - f(a)}{x^k - a}\right) \right| \\
&= \left| (x^k - a)\left(1 - \frac{f'(\zeta)}{f'(x^k)}\right) \right|, \quad \zeta \in (x^k, a) \\
&= \left| (x^k - a)\frac{f'(x^k) - f'(\zeta)}{f'(x^k)} \right| = \left| (x^k - a)(x^k - \zeta)\frac{1}{f'(x^k)}\,\frac{f'(x^k) - f'(\zeta)}{x^k - \zeta} \right| \\
&\leq |x^k - a|^2 \left| \frac{f''(\eta)}{f'(x^k)} \right|, \quad \eta \in (x^k, \zeta) \\
&\leq \frac{M}{m}\,|x^k - a|^2 \qquad M := \max |f''(\eta)|\,,\ m := \min f'(x^k) \neq 0
\end{aligned}$$

□

**Bemerkung 3.2.** *Wenn das Verfahren konvergiert, dann quadratisch, aber konvergiert es überhaupt?*

$$\begin{aligned}
&|x^1 - a| \leq K\,|x^0 - a|^2\,, \quad K = \frac{M}{m} \\
&|x^2 - a| \leq K\,|x^1 - a|^2 \leq K^{1+2}|x^0 - a|^{2\cdot 2} \\
&|x^3 - a| \leq K^{1+2+2^2}|x^0 - a|^{2\cdot 3} \\
&\qquad \vdots \quad \text{(Geom. Reihe für die Potenz von } K\text{)} \\
&|x^n - a| \leq K^{2^n-1}|x^0 - a|^{2\cdot n}
\end{aligned}$$

Denn:

$$|x^n - a| \leq K^{2^n-1}|x^0 - a|^{2\cdot n}$$

Haben wir den Startwert $x^0$ nahe genug bei der Nullstelle gewählt, dann gilt schon mal

$$\lim_{n\to\infty} |x^0 - a|^{2\cdot n} = 0\,,$$

aber wir müssen noch sicherstellen, dass uns

$$\lim_{n\to\infty} K^{2^n-1} = ?$$

keinen Strich durch die Rechnung zieht.

Dazu muss einfach die Umgebung $U(a)$ so klein gewählt werden, dass $K < 1$ ist.

Nicht umsonst sagt man dazu, dass das Verfahren $\mathbf{lokal}$ quadratisch konvergiert!

## 3.3 EOC: Experimental Order of Convergence

Soll ein Verfahren von der Ordnung $p$ konvergieren - Gerüchten zufolge, so erwarten wir ja nach $k$, bzw. $k-1$ Iterationen folgende Situation anzutreffen:

$$E_{k+1} \sim E_k^p\,,\ \text{bzw.} E_k \sim E_{k-1}^p\,.$$

Dividieren beider Gleichungen liefert

$$\frac{E_{k+1}}{E_k} \sim \left(\frac{E_k}{E_{k-1}}\right)^p\,,$$

was äquivalent ist zu

$$p \sim \log_{\frac{E_k}{E_{k-1}}}\left(\frac{E_{k+1}}{E_k}\right) = \frac{\ln\left(\frac{E_{k+1}}{E_k}\right)}{\ln\left(\frac{E_k}{E_{k-1}}\right)}\,.$$

**Bemerkung 3.3.** *Bei der Berechnung der Konvergenzordnung p für den EOC sind wir stillschweigend davon ausgegangen, dass K tatsächlich konstant ist. Das ist natürlich nicht zwingend korrekt. In K stecken Maximum, bzw. Minimum von $f''$ bzw. $f'$ bezogen auf ein bestimmtes Intervall. Da sich das betrachtete Intervall aber in jedem Iterationsschritt ändert, wirkt sich das auch gerne auf K aus. Dadurch variiert der Wert für p und entspricht auch nicht exakt dem theoretischen Wert der Ordnung. Gehen wir von einer Konvergenz des Verfahrens aus und einem Moment, ab dem der Algorithmus sich tatsächlich in jedem Iterationsschritt seinem Grenzwert nähert, so variiert K immer weniger. Das erklärt, warum, im Falle einer Konvergenz, sich für p im Wesentlichen eine Tendenz zeigt und das ist auch der Grund, weshalb man für den EOC mehrere Fehler berechnet und in Betracht zieht.*

function **p=EOC(ERR):**

```
N = length(ERR);

for k=1:N-2
  p = log(ERR(k+2)/ERR(k+1))/log(ERR(k+1)/ERR(k));
  fprintf('Konvergenzordnung p(%d)=%.2e\n',k,p);
end

end
```

## 3.4 Newton-Verfahren für höherdimensionale Probleme

Gegeben sei eine Funktion $f : \mathbb{R}^2 \to \mathbb{R}^2$ der Form

$$f(x,y) = \begin{pmatrix} f^1(x,y) \\ f^2(x,y) \end{pmatrix}.$$

Gesucht ist das Punktepaar $(x^\star, y^\star)$ so dass

$$f(x^\star, y^\star) = \vec{0}$$

gilt.

Das Newton-Verfahren hat dann die Form:

$$\begin{pmatrix} x^{k+1} \\ y^{k+1} \end{pmatrix} = \begin{pmatrix} x^k \\ y^k \end{pmatrix} - Df^{-1}(x^k, y^k)\, f(x^k, y^k)$$

Dabei ist $Df(x,y)$ die Jacobi-Matrix von $f$, also

$$Df(x,y) = \begin{pmatrix} f_x^1(x,y) & f_y^1(x,y) \\ f_x^2(x,y) & f_y^2(x,y) \end{pmatrix} \in \mathbb{R}^{2\times 2}.$$

Und mit $Df^{-1}$ bezeichnen wir die Inverse der Jacobi-Matrix; für $2 \times 2$-Matrizen bedeutet das

$$Df^{-1}(x,y) = \frac{1}{\det Df(x,y)} \begin{pmatrix} f_y^2(x,y) & -f_y^1(x,y) \\ -f_x^2(x,y) & f_x^1(x,y) \end{pmatrix} \in \mathbb{R}^{2\times 2},$$

mit

$$\det Df(x,y) = f_x^1(x,y)\, f_y^2(x,y) - f_y^1(x,y)\, f_x^2(x,y).$$

Sei $f : \mathbb{R}^n \to \mathbb{R}$ mit einem großen $n$. Dann ist

$$Df(x) \in \mathbb{R}^{n\times n}.$$

Wir wollen in diesem Fall die Inverse nicht berechnen müssen, da dies in der Regel viel zu rechenaufwändig ist. Wir formulieren den Algorithmus derart, dass wir ein LGS zu lösen haben:

$$\begin{aligned} & x^{k+1} = x^k - Df^{-1}(x^k)\, f(x^k) \qquad x^k \in \mathbb{R}^n, Df^{-1} \in \mathbb{R}^{n\times n} \\ \Leftrightarrow \quad & x^{k+1} - x^k = -Df^{-1}(x^k)\, f(x^k) \\ \Leftrightarrow \quad & Df(x^k)(x^{k+1} - x^k) = -f(x^k) \end{aligned}$$

Der Algorithmus gestaltet sich nun folgendermaßen:

Es sei $x^0 \in \mathbb{R}^n$ ein gegebener Startwert. Berechne $x^{k+1}$ aus $x^k$ für $k = 0, \ldots$ gemäß

Löse das LGS nach $z$ auf

$$Df(x^k)\, z = -f(x^k)$$

und aktualisiere

$$x^{k+1} = z + x^k\,.$$

Das Lösen des entstandenen LGS können Sie zum Beispiel mit dem LR- oder SOR-Verfahren bewerkstelligen.

Beispiel 15

An welcher Stelle $(x, y)$ ist das notwendige Kriterium für ein Extremum der Funktion

$$f(x, y) = x^3 + y^3 - 3\,x\,y$$

erfüllt? Starten Sie den Algorithmus mit $(x^0, y^0) = (2, 2)$.

Hier ist jetzt nicht nach der Nullstelle von $f$ gefragt, sondern nach der Nullstelle der ersten Ableitung, also dem Gradienten:

$$\nabla f(x, y) = \begin{pmatrix} 3\,x^2 - 3\,y \\ 3\,y^2 - 3\,x \end{pmatrix}$$

Wenn Sie sich den Graphen $f$ einmal anschauen wollen:

```
[xx,yy] = meshgrid(0:0.1:3,1:0.1:2);
f = @(x,y) x.^3+y.^3-3*x.*y;
zz=f(xx,yy);
surfc(xx,yy,zz);
```

Newton-Verfahren: Startwert: $x^0$ gegeben.

Löse das LGS nach $z$ auf

$$Hf(x^k)\, z = -\nabla f(x^k)$$

und aktualisiere

$$x^{k+1} = z + x^k\,.$$

Es ist dann $Hf$ die Hessematrix von $f$:

$$Hf(x,y) = \begin{pmatrix} f_{xx}(x,y) & f_{xy}(x,y) \\ f_{yx}(x,y) & f_{yy}(x,y) \end{pmatrix}$$

Für unser Beispiel also:

$$Hf(x,y) = \begin{pmatrix} 6\,x & -3 \\ -3 & 6\,y \end{pmatrix}$$

Das Newtonverfahren ist dann:

$$\begin{pmatrix} 6\,x^k & -3 \\ -3 & 6\,y^k \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = -\begin{pmatrix} 3\,x^{k,2} - 3\,y^k \\ 3\,y^{k,2} - 3\,x^k \end{pmatrix}$$
$$\begin{pmatrix} x^{k+1} \\ y^{k+1} \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} + \begin{pmatrix} x^k \\ y^k \end{pmatrix}$$

```
function (x k) = NewtonSysLGS(x0,LGSSolv,TOL,ItMax,flag):

x = x0;
[f Df] = FuncFdF(x);

k=0;
while norm(f)>TOL & k<ItMax

  z = LGSSolv(Df,-f'); % im Notfall geht immer z=-(Df\f')';
  x = x+z;

  [f Df] = FuncFdF(x);
  k = k+1;
  if flag
    fprintf('Res.: %.2e nach %5d Iterationen\n',norm(f),k)
  end
end

end

function (f Df)=FuncFdF(x):

Df = [[6*x(1) -3];[-3 6*x(2)]];
f = x(1)^3+x(2)^3-3*x(1)*x(2);
end
```

Das Hauptprogramm `NLGS/MyNewton.m` für Beispiel 15 könnte so aussehen:

```
addpath('../LGS');

x0 = [2 2];
[x k] = NewtonSys(x0,@MyLR,1.0e-08,1000,1);

[xx,yy] = meshgrid(-2:0.1:2,0:0.1:3);
f = @(x,y) x.^3+y.^3-3*x.*y;
zz=f(xx,yy);
surfc(xx,yy,zz);
```

# –4–
# Regression

Stellen Sie sich vor, Sie wollen einen Prozess beschreiben, der ein beschränktes Wachstum darstellt. Sie wissen wohl, dass die beschreibende Funktion die Form

$$f(t) = S - (S - B_0)\,\mathrm{e}^{-k\,t}$$

besitzt, kennen aber die Parameter $S$, $B_0$ und $k$ nicht. Um diese Parameter nun für Ihren speziellen Prozess ermitteln zu können nehmen Sie eine ganze Reihe Messungen vor. Die Daten, die Sie in der Regel erhalten passen nicht genau auf den Kurvenverlauf. Es treten Störungen Rauschen auf. Was Sie nun brauchen, neben möglichst vielen guten Messungen, ist eine Methode, die Ihnen sagt, für welche Parameter, hier $S$, $B_0$ und $k$, Ihre Funktion $f$ **"am besten"** in die gemessene Punktewolke hineinpasst (siehe Abb. 6). Hat man zunächst einmal nur Messungen vorliegen ist eine weitere Frage oft, welche **"Art"** einer Funktion $f$ ist eigentlich gesucht.

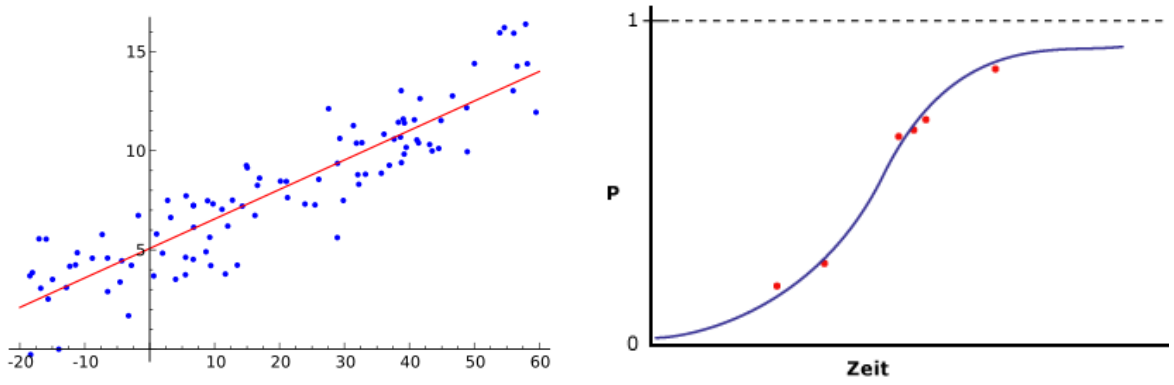



Abbildung 6: Regression durch Punktwolken

Es stellen sich uns drei wesentliche Fragen:

1. **Was heißt am "besten"?**

   Für eine Punktewolke

   $$\{P^1\,,\ P^2\,,\ldots\,,\ P^n\}\,,\ \text{mit}\ P^i = (P^i_x, P^i_y) \in \mathbb{R}^2$$

   bedeutet am besten, dass die Abweichung der Funkitonswerte $f(P_x)$ zu den entsprechenden Punktwerten $P_y$ minimal ist, das heißt, dass der Fehler $E$ mit

   $$E = \left(\sum_P |f(P_x) - P_y|^2\right)^{\frac{1}{2}} \tag{3}$$

minimal ist.

2. **Woher weiß ich welche "Art" die richtige ist?**

   Dabei ist nach der Sorte $f$ gefragt. Handelt es sich um eine Gerade

   $$f(x) = m\,x + b$$

   oder eine Exponentialfunktion, etwa der Form

   $$g(x) = a\,\mathrm{e}^{b\,x}$$

   oder ein Polynom vom Grad $4$

   $$p(x) = a_4\,x^4 + a_3\,x^3 + a_2\,x^2 + a_1\,x + a_0\,?$$

   Welche Funktiossorte, wir sagen dann *Ansatzfunktion* die Richtige ist, lässt sich nicht pauschal erklären. Das hängt jeweils vom Prozess ab, aus dem die Daten entstanden sind. Handelt es sich beispielsweise um eine Temperaturmessung oder ein Wachstumsprozess, ist wohl etwas im Bereich der Exponentialfunktionen gefragt, derer es ja nun auch wieder ganz verschiedene gibt.

3. **Wie berechne ich die gesuchte Funktion?**

   Damit beschäftigen wir uns ausführlich in den beiden folgenden Unterkapiteln.

   Jede Funktionsart enthält verschiedene und verschieden viele Parameter, die so zu bestimmen sind, dass der Fehler (3) minimal wird. Je nach gewählter Ansatzfunktion hängt der Fehler $E$ von den jeweilig aufkommenden Parametern ab. Zum Beispiel ist für $g(x)$ der Fehler von der Form $E:\ \mathbb{R}^2 \to \mathbb{R}$ mit

   $$E(a,b) = \left(\sum_P |g(P_x) - P_y|^2\right)^{\frac{1}{2}} \tag{4}$$

   und damit eine Funktion, die von den zwei Parametern $a$ und $b$ abhängt. Wir suchen einen Kandidaten eines Extremums von $E$, in dem wir zunächst mal nach Nullstellen der Ableitungen von $E$ Ausschau halten. Oftmals handelt es sich dann dabei um das Auflösen einer nicht linearen Gleichung oder eines nicht linearen Gliechungssystems, welches wir dann mit dem bereits bekannten Newton-Verfahren aus Kapitel 3.1 lösen können.

Bei der Überlegung des gesuchten Rechenwegs gibt es einen ganz wesentlichen Unterschied, wenn die Ansatzfunktion eine Gerade ist oder nicht. Wir betrachten den ersten Fall separat im folgenden Unterkapitel und anschließend gehen wir zum allgemeineinen Fall über.

## 4.1 lineare Ausgleichsrechnung

Wir beginnen also mit der Situation, in der klar ist, dass die beschreibende Funktion lineare Gestalt hat, also eine Gerade ist. Wir werden (Satz 4.1) sehen, dass die Berechnung so einer Geraden bestechnd simpel ist; zunächst aber wollen wir den Weg beschreiten, der durch intrinsisch getriebene Gedankengänge geebnet wird (Bsp. 16). Wir machen es dann erst einmal sehr umständlich und wiederholen dann später das Beispiel in einfacherer Weise (Bsp. 17). Der Sinn, erst einmal den Umweg zu gehen, besteht darin, dass wir dann bei beliebigen Ansatzfunktionen das grundlegende Konzept anhand einer einfach darstellbaren Funktion mit wenig Parametern, bereits kennengelernt haben.

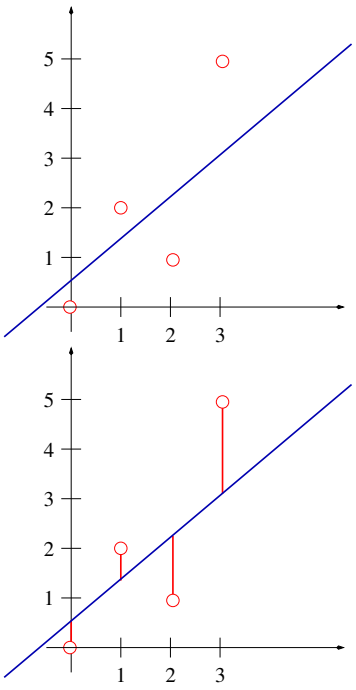
**Beispiel 16** Wir beginnen also exemplarisch mit einem Beispiel und gehen voneiner Punktewolke bestehend aus den vier Punkten

$$(0,0)\,,\ (1,2)\,,\ (2,1) \quad \text{und} \quad (3,5)$$

aus. Nun suchen wir die Gerade

$$f(x) = m\,x + b\,,$$

welche am besten durch die Punkte passt.




Wir beschreiben den Fehler über die Residuen

$$\mathrm{Res}_i = |f(P_x^i) - P_y^i|\,.$$

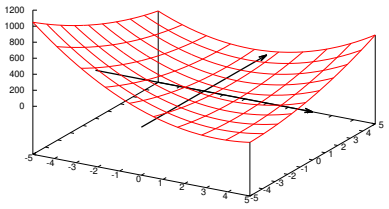
Dann ist der Quadratfehler

$$Q(m,b) = \sum_i \mathrm{Res}_i^2 \quad (= E^2(m,b))\,.$$

Wir wählen den Quadratfehler, da dessen Ableitungen leichter zu berechnen sind, und das Minimum von $E$ genau auch ein Minimum von $E^2$, bzw $Q$ ist. Im Beispiel:

$$\begin{aligned} f(0) &= b & \Rightarrow \quad \mathrm{Res}_1 &= b - 0 \\ f(1) &= m + b & \Rightarrow \quad \mathrm{Res}_2 &= m + b - 2 \\ f(2) &= 2\,m + b & \Rightarrow \quad \mathrm{Res}_3 &= 2\,m + b - 1 \\ f(3) &= 3\,m + b & \Rightarrow \quad \mathrm{Res}_4 &= 3\,m + b - 5 \end{aligned}$$

$$\begin{aligned} Q(m,b) = \sum_{i=1}^{4} \mathrm{Res}_i^2 &= b^2 + (m + b - 2)^2 + (2\,m + b - 1)^2 + (3\,m + b - 5)^2 \\ &= 4\,b^2 + 14\,m^2 + 12\,m\,b - 38\,m - 16\,b + 30 \end{aligned}$$




$Q(m,b)$

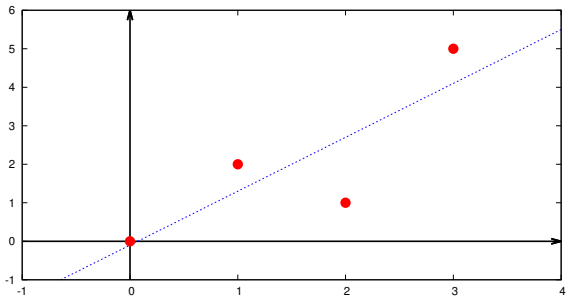
Minimum von $Q$ ist stationärer Punkt von $\binom{Q_m}{Q_b}$:

$$Q_m := \frac{d}{dm}Q(m,b)^2 = 28\,m + 12\,b - 38$$
$$Q_b := \frac{d}{db}Q(m,b)^2 = 12\,m + 8\,b - 16$$

$$\begin{pmatrix} 28 & 12 \\ 12 & 8 \end{pmatrix}\begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} 38 \\ 16 \end{pmatrix}$$

$$\Leftrightarrow \qquad \begin{pmatrix} m \\ b \end{pmatrix} = \frac{1}{5}\begin{pmatrix} 2 & -3 \\ -3 & 7 \end{pmatrix}\begin{pmatrix} 19 \\ 8 \end{pmatrix}\frac{1}{2}$$
$$= \frac{1}{10}\begin{pmatrix} 14 \\ -1 \end{pmatrix}$$

$$\Rightarrow \qquad f_{\text{best}}(x) = 1.4\,x - 0.1$$
$$Q(1.4, -0.1) \approx 2.05$$




Dieser Weg funktioniert immer sofern die Fehlerfunktion $Q$ auch tatsächlich ein Minimum hat und dieses eindeutig ist. Bei linearen Funktionen als Ansatzfunktion gibt es allerdings eine hilfreiche Eigenschaft, nämlich dass die gesuchten Funktione, die den Fehler minimiert, immer durch den Mittelwert aller Messpunkte verläuft. Diese Eigenschaft führt dazu, dass man in diesem Fall eine Formel zur Hand kriegt. Das macht die Situation extrem einfach, gibt es aber wirklich nur für Geraden als Ansatzfunktion. Es gilt der

**Satz 4.1.** *Es sei $S = (S_x, S_y) \in \mathbb{R}^2$ das arithmetische Mittel der Punktdaten $P = \{P_1, \ldots, P_N\}$ und $f = m\,x + b$ eine lineare Funktion, die den Quadratfehler $\|\mathrm{Res}(m,b)\|^2 = \|P_y - f_{m,b}(P_x)\|^2$ minimiert. Dann gilt*

$$f(S_x) = S_y\,.$$

**Beweis Satz 4.1:**

*Da die Funktion $f$ die Residuennorm bereits minimiert, gilt für den Quadratfehler $Q$*

$$Q_b(m,b) = 2\sum_{i=1}^{N}(P_y^i - (m\,P_x^i + b))(-1) = 0\,.$$

$$\begin{aligned}
\Rightarrow \qquad & \sum_{i=1}^{N} P_y^i - \sum_{i=1}^{N} m\,P_x^i - \sum_{i=1}^{N} b = 0 \\
\Leftrightarrow \qquad & \frac{1}{N}\sum_{i=1}^{N} P_y^i - m\,\frac{1}{N}\sum_{i=1}^{N} P_x^i - b = 0 \\
\Leftrightarrow \qquad & S_y - m\,S_x - b = 0 \\
\Leftrightarrow \qquad & S_y = m\,S_x + b \\
& \quad = f(S_x)
\end{aligned}$$

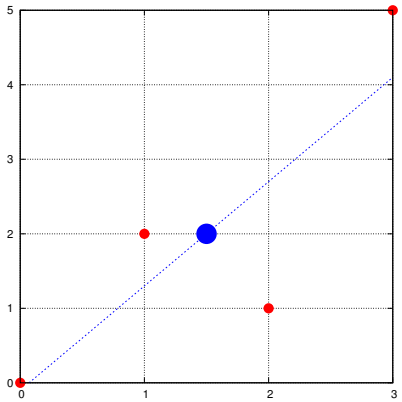
□

**Beispiel 17**

Im Beispiel 16 gilt am Schwerpunkt

$$S = (1.5\,,\ 2)$$

$$\begin{aligned}
\Rightarrow \qquad f(1.5) &= 1.4 \cdot 1.5 - 0.1 \\
&= \frac{7}{5}\cdot\frac{3}{2} - \frac{1}{10} \\
&= \frac{21}{10} - \frac{1}{10} = 2
\end{aligned}$$




Mit diesem Resultat können wir unser Problem auf ein eindimensionales Problem reduzieren:

Der Ansatz für die Gerade lautet jetzt:

$$y = m\,(x - S_x) + S_y$$

und für die Quadratsumme erhalten wir

$$Q(m) = \sum_{i=1}^{n}\left(P_y^i - m\,(P_x^i - S_x) - S_y\right)^2\,.$$

Dies ist eine quadratische Funktion mit der unabhängigen Variablen $m$. Gesucht ist die Scheitelstelle $Q'(m) = 0$.

$$Q(m) = \sum_{i=1}^{n} \left(P_y^i - m\,(P_x^i - S_x) - S_y\right)^2$$

$$\Rightarrow \qquad Q'(m) = 2\sum_{i=1}^{n} \left(P_y^i - m\,(P_x^i - S_x) - S_y\right)(-P_x^i + S_x)$$

$Q'(m) = 0 \;\Leftrightarrow$

$$\sum_{i=1}^{n} P_y^i\,(P_x^i - S_x) - m\,(P_x^i\,(P_x^i - S_x) - S_x\,(P_x^i - S_x)) - S_y\,(P_x^i - S_x) = 0$$

$$\Leftrightarrow \qquad < P_y, P_x > -m\,(\|P_x\|^2 - N\,S_x^2) - N\,S_y\,S_x = 0$$

$$\Leftrightarrow \qquad \frac{< P_y, P_x > -N\,S_y\,S_x}{\|P_x\|^2 - N\,S_x^2} = m$$

Insgesamt ergibt sich dann die bestechend simple Formel

$$f(x) = \frac{< P_y, P_x > -N\,S_y\,S_x}{\|P_x\|^2 - N\,S_x^2}\,(x - S_x) + S_y \tag{5}$$

**Beispiel 18** Wir berechnen die Funktion $f$ aus Beispiel 16 mit Hilfe der Formel (5) und vergleichen das Resultat mit dem ursprünglichen Ergebnis. Wenn wir alles richtig gemacht haben sollte das gleiche Ergebnis da stehen.

Es ist

$$S = (S_x, S_y) = (1.5, 2)$$

und

$$P_x = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \end{pmatrix} \quad \text{und} \quad P_y = \begin{pmatrix} 0 \\ 2 \\ 1 \\ 5 \end{pmatrix}.$$

Damit ist

$$< P_x, P_y >= 19\,,\ \|P_x\|^2 = 14$$

$$f(x) = \frac{< P_y, P_x > -N\,S_y\,S_x}{\|P_x\|^2 - N\,S_x^2}\,(x - S_x) + S_y$$

$$= \frac{19 - 4 \cdot 2 \cdot \frac{3}{2}}{14 - 4 \cdot \frac{9}{4}}\left(x - \frac{3}{2}\right) + 2 = \frac{7}{5}\left(x - \frac{3}{2}\right) + 2 = \frac{7}{5}\,x - \frac{7}{5} \cdot \frac{3}{2} + 2$$

$$= 1.4\,x - 0.1$$

## 4.2 nicht lineare Regression/Ausgleichsrechnung

Aus dem Modell, dem physikalischen Prozess, der zu Grunde liegt wissen wir vielleicht, dass die gesuchte Funktion eine Exponentialfunktion ist; vielleicht weil wir wissen, dass es sich bei den Messdaten um Bestandsaufnahmen eines wachsenden Prozesses handelt.

Welche Ansatzfunktion nun die passende ist hängt direkt vom Prozess ab, der die Punktedaten zu Tage gebracht hat.

Nehmen wir einmal an

$$g(x) = \alpha \, \mathrm{e}^{\beta \, x}$$

ist die passende Funktion, weil die Daten einem Wachstumsprozess entstammen. Dann gilt:

$$\begin{array}{lcl} g(0) = \alpha & \Rightarrow & \mathrm{Res}_1 = \alpha - 0 \\ g(1) = \alpha \, \mathrm{e}^{\beta} & \Rightarrow & \mathrm{Res}_2 = \alpha \, \mathrm{e}^{\beta} - 2 \\ g(2) = \alpha \, \mathrm{e}^{2\,\beta} & \Rightarrow & \mathrm{Res}_3 = \alpha \, \mathrm{e}^{2\,\beta} - 1 \\ g(3) = \alpha \, \mathrm{e}^{3\,\beta} & \Rightarrow & \mathrm{Res}_4 = \alpha \, \mathrm{e}^{3\,\beta} - 5 \end{array}$$

$\Rightarrow$

$$Q(\alpha, \beta) = \sum_{i=1}^{4} \mathrm{Res}_i^2$$

Das führt auf das Problem

$$\nabla \, Q(\alpha, \beta) = 0$$

$\Leftrightarrow$

$$\begin{pmatrix} \alpha + (\alpha \, \mathrm{e}^{\beta} - 2) \, \mathrm{e}^{\beta} + \mathrm{e}^{2\,\beta} \, (\alpha \, \mathrm{e}^{2\,\beta} - 1) + \mathrm{e}^{3\,\beta} \, (\alpha \, \mathrm{e}^{3\,\beta} - 5) \\ \alpha \, (\alpha \, \mathrm{e}^{\beta} - 2) \, \mathrm{e}^{\beta} + 2 \, \alpha \, \mathrm{e}^{2\,\beta} \, (\alpha \, \mathrm{e}^{2\,\beta} - 1) + 3 \, \alpha \, \mathrm{e}^{3\,\beta} \, (\alpha \, \mathrm{e}^{3\,\beta} - 5) \end{pmatrix} = \vec{0}$$

was nicht mehr in ein LGS überführt werden kann. Hier wird eine Nullstelle einer Funktion $\nabla \, Q \; : \; \mathbb{R}^2 \to \mathbb{R}^2$ gesucht, die wir - unter Umständen - mit dem Newton-Verfahren für höherdimensionale Probleme bestimmen können. Im Newton-Algorithmus werden wir dann wieder auf das Lösen eines LGS, nämlich

$$H \, Q \cdot y = -\nabla \, Q$$

stoßen. Dabei ist $H \, Q$ die Hessematrix von $Q$.

**MyNonLinReg.m**

```
f = @(x) c(1)*x.^2+c(2)*x;  % Ansatzfunktion für Bsp 16
P=[[0;1;2;3] [0;2;1;5]];

c0 = [0 0];
[c k] = NewtonSys(c0,@MyLR,1.0e-02,100,1);
```

In `NLGS/NewtonSys.m` dann die Subfunktion

**function (f Df)=FuncFdF(x)**

```
Df = [[196 72];[72 28]];
f = [196*x(1)+72*x(2)-102, 72*x(1)+28*x(2)-38];

end
```

Da man schnell auf sehr komplexe Situationen stößt wollen wir den Moment nutzen, und eine vorimplementierte Methode in Matlan kennenlernen, die beeindruckend einfache zu verwenden ist und genausogut funktioniert; und schnell!

Die MatLab-Funktion trägt den Namen

`lsqnonlin`

was für "least-square-nonlin", auf deutsch "Methode der kleinsten Quadrate", steht.

Beim Lesen des Hilfetextest mittels `>>help lsqnonlin` werden wir feststellen, dass sowohl die Argument- als auch die Ausgabeliste dieser Funktion verschiedene Formen haben kann. Das ist typisch für Matlab-Funktionen. Neben den obligatorischen Argumenten kann es auch optionale geben. Genauso kann man sich optionale Rückgabewerte ausgeben lassen.

obligatorisch ist hier

```
X = lsqnonlin(@FUN,X0)
```

die Routine, die das Residuum berechnet, hier `@FUN` und ein Startwert für die zu bestimmenden Koeffizienten, hier `X0`. Als Ergebnis erhalten wir die "besten", hier `X`.

Beispiel:

| | | |
|---|---|---|
| `X0` | Startwerte: | $c^0 = \binom{c^0(1)}{c^0(2)} = \binom{\alpha^0}{\beta^0} = \binom{0}{0}$ |
| `@FUN` | Zeiger auf Residuum: | $P_y - f(P_x)$ |
| in `FUN` | Ansatzfunktion $f$: | $f(x) = c(1)\,\mathrm{e}^{c(2)\,x}$ |
| nach $k$ Iterationen | Ergebnis `X`: | $c^k = \alpha^k, \beta^k$ |

**MyReg.m**

```
⋮
P=((0 0); (1 2); (2 1); (3 5));
c = lsqnonlin(@Residuum,(0 0));
```

**function y=Residuum(c)**

```
% Nur c als Argument erlaubt
% Weitere müssen global deklariert werden.
  global P
  y = zeros(1,length(P(:,1)));
  y = P(:,2)-Ansatzfunktion(P(:,1),c);
end
```

**function y=AnsatzFunktion(x,c)**

```
  y = c(1)*exp(c(2)*x); end
```

Lösung:

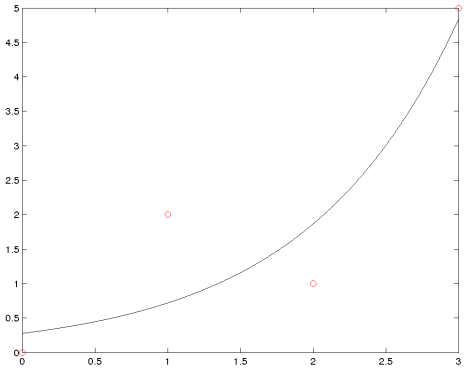
$$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}0.2766\\ 0.9541\end{pmatrix}$$

ergibt

$$f(x) = 0.2766\,\mathrm{e}^{0.9541\,x}$$

mit

$$\|\mathrm{Res}\|^2 = 2.4924\,.$$



# –5–
# Numerisch Integrieren

Bei der numerischen Integration geht es darum, ein bestimmtes Integral über eine Funktion $f$ auf einem Intervall $I = [a, b]$ zu berechnen, ohne mittels Hauptsatz der Differential- und Integralrechnung zuvor eine Stammfunktion bestimmen zu müssen. Vielleicht, weil es die Stammfunktion

$$F(x) = \int f(x)\, dx$$

in analytischer Form nicht gibt, oder der genaue Funktionsterm a-priori gar nicht bekannt ist. Vielleicht auch weil $f$ als Funktion $f : \mathbb{R} \to \mathbb{R}$ gar nicht gegeben ist, sonder nur als endliche Punktmenge $\{f(x_1), f(x_2), \ldots, f(x_N)\}$. Es gibt verschiedenste Gründe dafür. Wie auch immer erhalten wir in den meisten Fällen, aber nicht zwingend, nur eine Näherung an das bestimmte Integral.

Problemstellung:

Formeln zur (näherungsweisen) Berechnung bestimmter Integrale heißen Quadraturformeln und sind von der Form

$$\int_a^b f(s)\, ds \approx (b-a) \sum_{l=1}^{L} \omega_l\, f(s_l)\,.$$

Die Wahl der Stützstellen $s_l$ und Gewichte $\omega_l$ bestimmt die Güte der Quadraturformel.

## 5.1 Quadraturformeln

Der Quadratur zu Grunde liegt in der Regel die Idee, den Integranten $f$ durch eine Approximation zu ersetzen, etwa ein Taylorpolynom, dessen Stammfunktion man dann ja kennt und das führt dann auf eine handliche Formel. Der Fehler der Quadraturformel, aberdazu mehr später, ergibt sich dann aus dem Fehlre der Funktionsapproximation.

Schauen wir uns mal was ganz Einfaches an. Wir approximieren den Integranten $f$ durch ein Taylorpolynom $T_{f,0}(x, a)$ 0-ter Ordnung um den Entwicklungspunkt $a$, was gerade der linke Intervallrand ist:

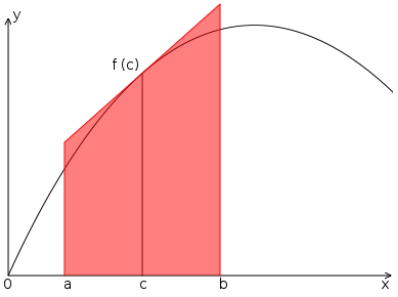
$$\begin{aligned}\int_a^b f(x)\,dx &\approx \int_a^b T_{f,0}(x,a)\,dx \\ &= \int_a^b f(a)\,dx \\ &= (b-a)\,f(a) \\ &= (b-a)\sum_{l=1}^{1} \underbrace{\omega_l}_{=1}\, f(\underbrace{s_l}_{=a})\end{aligned}$$

schon fertig; eine Stützstelle $s_1 = a$ und ein Gewicht $\omega_1 = 1$.

Wollen wir die Stützstelle in der Mitte des Intervalls $[a,b]$ also bei $\frac{1}{2}(a+b)$ ansetzen, so wählen wir genau diesen Punkt als Entwicklungspunkt des Taylorpolynoms vom Grad $0$. Wir erhalten dann die sogenannte

Tangenten-Trapez Formel

$$\int_a^b f(s)\,ds \approx (b-a) f\left(\frac{a+b}{2}\right)$$



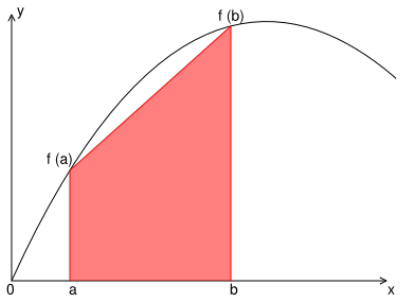

Die Approximation kann man verbessern, indem man im Taylorpolynom eine Ordnung höher geht:

$$
\begin{aligned}
\int_a^b f(x)\,dx &\approx \int_a^b T_{f,1}(x,a)\,dx \\
&= \int_a^b f(a) + \frac{1}{2} f'(a)\,(x-a)^2\,dx \\
&\approx (b-a)\,f(a) + \frac{1}{2}\,\frac{f(b)-f(a)}{b-a}\,(b-a)^2 \\
&= (b-a)\,f(a) + \frac{1}{2}\,(f(b)-f(a))\,(b-a) \\
&= (b-a)\,\left(\frac{1}{2} f(a) + \frac{1}{2} f(b)\right) \\
&= (b-a) \sum_{l=1}^{2} \underbrace{\omega_l}_{=\frac{1}{2}}\, f(\underbrace{s_l}_{\in\{a,b\}})
\end{aligned}
$$

Hier haben wir jetzt zwei Stützstellen, nämlich $s_1 = a$ und $s_2 = b$ mit jeweils $\omega_= \omega_2 = \frac{1}{2}$ als zugehöriges Gewicht. Diese Formel trägt den Namen

Sehnen-Trapez Formel :

$$\int_a^b f(s)\,ds \approx \frac{(b-a)}{2}\,(f(a)+f(b))$$



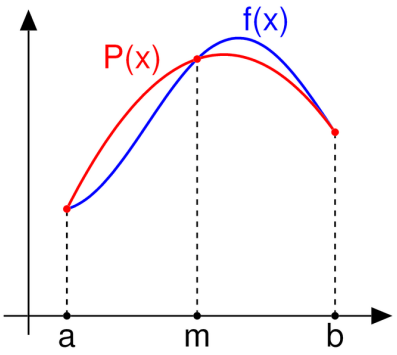

Eine weitere Idee beruht auf der Lagrange-Interpolation 2-ten Grades. Wir nehmen an, dass $f(a)$, $f\left(\frac{a+b}{2}\right)$ und $f(b)$ gegebene Werte sind und nähern $f$ durch das Polynom $p \in \mathbb{P}_2$ an, welches gerade die Bedingung

$$p(a) = f(a)\,, \quad p\left(\frac{a+b}{2}\right) = f\left(\frac{a+b}{2}\right) \text{ und } \quad p(b) = f(b)$$

erfüllt. Das Ergebins sieht so aus:

Simpsonregel (Keplersche Fassregel ):

$$\int_a^b f(s)\,ds \approx \frac{(b-a)}{6}\left(f(a)+4\,f\left(\frac{a+b}{2}\right)+f(b)\right)$$




Und die Rechnung dazu geht so: Wir setzen $p \in \mathbb{P}_2$ an durch

$$p(x) = p_2\,x^2 + p_1\,x + p_0 \;\in\; \mathbb{P}_2$$

mit $p(a) = f(a)$, $p\left(\frac{a+b}{2}\right) = f\left(\frac{a+b}{2}\right)$, $p(b) = f(b)$ und das führt dann auf das lineare Gleichungssystem

$$\begin{bmatrix} a^2 & a & 1 \\ \left(\frac{a}{2}+\frac{b}{2}\right)^2 & \frac{a}{2}+\frac{b}{2} & 1 \\ b^2 & b & 1 \end{bmatrix} \begin{bmatrix} p_2 \\ p_1 \\ p_0 \end{bmatrix} = \begin{bmatrix} f(a) \\ f\left(\frac{a+b}{2}\right) \\ f(b) \end{bmatrix}$$

LGS auflösen liefert

$$\begin{aligned} p(x) = \frac{b-a}{\det}\Bigg[&\left(f\left(\frac{a+b}{2}\right)-\frac{1}{2}\left(f(a)+f(b)\right)\right)x^2 \\ &+\left(\frac{(a+3\,b)}{4}f(a)-(a+b)\,f\left(\frac{a+b}{2}\right)+\frac{(3\,a+b)}{4}f(b)\right)x \\ &+\left(\frac{b\,(a+b)}{4}f(a)+b\,a\,f\left(\frac{a+b}{2}\right)+\frac{a\,(a+b)}{4}f(b)\right)\Bigg] \end{aligned}$$

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

Wendet man die Simpsonformel auf quadratische Polynome an, so ist sie exakt.

Formeln im Überblick

$$\int_a^b g(x)\,dx \approx (b-a)\,g(a)$$

**Sehnen-Trapez Formel**

$$\int_a^b g(x)\,dx \approx (b-a)\,\frac{g(a)+g(b)}{2}$$

**Tangenten-Trapez Formel**

$$\int_a^b g(x)\,dx \approx (b-a)\,g\left(\frac{a+b}{2}\right)$$

**Simpsonregel (Keplersche Fassregel)**

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

In der Praxis sieht das dann so aus, dass zunächst das Integrationsgebiet in Teilgebiete zerlegt wird durch eine nicht notwendigerweise - aber bei uns jetzt übliche - äquidistante Zerlegung

$$[c,d] = \bigcup_{k=1}^{N-1} [x_k, x_{k+1}]\,.$$

Gemäß dieser Zerlegung zerlegen wir auch das bestimmte Integral über $[c,d]$:

$$\int_c^d f(x)\,dx = \sum_{k=1}^{N-1} \int_{x_k}^{x_{k+1}} f(x)\,dx$$

Die Teilintegrale $\int_{x_k}^{x_{k+1}} f(x)\,dx$ ersetzen wir dann durch Quadraturformeln über $[a,b]$ wie wir sie oben gesammelt haben.

## 5.2 Fehlerrechnung: EOC = Experimental Order of Convergence

Es sei $\mathcal{E}(h)$ der Fehler einer Approximation bezüglich einer verwendeten Gitterweite $h$. Wir erwarten ein Konvergenzverhalten der Art

$$\mathcal{E}(h) = C\,h^{\alpha}$$

mit einer Konstanten $C$, die nicht von der Gitterweite $h$ abhängt. Meist hängt diese Konstante von der Gebietsgröße und Daten des Problems ab. Ist für uns aber im Moment nicht relevant. Wir interessieren uns für den Wert von $\alpha$; der gibt nämlich an, von welcher Ordnung ein Verfahren ist. Um den Wert von $\alpha$ zu ermitteln gibt es einen analytischen Weg (a-priori Fehlerrechnung) oder den experimentellen (Experimental Order of Convergence). Idealerweise stehen beide Wege zur Verfügun, denn dann lässt sich das Programm damit verifizieren.

Nehmen wir zwei Fehlerwerte $\mathcal{E}(h_k), \mathcal{E}(h_l)$, entsprechend zu zwei verschiedenen Gitter- oder Schrittweiten $h_k$, $h_l$. Dann gilt:

$$\begin{aligned}
& & \mathcal{E}(h_k) &= C\,h_k^{\alpha} \\
\wedge & & \mathcal{E}(h_l) &= C\,h_l^{\alpha} \\
\Rightarrow & & \frac{\mathcal{E}(h_l)}{\mathcal{E}(h_k)} &= \left(\frac{h_l}{h_k}\right)^{\alpha} \\
\Leftrightarrow & & \log_{\left(\frac{h_l}{h_k}\right)}\left(\frac{\mathcal{E}(h_l)}{\mathcal{E}(h_k)}\right) &= \alpha
\end{aligned}$$

**Formel 1: EOC:** Es seien $\mathcal{E}(h_k)$ und $\mathcal{E}(h_l)$ berechnete Fehler zu verschiedenen Gitterweiten $h_k \neq h_l$. Wir erhalten experimentel, die Fehlerordnung $\alpha \in \mathbb{R}^+$ mit

$$\mathcal{E}(h) = C\,h^{\alpha}$$

durch

$$\alpha = \frac{\ln\left(\frac{\mathcal{E}(h_l)}{\mathcal{E}(h_k)}\right)}{\ln\left(\frac{h_l}{h_k}\right)}\,.$$

# –6–
# Runge-Kutta Verfahren

Eine gewöhnliche Differentialgleichung (kurz GDGL oder ODE aus dem Englischen "ordinary differential equation") ist eine Differentialgleichung, bei der zu einer gesuchten Funktion nur Ableitungen nach genau einer Variablen auftreten. Viele Vorgänge in der Natur lassen sich mit solchen Gleichungen mathematisch beschreiben, z. B. der radioaktive Zerfall, Bewegungsvorgänge von Körpern, viele Arten von Schwingungsvorgängen oder das Wachstumsverhalten von Populationen.

**Definition 6.1** (gewöhnliche Differentialgleichung)**.** *Seien* $\Omega \subseteq \mathbb{R} \times (\mathbb{R}^m)^{n+1}, n \in \mathbb{N}$ *und* $f\colon \Omega \to \mathbb{R}^m$ *eine stetige Funktion. Dann heißt*

$$f\left(x, y, y', y'', \ldots, y^{(n)}\right) = 0$$

*ein gewöhnliches Differentialgleichungssystem* $n$*-ter Ordnung von* $m$ *Gleichungen. Im Fall* $m = 1$ *nennt man dies eine gewöhnliche Differentialgleichung* $n$*-ter Ordnung.*

Das wissenschaftliche Interesse an Differentialgleichungen ist im Wesentlichen darin begründet, dass mit ihnen auf Grund vergleichsweise einfacher Beobachtungen und Experimente vollständige Modelle geschaffen werden können. Nur wenige Typen von Differentialgleichungen lassen sich analytisch lösen[1]. In solchen Fällen muss dann auf numerische Lösungsmethoden zurückgegriffen werden. Die Lösungsmethode in diesem Kapitel, das sogenannte *Runge-Kutta-Verfahren* ist nur eine von verschiedenen Möglichkeiten und beinhaltet selbst Varianten.

Wir werden uns ausschließlich mit Differentialgleichungen 1-ter Ordnung befassen:

Probelmstellung:

Gesucht ist die Funktion $y : [0, T] \to \mathbb{R}^m$, die zu gegebenem Startwert $y_0 \in \mathbb{R}^m$ und gegebener rechten Seite $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$ die Gleichungen

$$y'(t) = f(t, y(t))$$

erfüllt.

## 6.1 Motivation

Wie kann so ein System von ODEs zustande kommen? Ein Beispiel: Wir wollen uns mit der Dynamik des Liebesverhältnisses zwischen Romeo und Julia befassen. Dabei bezeichnen wir

[1] frei nach Wikipedia

mit $R(t)$ die Liebe, die Romeo gegenüber Julia empfindet, und mit $J(t)$ die Liebe, die Julia zu Romeo hegt. Die Zeit $t$ messen wir in Tagen und es ist $t_0 = 0$. Aufmerksame Beobachter haben folgende Entwicklung festgehalten:

Von Anfang an (also ab $t = 0$) liebt Romeo seine Julia sehr:

$$R(0) = 4$$

Allerdings ist Julia dem Romeo anfangs eher neutral eingestellt:

$$J(0) = 0$$

Auch in der Entwicklung ihrer Liebe zueinander unterscheiden sie sich vom Typ her. Für Romeo ist die Sache ganz einfach: Je mehr Julia ihn liebt, desto mehr liebt er auch sie (und natürlich umgekehrt: je weniger Julia ihn liebt, desto weniger liebt er sie). Präziser:

$$R'(t) = \frac{4}{5}\, J(t) \tag{6}$$

Julias Gefühle lassen sich jedoch nicht so einfach beschreiben: Ihre Liebe zu Romeo lässt sofort nach, wenn Romeo beginnt, sie mehr zu lieben. Falls sich jedoch Romeos Gefühle abkühlen, dann fängt sie sofort an, ihn mehr zu lieben. Und zu guter Letzt wächst ihre Liebe zu ihm, je mehr sie ihn liebt. Präziser:

$$J'(t) = -\frac{1}{5}R(t) + \frac{2}{5}J(t). \tag{7}$$

Schon ganz erschöpft von dem ganzen Durcheinander, stellt sich Julia ein Ultimatum: Wenn Romeo sie am 50. Tag nicht liebt (d.h. falls $J(50) \leq 0$), trennt sie sich von ihm. Anderenfalls will sie mit ihm bis ans Ende ihrer Tage zusammen Bleiben Romeo und Julia zusammen? Die Antwort auf diese Frage werden wir erst am Ende des Kapitels klären.

Wir wollen zunächst die Schreibweise der Liebessitutionen in einen Ausdruck der Form $y' = f(t, y)$ bringen, damit wir den Zusammenhang zur Definition 6.1 wiedererkennen. Es ist

$$y(t) = \begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} = \begin{pmatrix} R(t) \\ J(t) \end{pmatrix} \quad \text{und} \quad y_0 = \begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} = \begin{pmatrix} 4 \\ 0 \end{pmatrix}$$

und

$$f(t, y) = \begin{pmatrix} f_1(t) \\ f_2(t) \end{pmatrix} = \begin{pmatrix} \frac{4}{5}\, y_2(t) \\ -\frac{1}{5}\, y_1(t) + \frac{2}{5}\, y_2(t) \end{pmatrix} = \begin{pmatrix} 0 & \frac{4}{5} \\ -\frac{1}{5} & \frac{2}{5} \end{pmatrix} y(t)\,.$$

Hinter dieser "Bewegungsform" steckt ein stark vereinfachtes Räuber-Beute-Modell. Wenn etwa $J$ einen Hasenbestand beschreibt und $R$ den Fuchsbestand, dann könnte man die Gleichungen (6) und (7) auch so verstehen: Der Hasenbestand fördert das Wachstum der Füchse, da sie als Nahrung dienen, der Hasenbestand hingegen wird durch den Fuchsbestand dezimiert, was aber wiederum dadurch kompensiert wird, dass die Hasen sich schnell vermehren.

Natürlich sind interessantere Situationen wesentlich komplexer in ihrer Darstellung aber wir wollen den umgekehrten Weg gehen und die Situation so weit möglich vereinfachen, um uns zunächst die Grundidee des numerischen Algorithmus klar zu machen.

## 6.2 Der Algorithmus

Wir betrachten eine ODE 1-ter Ordnung bestehend aus einer Gleichung der Form $y'(t) = f(t, y(t))$ auf $[0, T]$. Das Intervall $[0, T]$, auf dem die Lösung gesucht ist zerlegen wir in äquidistante Teilintervalle

$$[0, T] = \bigcup_{n=0}^{N-1} [x_n, x_{n+1}] \quad \text{mit} \quad x_0 = 0 \, x_N = T \, .$$

Die Gleichung

$$y' = f(t, y)$$

integrieren wir auf dem Intervall $[x_n, x_{n+1}]$ und erhalten

$$\int_{t_n}^{t_{n+1}} y' \, dt = \int_{t_n}^{t_{n+1}} f(t, y) \, dt$$

$$\Leftrightarrow \qquad y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y) \, dt \, .$$

Das Integral auf der rechten Seite approximieren wir durch eine Quadraturformel, die wir kurz $Q(f)$ nennen, so lange wir nicht wissen, was wir dafür einsetzen wollen. Es ist dann weiter

$$y(t_{n+1}) \approx y(t_n) + Q(f) \, .$$

Da bis auf $n = 0$ alle berechneten $y(t_n)$ nur Approximationen sind ersetzen wir die Ausdrücke jeweils durch $y_n \approx y(t_n)$ und schreiben die Iteration entsprechend um in

$$y_{n+1} = y_n + Q(f) \, .$$

Im Grunde genommen war's das schon. Es ist jetzt nur noch die Frage, welche Quadraturformel $Q(f)$ wir einsetzen wollen. Betrachten wir den simpelsten Fall, den wir mit der Quadratur

$$\int_{t_n}^{t_{n+1}} f(t, y) \, dt = (t_{n+1} - t_n) \, f(t_n, y(t_n))$$

erhalten, was auf das sogenannte explizite Euler-Verfahren führt:

*explizites Euler Verfahren:* $y_0$ gegeben und $h := t_{n+1} - t_n$:

$$y_{n+1} = y_n + h\, f(t_n, y_n)$$

Beispiel 19 explizites Euler

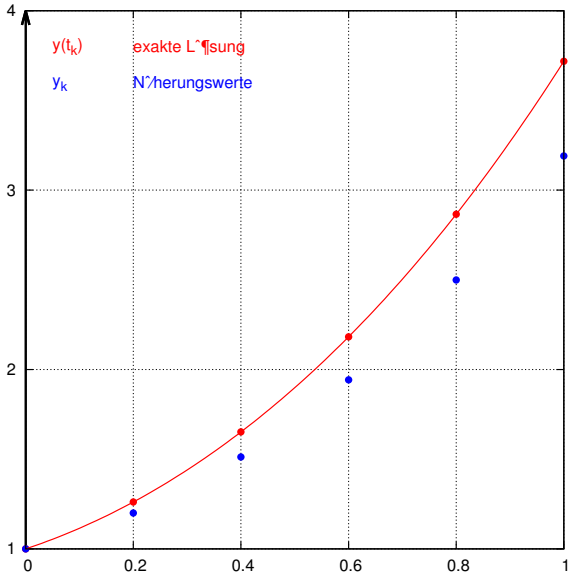
$$\begin{aligned} y'(t) &= y(t) - t(t-2) && = f(t, y(t)) \\ y(0) &= 1 && = y_0 \end{aligned}$$

Unsere Formel lautet demnach:

$$y_0 = 1 \quad y_{n+1} = y_n + (t_{n+1} - t_n)\,(y_n - t_n(t_n - 2)) \quad n = 0, 1, 2, \ldots$$

Es sei $h = 0.2$

$$\begin{aligned} y_1 &= y_0 + h\ (y_0 - 0\,(0-2)) \\ &= 1 + 0.2\ (1 - 0\,(0-2)) \\ &= 1.2 \\ y_2 &= y_1 + h\, f(0.2, y_1) = 1.512 \\ y_3 &= y_2 + h\, f(0.4, y_2) = 1.9424 \\ y_4 &= y_3 + h\, f(0.6, y_3) = 2.49888 \\ y_5 &= y_4 + h\, f(0.8, y_4) = 3.190656 \end{aligned}$$




Die exakte Lösung ist $y(t) = e^t + t^2$.

Wählen wir eine andere Quadraturformel, so erhalten wir auch ein anderes Runge-Kutta Verfahren; jetzt mit der Sehnen-Trapez Formel:

$$Q(f) = (t_{n+1} - t_n)\frac{f(t_n, y_n) + f(t_{n+1}, y_{n+1})}{2}$$

Das liefert zunächst mal den Algorithmus

$$\begin{aligned} y_{n+1} &= y_n + Q(f) \\ &= y_n + h\,\frac{f(t_n, y_n) + f(t_{n+1}, y_{n+1})}{2} \end{aligned}$$

Wir entledigen uns der $y_{n+1}$-Abhängigkeit auf der rechten Seite, indem wir hier ein Mal das explizite Euler Verfahren anwenden:

$$\approx y_n + h\,\frac{f(t_n, y_n) + f(t_{n+1}, \tilde{y}_{n+1})}{2}$$

mit

$$\tilde{y}_{n+1} = y_n + h\,f(t_n, y_n)$$

führt auf das

*Euler-Cauchy Verfahren:* $h := t_{n+1} - t_n$

$$y_{n+1} = y_n + \frac{h}{2}\,f(t_n, y_n) + \frac{h}{2}\,f(t_{n+1}, y_n + h\,f(t_n, y_n))$$

Hier einmal einige Ausdrücke zu verschieden gewählten Quadraturformeln:

$$y_{n+1} = y_n + Q(f) = y_n + h\sum_{l=1}^{L} \omega_l\,f(t_l, y(t_l))$$

$$= y_n + h \begin{cases} 1\,f(t_n, y_n) & \text{Euler} \\ \frac{1}{2}\,f(t_n, y_n) + \frac{1}{2}\,f(t_{n+1}, y_{n+1}) & \text{Sehnen-Trapez} \\ 1\,f(t_n + \frac{1}{2}\,h, y_{n+\frac{1}{2}\,h}) & \text{Tangenten-Trapez} \\ \frac{1}{6}\,f(t_n, y_n) + \frac{2}{3}\,f(t_n + \frac{1}{2}\,h, y_{n+\frac{1}{2}\,h}) + \frac{1}{6}\,f(t_{n+1}, y_{n+1}) & \text{Simpson} \end{cases}$$

Die impliziten Ausdrücke jeweils auf den rechten Seiten muss man dann noch durch ein jeweils passendes Verfahren ersetzen.

*Explizite, m-stufige Runge-Kutta Verfahren:*

$$y_{n+1} = y_n + h \sum_{l=1}^{L} \omega_l \, k_l(t_n, y_n)$$

$$k_l(t, y) = f(t + \delta_l \, h, y + h \sum_{j=1}^{l-1} \gamma_{lj} \, k_j(t, y))$$

$$\omega = \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_L \end{pmatrix}, \; \Delta = \begin{pmatrix} \delta_1 \\ \vdots \\ \delta_L \end{pmatrix}, \; \Gamma = \begin{pmatrix} \gamma_{11} & \cdots & \gamma_{1L} \\ \vdots & \ddots & \vdots \\ \gamma_{L1} & \cdots & \gamma_{LL} \end{pmatrix}, \; \gamma_{ij} = 0 \text{ falls } j \geq i$$

Butcher Tabelle :

$$\begin{array}{c|c} \Delta & \Gamma \\ \hline & \Omega^T \end{array} \quad = \quad \begin{array}{c|ccc} \delta_1 & 0 & \cdots & 0 \\ \vdots & \gamma_{21} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \delta_L & \gamma_{L1} & \cdots & 0 \\ \hline & \omega_1 & \cdots & \omega_L \end{array}$$

$$\begin{aligned} k_1(t, y) &= f(t + \delta_1 \, h, y) \\ k_2(t, y) &= f(t + \delta_2 \, h, y + h \, \gamma_{21} \, k_1(t, y)) \\ k_3(t, y) &= f(t + \delta_3 \, h, y + h \, \gamma_{31} \, k_1(t, y) + h \, \gamma_{32} \, k_2(t, y)) \\ &\vdots \end{aligned}$$

**Beispiel 20** Euler-Cauchy

$$y_{n+1} = y_n + \frac{1}{2} h \, f(t_n, y_n) + \frac{1}{2} h \, f(t_n + h, y_n + h \, f(t_n, y_n))$$

Wir sehen direkt, dass $L = 2$.

$$y_{n+1} = y_n + \frac{1}{2} h \underbrace{f(t_n + 0 \cdot h, y_n)}_{=k_1} + \frac{1}{2} h \underbrace{f(t_n + 1 \cdot h, y_n + 1 \cdot h \cdot k_1)}_{=k_2}$$

Jetzt können die Werte für die Butcher-Tabelle ablesen:

$$\omega = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}, \; \Delta = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \; \Gamma = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$$

**Beispiel 21** Butcher-Tabellen

| | |
|---|---|
| $0$ | |
| | $1$ |

Euler
(Ordnung 1)

| | | |
|---|---|---|
| $0$ | | |
| $1$ | $1$ | |
| | $\frac{1}{2}$ | $\frac{1}{2}$ |

Euler-Cauchy
(Ordnung 2)

| | | |
|---|---|---|
| $0$ | | |
| $\frac{1}{2}$ | $\frac{1}{2}$ | |
| | $0$ | $1$ |

Runge-Kutta
(Ordnung 2)

| | | |
|---|---|---|
| $0$ | | |
| $1$ | $\frac{1}{2}$ | $\frac{1}{2}$ |
| | $\frac{1}{2}$ | $\frac{1}{2}$ |

impl. Trapez
(Ordnung 2)

| | | | |
|---|---|---|---|
| $0$ | | | |
| $\frac{1}{2}$ | $\frac{1}{2}$ | | |
| $1$ | -1 | 2 | |
| | $\frac{1}{6}$ | $\frac{4}{6}$ | $\frac{1}{6}$ |

Runge-Kutta
(Ordnung 3)

| | | | |
|---|---|---|---|
| $0$ | | | |
| $\frac{1}{3}$ | $\frac{1}{3}$ | | |
| $\frac{2}{3}$ | $0$ | $\frac{2}{3}$ | |
| | $\frac{1}{4}$ | $0$ | $\frac{3}{4}$ |

Heun
(Ordnung 3)

| | | | | |
|---|---|---|---|---|
| $0$ | | | | |
| $\frac{1}{2}$ | $\frac{1}{2}$ | | | |
| $\frac{1}{2}$ | $0$ | $\frac{1}{2}$ | | |
| $1$ | $0$ | $0$ | $1$ | |
| | $\frac{1}{6}$ | $\frac{1}{3}$ | $\frac{1}{3}$ | $\frac{1}{6}$ |

kl. Runge-Kutta
(Ordnung 4)

Die angegebenen Ordnungszahlen stehen für die Konvergenzordnungen, der jeweiligen Verfahren. Wir werden das Thema Konvergenzordnung der Runge-Kutta Verfahren im folgenden Kapitel, Kapitel 6.3 behandeln.

**function (y t)=RungeKutta(f,y0,t0,tend,h,butchertabelle):**

```
y(:,1) = y0;
t(1) = t0;

Q=load(butchertabelle);

[M N]=size(Q);

Delta=Q(:,1);
Gamma=Q(:,2:N-1);
Omega=Q(:,N);

n=1;
while t(n)<tend
% Erst mal k1 und k2

  for l=1:M
    % jetzt k(l) berechnen
    sum=zeros(length(y0),1);
    for j=1:l-1
      sum = sum + Gamma(l,j)*k(:,j);
    end
    k(:,l) = f(t(n)+h*Delta(l),y(:,n)+h*sum);
  end

% neues y ist y1
  sum=zeros(length(y0),1);
  for l=1:M
    sum = sum + Omega(l)*k(:,l);
  end
  y(:,n+1) = y(:,n) + h*sum;
  t(n+1) = t(n)+h;
  n = n+1;
end

end
```

Aus einem Hauptprogramm aufgerufen könnte das Ganze dann so aussehen:

**MyRungeKutta.m:**

```
h = 0.25;
y0 = 1;
tstart = 1;
tend = 5;

f = @(t,y) (exp(t)-y)./t; yex = @(t) ((y0-exp(1))+exp(t))./t;

butchertabelle = 'EulerCauchy.but';

[y t] = RungeKutta(f,y0,tstart,tend,h,butchertabelle);

ye = yex(t);

plot(t,y,'r-',t,ye,'k')
title('y (rot) und yex (schwarz)');
```

## 6.3 Fehlerrechnung

Zur Lösung des Anfangswertproblems (AWP)

$$\begin{aligned} y_0 &= y(t_0) \\ y' &= f(t,y) \end{aligned} \qquad (\mathrm{A})$$

schreiben wir das Verfahren allgemein in der Form

$$\begin{aligned} y_0 &= y(t_0) \\ y_{n+1} &= y_n + h\, f_h(t_n, y_n) \end{aligned} \qquad (\mathrm{A}_h)$$

$f_h$ nennen wir die Verfahrensfunktion.

Beispiel 22 Euler explizit

$$f_h(t_n, y_n) = f(t_n, y_n)$$

Beispiel 23 Euler-Cauchy

$$f_h(t_n, y_n) = \frac{1}{2} f(t_n, y_n) + \frac{1}{2} f(t_n + h, y_n + h\, f(t_n, y_n))$$

**Definition 6.2** (lokaler Diskretisierungsfehler ).

$$\tau_h(t_j) = \frac{y(t_j + h) - y(t_j)}{h} - f_h(t_j, y(t_j))$$

$h\,\tau_j(t_j)$ gibt an wie groß der Fehler nach einem Schritt des Verfahrens ist. Angenommen $y_j = y(t_j)$ dann erhalten wir

$$h\,\tau_h(t_j) = y(t_j + h) - (y_j + h\, f_h(t_j, y_j)) = y(t_j + h) - y_{j+1}$$

und das ist gerade die Verfälschung beim Zeitschritt $t_j \to t_{j+1}$.

**Definition 6.3** (Konsistenz ). *Das Verfahren* $(\mathrm{A}_h)$*heißt konsistent mit* (A)*, wenn*

$$\max_{t_j} \|\tau_h(t_j)\| \to 0 \quad \text{für} \quad h \to 0\,.$$

**Definition 6.4** (Konsitenzordnung ). *Das Verfahren* $(\mathrm{A}_h)$ *hat die Konsitenzordnung* $p$*, falls es eine Konstante* $C$ *gibt, mit*

$$\max_{t_j} \|\tau_h(t_j)\| \le C\, h^p\,.$$

**Definition 6.5** (globaler Diskretisierungsfehler ).

$$e_h(t_j) := y_j - y(t_j)$$

**Definition 6.6** (Konvergenz ). *Das Verfahren* $(\mathrm{A}_h)$ *heißt konvergent, falls*

$$\max_{t_j} \|e_h(t_j)\| \to 0 \quad \text{für} \quad h \to 0$$

und es besitzt die *Konvergenzordnung* $p$, falls es eine Konstante $C$ gibt mit

$$\max_{t_j} \|e_h(t_j)\| \leq C\, h^p\,.$$

Die Konvergenzordnung ist über den globalen und die Konsistenzordnung über den lokalen Diskretisierungsfehler definiert. Bei Konvergenz stimmen die beiden Ordnungen überein. Wie man diese Ordnung experimentel bestimmen kann ist in Kapitel 5.2 beschrieben.

**MyRungeKutta.m:**

```
h = 0.25;
y0 = 1;
tstart = 1;
tend = 5;

% exakte Loesung
f = @(t,y) (-y+2.71828.^t)./t;
yex = @(t) (y0-exp(1)+exp(t))./t;

butchertabelle = 'EulerCauchy.but';
for loop=1:4
  [y t] = RungeKutta(f,y0,tstart,tend,h,butchertabelle);

  ye = yex(t);
  Err(loop) = norm(y-ye,inf);
  Step(loop) = h;
  fprintf('%.2e %.2e\',h,Err(loop));
  h = h/2;
end

EOCh(Err,Step);

plot(t,y,'r-',t,ye,'k')
title('y (rot) und yex (schwarz)');
```

# –7–
# Finite-Differenzen Methode

Im vorherigen Kapitel hatten wir uns mit Anfangswertproblemen erster Ordnung beschäftigt, auch gekoppelte Systeme, und brauchten dabei Kenntnisse zum numerischen Integrieren. In diesem Kapitel und auch im folgenden Kapitel 8 werden wir uns Randwertproblemen zweiter Ordnung widmen. Diese haben in etwa die Form

Probelmstellung:

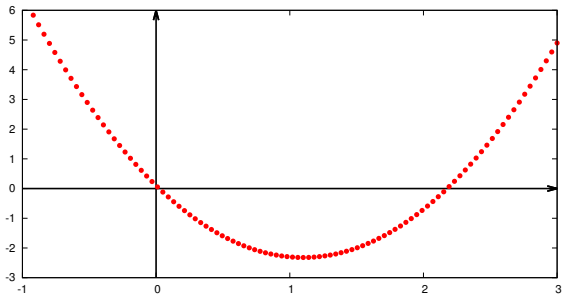
Zu gegebenen Zahlen $\alpha$ , $\beta \in \mathbb{R}$ und Gebiet $\Omega \subset \mathbb{R}^2$ mit $\partial\Omega = \Gamma_D \cup \Gamma_N$ ist die Funktion $u : \Omega \to \mathbb{R}$ gesucht mit

$$\begin{aligned} \alpha\, u - \beta\, \Delta u &= F \quad && \text{in } \Omega\,, \\ u &= u_D && \text{auf } \Gamma_D\,, \\ \nabla u \cdot \nu &= u_N && \text{auf } \Gamma_N\,. \end{aligned}$$

In diesem Zusammenhang werden wir uns im Unterkapitel 7.1 zunächst mit dem numerischen Differenzieren auseinandersetzen. In Kapitel 7.2 erlernen wir das sogenannte Finite-Differenzen Verfahren, welches wir an einem eindimensionalen Problem aufstellen. Für höherdimensionale Probleme verwenden wir Routinen mit dem Matlab-pdetool.

## 7.1 Numerich Differenzieren

Wenn wir die Ableitung einer Funktion benötigen und uns mitten in einer numerischen Berechnung befinden, dann kann es sein, dass wir keine analytische Funktion vorliegen haben sondern nur Werte einer Funktion an ausgewählten Stützstelllen:




Frage: Wie komme ich an die Ableitung dieser Funktion ran?

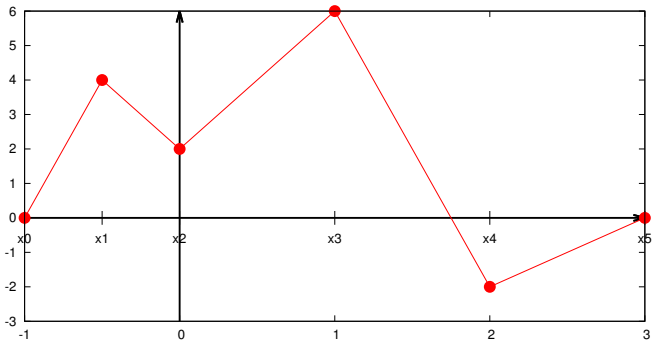
### 7.1.1 Erste Ableitung

Es sei $I = [0, 1]$ und $f \in C^2(I)$ mit $f : I \to \mathbb{R}$.

Unsere diskrete Funtkion $f_h$ approximiere $f$ und sei folgendermaßen gegeben:

$$f_h(x_i) = f(x_i)$$
$$f_h|_{[x_i, x_{i+1}]} \in \mathbb{P}_1 \, .$$

wobei $I = \bigcup_{i=0}^{N-1} [x_i, x_{i+1}]$.




Bei $x_i$ ist dieses $f_h$ gar nicht differenzierbar. Da aber $f_h \approx f$ und $f$ sehr wohl differenzierbar ist, will man ja $f_h$ auch irgendwie ableiten können.

Die Ableitung kann angenähert werden durch den Differenzenquotient.

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \qquad \text{Vorwärtsdifferenzenquotient}$$
$$f'(x) \approx \frac{f(x) - f(x-h)}{h} \qquad \text{Rückwärtsdifferenzenquotient}$$

Wir wählen das arithmetische Mittel:

$$f'(x) \approx \frac{1}{2} \left( \frac{f(x+h) - f(x)}{h} + \frac{f(x) - f(x-h)}{h} \right) = \frac{f(x+h) - f(x-h)}{2\,h}$$

Für unsere diskrete Funktion $f_h$ sieht das dann so aus: Es sei $h_i := x_{i+1} - x_i$

$$f_h'(x_i) \approx \frac{f_h(x_{i+1}) - f_h(x_i)}{h_i} \qquad \text{Vorwärtsdifferenzenquotient}$$
$$f_h'(x_i) \approx \frac{f_h(x_i) - f_h(x_{i-1})}{h_{i-1}} \qquad \text{Rückwärtsdifferenzenquotient}$$

Und das arithmetische Mittel:

$$f_h'(x_i) \approx \frac{1}{2} \left( \frac{f_h(x_{i+1}) - f_h(x_i)}{h_i} + \frac{f_h(x_i) - f_h(x_{i-1})}{h_{i-1}} \right) = \underbrace{\frac{f_h(x_{i+1}) - f_h(x_{i-1})}{2\,h}}_{\text{für äquidistante Gitter mit } h = h_i}$$

### 7.1.2 Fehlerrechnung

Der Fehler setzt sich aus zwei Komponenten zusammen:

1. Verfahrensfehler  2. Rundungsfehler

Der Verfahrensfehler ist der Fehler zwischen dem exakten Wert der Ableitung und dem exakten Wert des Differenzenqotienten, denn der Differenzenquotient ist das Verfahren zur Berechnung der Ableitung

$$\left| f'(x) - \frac{f(x+h) - f(x-h)}{2\,h} \right|$$

Zur Berechnung dieses Fehlers stellen wir die Funktionsausdrücke im Differenzenquotienten durch Taylorentwicklungen dar:

Taylorentwicklung von $f$ um $x_0$, ausgewertet bei $x$ lautet $T_{f,x_0}(x) = f(x)$, genauer

$$T_{f,x_0}(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k\,.$$

Dann ist für $x_0 = x$ und $x = x \pm h$:

$$f(x+h) = T_{f,x}(x+h) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x)}{k!}h^k = f + h\,f' + \frac{1}{2}\,h^2\,f'' + \frac{1}{6}\,h^3\,f^{(3)} + \cdots$$

$$\text{und} \quad f(x-h) = T_{f,x}(x-h) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x)}{k!}(-h)^k = f - h\,f' + \frac{1}{2}\,h^2\,f'' - \frac{1}{6}\,h^3\,f^{(3)} + \cdots$$

Damit gilt weiter

$$\begin{aligned}
& & f(x+h) - f(x-h) &= 2\,h\,f' + \frac{1}{3}\,h^3\,f^{(3)} + \frac{2}{5!}\,h^5\,f^{(5)}\cdots \\
\Rightarrow & & \frac{f(x+h) - f(x-h)}{2\,h} &= f' + \frac{1}{6}\,h^2\,f^{(3)} + \frac{1}{5!}\,h^4\,f^{(5)}\cdots \\
\Rightarrow & \left| f'(x) - \frac{f(x+h) - f(x-h)}{2\,h} \right| &= \left| \frac{1}{6}\,h^2\,f^{(3)} + \frac{1}{5!}\,h^4\,f^{(5)}\cdots \right| \\
& & &= h^2 \left| \frac{1}{6}\,f^{(3)} + \underbrace{\frac{1}{5!}\,h^2\,f^{(5)}\cdots}_{\to 0\,,\ h\to 0} \right| \\
& & &\le C\,h^2 = \mathcal{O}(h^2)
\end{aligned}$$

Wir erhalten also insgesamt für dne Verfahrensfehler:

$$\left| f'(x) - \frac{f(x+h) - f(x-h)}{2\,h} \right| = \mathcal{O}(h^2)$$

Zusätzlich zum Verfahrensfehler produzieren wir aber auch Rundungsfehler. Bei der Auswertung von Funktionen auf dem Computer sind wir natürlich nicht infinitesimal genau. Wie ungenau unsere Rechnungen sind hängt speziell vom verwendetetn Prozessor ab. Es sei $f_\epsilon := f(x) \pm \epsilon$. Wir betrachten:

$$|D_\epsilon - D| := \left| \underbrace{\frac{f_\epsilon(x+h) - f_\epsilon(x-h)}{2\,h}}_{\text{Das rechnen wir}} - \underbrace{\frac{f(x+h) - f(x-h)}{2\,h}}_{\text{Das wollen wir rechnen}} \right|$$

$$\leq \frac{1}{2\,h} \left( |f_\epsilon(x+h) - f(x+h)| + |f_\epsilon(x-h) - f(x-h)| \right)$$

$$\leq \frac{\epsilon}{h}$$

Der Fehler, den wir insgesamt machen ist dann

$$\begin{aligned} \left| f'(x) - \frac{f_\epsilon(x+h) - f_\epsilon(x-h)}{2\,h} \right| &= |f'(x) - D_\epsilon| \\ &= |f'(x) - D + D - D_\epsilon| \\ &\leq \underbrace{|f'(x) - D|}_{\text{Verfahrensfehler}} + \underbrace{|D - D_\epsilon|}_{\text{Rundungsfehler}} \\ &\leq C\,h^2 + \frac{\epsilon}{h} \end{aligned}$$

Je nach Prozessor ist so ein $\epsilon$ von der Größenordnung $10^{-7}$ oder $10^{-16}$.

Wahl einer optimalen Gitterweite $h$:

$$\left| f'(x) - \frac{f_\epsilon(x+h) - f_\epsilon(x-h)}{2\,h} \right| \leq C\,h^2 + \frac{\epsilon}{h} =: E(h)$$

Wähle $h$ so, dass $E(h)$ minimal wird:

$$0 = E'(h) = C\,h - \frac{\epsilon}{h^2} \quad \Leftrightarrow \quad h = \sqrt{\frac{\epsilon}{C}}$$

Genau kriegen wir das nicht; wir kennen ja $C$ auch gar nicht, aber es gibt uns eine Idee von einer Größenordnung für $h$, die nicht unterschritten werden soll. So lange $h >> \sqrt{\epsilon}$ ist wird der Rundungsfehler nicht sichtbar werden.

### 7.1.3 Zweite Ableitung

Die zweite Ableitung approximieren wir im ersten Schritt mit dem Vorwärtsdifferenzenquotient auf die erste Ableitung:

$$f''(x) \approx \frac{f'(x+h) - f'(x)}{h}$$

Die noch auftretenden ersten Ableitungen approximieren wir nun mit dem Rückwärtsdifferenzenquotienten:

$$f'(x) \approx \frac{f(x) - f(x-h)}{h}$$
$$f'(x+h) \approx \frac{f(x+h) - f(x)}{h}$$

Zusammen führt das auf

$$f''(x) \approx \frac{f'(x+h) - f'(x)}{h} \approx \frac{f(x+h) - 2\,f(x) + f(x-h)}{h^2}$$

Es sei $\bar{f} = (\bar{f}_1, \ldots, \bar{f}_N) = (f_h(x_1), \ldots, f_h(x_N)) = (f(x_1), \ldots, f(x_N))$ die Knotenwerte von $f$. Dann gilt

$$\begin{pmatrix} f''(x_1) \\ \vdots \\ \vdots \\ \vdots \\ f''(x_N) \end{pmatrix} \approx \frac{1}{h^2} \begin{pmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{pmatrix} \begin{pmatrix} \bar{f}_1 \\ \vdots \\ \vdots \\ \vdots \\ \bar{f}_N \end{pmatrix}$$

(Die Randknoten ignorieren wir zunächst; mehr dazu später)

## 7.2 Das numerische Verfahren

Die Finite-Differenzen Methode ist ein Verfahren, um Differentialgleichungen zu lösen unter Zuhilfenahme der Differenzenquotienten als Verfahren zur Berechnung der auftretenden Ableitungen. Das ist jetzt so formuliert, dass kein Mensch versteht was gemeint ist. Machen wir doch besser zur Veranschaulichung ein Beispiel. Es sei folgendes Randwertproblem zu lösen:

$$\begin{aligned} -f''(x) + f(x) &= (1+\pi^2)\,\sin(\pi\,x) \quad \text{in } I = [0,1] \\ f(0) = f(1) &= 0 \qquad\qquad\qquad\qquad \text{“Dirichlet-Randwerte ”} \end{aligned} \tag{8}$$

Im Moment interessiert uns in keinster Weise, welchen physikalischen Vorgang das $f$ beschreiben soll. Es ist nur ein Modellproblem. Wir diskretisieren das Problem und verwenden dafür die Notationen: $f_i = f(x_i)$ und $g_i := (1+\pi^2)\,\sin(\pi\,x_i)$

$$\begin{aligned} f_1 &= 0 \\ \frac{-f_{i-1} + 2\,f_i - f_{i+1}}{h^2} + f_i &= g_i \qquad\qquad i = 2,\ldots,N-1 \\ f_N &= 0 \end{aligned}$$

$\Leftrightarrow$

$$\begin{aligned} f_1 &= 0 \\ -f_{i-1} + (2+h^2)\,f_i - f_{i+1} &= h^2\,g_i \qquad\qquad i = 2,\ldots,N-1 \\ f_N &= 0 \end{aligned}$$

In Matrix-Vektor Schreibweise sieht das viel eleganter aus, nämlich so: $S\,\bar{f} = \bar{g}$ mit

$$\begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -1 & 2+h^2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2+h^2 & -1 \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_{N-1} \\ f_N \end{pmatrix} = \begin{pmatrix} 0 \\ h^2\,g_2 \\ \vdots \\ \vdots \\ h^2\,g_{N-1} \\ 0 \end{pmatrix} \tag{9}$$

Beachten Sie wie die Konstruktion der ersten und letzten Zeile der Matrix dafür sorgen, dass die Randwerte automatisch erfüllt sind. Das sieht jetzt nicht nur elegant aus, sondern lässt sich auch elegant lösen.

Wir fordern nun zum Modell leicht veränderte Randbedingungen, nämlich geben wir nun erste Ableitung am rechten Rand vor:

$$\begin{aligned} -f''(x) + f(x) &= (1+\pi^2)\,\sin(\pi\,x) \quad \text{in } I = [0,1] \\ f(0) &= 0 \qquad\qquad\qquad\qquad \text{“Dirichlet-Randwert”} \\ f'(1) &= \pi \qquad\qquad\qquad\qquad \text{“Neumann-Randwert ”} \end{aligned} \tag{10}$$

Diskretisiertes System lautet nun:

$$\begin{aligned} f_1 &= 0 \\ \frac{-f_{i-1} + 2\,f_i - f_{i+1}}{h^2} + f_i &= g_i \qquad\qquad i = 2, \ldots, N-1 \\ \frac{f_N - f_{N-1}}{h} &= \pi \end{aligned}$$

$$\begin{aligned} f_1 &= 0 \\ -f_{i-1} + (2 + h^2)\,f_i - f_{i+1} &= h^2\,g_i \qquad\qquad i = 2, \ldots, N-1 \\ f_N - f_{N-1} &= h\,\pi \end{aligned}$$

In Matrix-Vektor Schreibweise - wieder bestechend elegant: $S\,\bar{f} = \bar{g}$ mit

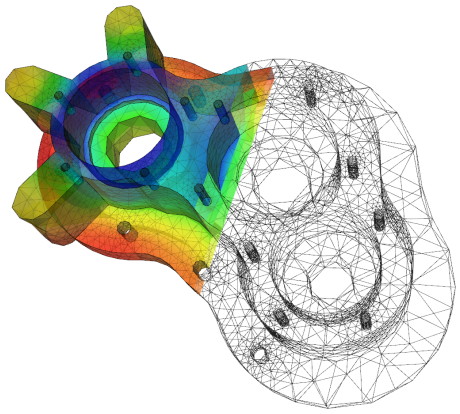
$$\begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -1 & 2+h^2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2+h^2 & -1 \\ 0 & \cdots & \cdots & -1 & 1 \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_{N-1} \\ f_N \end{pmatrix} = \begin{pmatrix} 0 \\ h^2\,g_2 \\ \vdots \\ \vdots \\ h^2\,g_{N-1} \\ h\,\pi \end{pmatrix} \tag{11}$$

# –8–
# Finite-Elemente Methode

Kapitel 8.1 ist **nicht** prüfungsrelevant. Es dient einzig Ihrer Horizonterweiterung.

Die Finite-Elemente Methode (FEM ) ist die bekannteste und am meisten angewandte numerische Lösungsmethode im Bereich wissenschaftlich technischer Aufgabenstellungen. Mit der FEM können physikalische Vorgänge (beispielsweise Kraftwirkungen auf deformierbare Festkörper) simuliert werden, deren Verlauf sich nicht oder nur sehr aufwendig mit anderen Mitteln bestimmen lässt.

Der Festkörper mit beliebiger Form wird in viele Teilkörper einfacher Form - die "Finiten-Elemente" - aufgeteilt, deren physikalisches Verhalten aufgrund ihrer einfachen Geometrie mit einfachen Ansatzfunktionen gut wiedergegeben werden kann. Im Pumpengehäuse rechts sind die Elemente mit schwarzen Linien angedeutet. Die Ansatzfunktionen enthalten Parameter, die in der Regel eine physikalische Bedeutung besitzen, wie z.B. die Verschiebung eines materiellen Punktes zu einem bestimmten Zeitpunkt. Die Suche nach der Bewegungsfunktion ist auf diese Weise auf die Suche nach den Parametern der Funktionen zurückgeführt. In dem immer mehr Parameter oder immer höherwertigere Ansatzfunktionen benutzt werden, kann die Genauigkeit der Näherungslösung verbessert werden.



Wärmeverteilung in einem Pumpengehäuse. (Quelle: Wikipedia)

Ihre Popularität verdankt die FEM zum einen dem Umstand, dass ihre Entwicklung in wesentlichen Etappen mit der Entwicklung leistungsfähiger Computer zusammenfällt und sie daher von vornherein computergerecht formuliert wurde. Zum anderen entspricht die Art der Modellierung sehr der Arbeitsweise des Ingenieurs und gestattet auch eine Anwendung bei komplexen Problemen.

Mit der FE-Methode können Probleme aus verschiedenen physikalischen Disziplinen berechnet werden, da es sich grundsätzlich um ein numerisches Verfahren zur Lösung von Differentialgleichungen handelt. Zunächst wird das Berechnungsgebiet in eine beliebig große Anzahl von Elementen unterteilt. Diese Elemente sind "endlich" (finit) und nicht unendlich (infinit) klein. Das Aufteilen des Gebiets in eine bestimmte Anzahl Elemente finiter Größe, die sich mit einer endlichen Zahl von Parametern beschreiben lassen, gab der Methode den Namen "Finite-Elemente-Methode".

Innerhalb dieser Elemente werden Ansatzfunktionen definiert (z. B. lokale Ritz-Ansätze je Element). Setzt man diese Ansatzfunktionen in die zu lösende Differentialgleichung ein, erhält

man zusammen mit den Anfangs-, Rand- und Übergangsbedingungen ein Gleichungssystem, welches in der Regel numerisch gelöst wird. Die Größe des zu lösenden Gleichungssystems hängt maßgeblich von der Anzahl der finiten Elemente ab. Seine Lösung stellt letztlich die numerische Lösung der betrachteten Differentialgleichung dar.[2]

Ich möchte diesem Text aus Wikipedia noch etwas hinzufügen: Die Gleichungssysteme, die man durch diese Wahl der Diskretisierung erhält sind dünn besetzt, das heißt, dass die Matrizen wenige nicht Null Einträge besitzt. Darin liegt schon mal ein Vorteil der FEM. Es werden nämlich nur nicht Null Einträge gespeichert. Hat eine 1000x1000-Matrix etwa Tridiagonalgestalt, so werden statt $10^6$ eben nur $3 \cdot 10^3$ float- oder double Zahlen gespeichert. Das macht die Rechnung nicht nur schlank in Bezug auf den Speicher sondern auch schnell. Bei den typischen Lösungsalgorithmen (meist iterativ) für die zu lösenden LGS ist oft die Matrix-Vektor-Multiplikation gerade das rechenaufwendigste. Bei dünn besetzten Matrizen, die sparsam gespeichert sind führt man entsprechend wenige Multiplikationen aus, was zu einer abermaligen Beschleunigung der Rechnung führt.

Kapitel 8.1 beschreibt die groben Schritte, die hinter der Theorie der FE-Methode liegt anhand eines einfachen eindimensionalen Modellproblems. Es ist dem geneigte Leser gewidment. Für unseren Unterricht - um der Problematik der Zeitknappheit Rechnung zu tragen - befassen wir uns mit der reinen Bedienung des `pdetool` von Matlab, das auf einer FEM Diskretisierung basiert und der Fragestellung, wie man die Toolbox veranlasst ein gewünschtes Problem zu lösen. Eine Anleitung dazu ist im Tutorial `11_FEM_PDE.pdf` gegeben. Des Weiteren wollen wir uns mit der Datenstruktur von FE-Programmen befassen, damit wir mit den berechneten Lösungen im Postprocessing weiterrechnen können.

## 8.1 Beschreibung in einer Raumdimension am Modellproblem

### 8.1.1 Das klassische Problem

Sei $I = [a, b] \subset \mathbb{R}$, $f, g : T \to \mathbb{R}$ gegeben. Gesucht ist $u : I \to \mathbb{R}$ mit

$$\begin{aligned} -u'' &= f \quad \text{in } I \\ u &= g \quad \text{auf } \partial I \end{aligned} \tag{12}$$

### 8.1.2 Schwache Formulierung

Multipliziere die Gleichung in (12) mit einer Testfunktion $\varphi \in C_0^\infty(I)$ und Integriere über $I$. Durch partielle Integration können wir uns die zweite Ableitung vom Hals schaffen. Es ist

$$\int_I u''\varphi \, dx = \underbrace{[u'\varphi]_a^b}_{=0\,,\ \text{da } \varphi\big|_{\partial I}=0} - \int_I u'\varphi' \, dx \,.$$

[2] Quelle: Wikipedia

Damit erhalten wir die schwache Formulierung

Sei $I = [a, b] \subset \mathbb{R}$, $f, g : T \to \mathbb{R}$ gegeben. Gesucht ist $u : I \to \mathbb{R}$ mit

$$\int_I u'\varphi' \, dx = \int_I f\varphi \, dx \quad \forall \varphi \in C_0^\infty(I) \tag{13}$$

+ Randwerte

### 8.1.3 Diskretisierung

Wir definieren folgende Funktionenräume

| | |
|---|---|
| $X := H^{1.2}(I)$ | der Raum, indem $u$ und $\varphi$ "leben" |
| $\mathring{X} := \mathring{H}^{1.2}(I)$ | Funktion aus $X$ mit Nullrandwerten |
| $X_h \subset X \wedge \mathring{X}_h \subset \mathring{X}$ | diskrete Teilräume mit endlicher Dimension $N$ |

Idee: Wir wollen alle gesuchten Funktionen als Linearkombination mit Basisfunktionen des entsprechenden Funktionenraumes darstellen, etwa

$$u(x) = \sum_l u_l \psi_l(x) \qquad \text{mit} \qquad u \in X = \mathrm{Span}\{\psi_l\}\,,\ u_l \in \mathbb{R}\,,$$

wobei die $\psi_l$ Basisfunktionen sind und $u_l$ zu berechnende Koeffizienten. Nun ist aber $X$ ein unendlich dimensionaler Raum, so dass wir unendlich viele Koeffizienten zu berechnen hätten.

Das geht nicht numerisch und deshalb approximieren wir den kontinuierlichen durch einen diskreten Funktionenraum endlicher Dimension. Diesen Räumen und sonst allem, was diskret (endlich dimensional) ist, haften wir einen Index $h$ an. Das Problem (13) wird genau wie vorher beschrieben, nur erhält nun alles diesen Index $h$ und wir meinen dann natürlich auch etwas anderes.

Wir suchen $u_h \in X_h$, so dass

$$\int_I u_h'\varphi_h' \, dx = \int_I f_h\varphi_h \, dx \quad \forall \varphi_h \in \mathring{X}_h \tag{14}$$

+ Randwerte

Sinn: Angefangen haben wir mit Gleichungen, die aus physikalischer Sicht sinnvoll sind und die wir in Gleichung (12) gesammelt haben. Die Frage ist nun berechtigt, was unser

schwaches, diskretes $u_h$ aus Gleichung (14) mit dem klassischen $u$ aus Gleichung (12) zu tun hat.

Beim ersten Schritt sind wir vom klassischen Problem zur schwachen Formulierung übergegangen. Streng genommen müssten wir unserer gesuchten Funktion $u$ bei diesem Wechsel schon einen anderen Namen geben, denn die Lösung der schwachen Gleichung muss nicht zwingend auch die Lösung des klassischen Problems sein. Von der Lösung des klassischen Problems wird zweite Ableitung verlangt, von der Lösung des schwachen Problems hingegen nicht. Kann man von einer schwachen Lösung genügend Regularität zeigen, so ist diese auch Lösung des klassischen Problems. Es gilt

$$\begin{aligned} & u \text{ Lösung von (12)} \\ \Leftrightarrow \quad & u \text{ Lösung von (13) und } u \in C^2(I) \qquad \text{(Regularität!)} \end{aligned}$$

Das ist deshalb wichtig, weil wir ja eigentlich an einer Lösung von (12) interessiert sind und alle Schritte, die wir tun auf dem Weg zu einer Lösung, nicht dazu führen dürfen, dass wir am Ende was ganz anderes berechnen.

Beim zweiten Schritt haben wir von einem unendlichen zu einem endlich dimensionalen Funktionenraum gewechselt. Wenn die Diskretisierung $X_h$ stabil ist, werden wir durch die Lösung $u_h$ von (14) eine quantitativ "gute" Näherung an die Lösung $u$ von (13) erhalten. "Gut" heißt, dass

$$\|u - u_h\|_X \leq C\, h^\alpha \qquad \text{für } \alpha > 0$$

erfüllt ist oder in anderen Worten gesagt, dass

$$\lim_{h \to 0} \|u - u_h\|_X \to 0$$

gilt. $u_h$ konvergiert schneller gegen $u$, wenn $\alpha$ groß ist. Wie $\alpha$ genau aussieht, hängt von der Norm $\|\cdot\|_X$ ab und von der Diskretisierung selbst.

### 8.1.4 Konstruktion von $X_h$

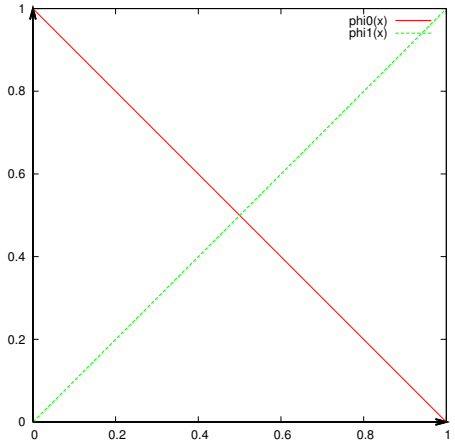
Die Frage, die sich als nächstes stellt ist nun etwas konkreter: Wie genau sieht $X_h$ aus? Anders ausgedrückt: Wie genau sehen die Basisfunktionen von $X_h$ aus? Das klären wir jetzt.

Basisfunktionen von $X_h$ werden zunächst auf einem Referenzintervall $\hat{I} := [0, 1]$ definiert. Sie sind dann per Transformation auf beliebige Intervalle übertragbar.

Es gibt eine Vielzahl von möglichen Basisfunktionen, gerade in höheren Raumdimensionen. Wir werden uns auf polynomiale einschränken. Genau genommen betrachten wir zunächst nur lineare Ansatzfunktionen (eine übliche Bezeichnung für die Basisfunktionen). Es gilt stets für $\hat{x}_0 = 0$ und $\hat{x}_1 = 1$

$$\hat{\varphi}_i(x_j) = \delta_{ij}\,.$$

**Beispiel 24** lineare Elemente



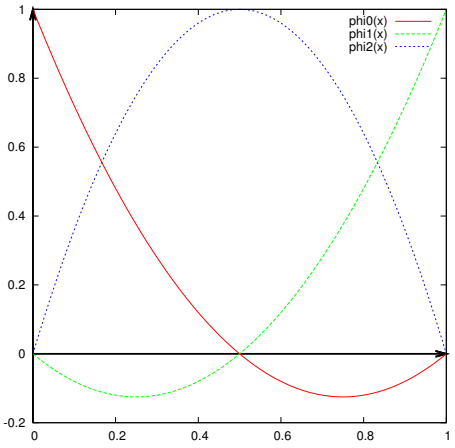

Auf $\hat{I}$ definieren wir $\hat{\varphi}\big|_{\hat{I}} \in \mathbb{P}_1$ durch:

$$\begin{aligned} \hat{\varphi}_0(\hat{x}) &:= 1 - \hat{x} \\ \hat{\varphi}_1(\hat{x}) &:= \hat{x} \end{aligned}$$

$$\begin{aligned} \Rightarrow \quad \hat{\varphi}_0'(\hat{x}) &= -1 \\ \hat{\varphi}_1'(\hat{x}) &= 1 \end{aligned}$$

$$\begin{aligned} \Rightarrow \quad \hat{\varphi}_0''(\hat{x}) &= 0 \\ \hat{\varphi}_1''(\hat{x}) &= 0 \end{aligned}$$

**Beispiel 25** quadratische Elemente




Auf $\hat{I}$ definieren wir $\hat{\varphi}\big|_{\hat{I}} \in \mathbb{P}_2$ durch:

$$\begin{aligned} \hat{\varphi}_1(\hat{x}) &:= \hat{x}\,(2\hat{x} - 1) \\ \hat{\varphi}_0(\hat{x}) &:= \hat{\varphi}_1(1 - \hat{x}) \\ \hat{\varphi}_2(\hat{x}) &:= 4\,\hat{x}(1 - \hat{x}) \end{aligned}$$

$$\begin{aligned} \Rightarrow \quad \hat{\varphi}_1'(\hat{x}) &= 4\,\hat{x} - 1 \\ \hat{\varphi}_0'(\hat{x}) &= -\hat{\varphi}_1' \\ \hat{\varphi}_2'(\hat{x}) &= 4(1 - 2\,\hat{x}) \end{aligned}$$

$$\begin{aligned} \Rightarrow \quad \hat{\varphi}_1''(\hat{x}) &= 4 \\ \hat{\varphi}_0''(\hat{x}) &= \hat{\varphi}'' \\ \hat{\varphi}_2''(\hat{x}) &= -8 \end{aligned}$$

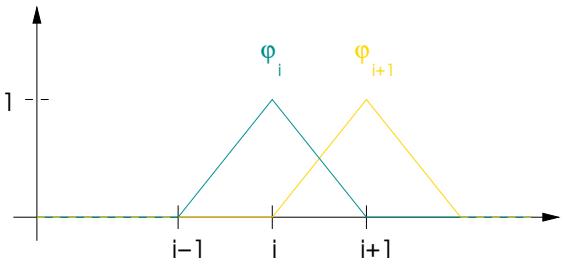
Natürlich arbeiten wir nicht zwingend auf dem Intervall $[0, 1]$ und vor allem mit mehr als nur zwei, drei Freiheitsgraden. Unser Intervall ist $I = [a, b]$. Dies zerlegen wir in $N$ Teilintervalle

$$I_k := [x_{k+1}, x_k]\,,\ k = 0, \ldots, N-1\,,$$

mit $x_0 = a$ und $x_N = b$, so daß

$$\begin{aligned} I &= \bigcup_{k=1}^{N} I_k \\ &= [x_0, x_1] \cup [x_1, x_2] \cup \cdots \cup [x_{N-2}, x_{N-1}] \cup [x_{N-1}, x_N] \\ &= [a, x_1] \cup [x_1, x_2] \cup \cdots \cup [x_{N-2}, x_{N-1}] \cup [x_{N-1}, b] \end{aligned}$$

Wir bleiben zunächst mal bei den stückweise linearen Elementen, also $\mathbb{P}_1$. Für jedes Teilintervall setzen wir Basisfunktionen $\varphi_i(x), i = 0, \ldots, N+1$ an mit folgenden Eigenschaften:




$$\varphi_i(x)\big|_{I_k} \in \mathbb{P}_1(I_k)$$
$$\varphi_i(x_j) = \delta_{ij}$$

Die $x_i$ sind die Knoten, an den Teilintervallgrenzen. $u_i := u_h(x_i)$ beschreiben die Werte der Funktion an den Knoten. Mit diesen sogenannten Koeffizienten und den oben beschriebenen Basisfunktionen lassen sich alle Funktionen aus $X_h$ darstellen als

$$u_h(x) = \sum_{i=0}^{N} u_i \varphi_i(x) \tag{15}$$

und

$$\varphi_h(x) = \sum_{i=0}^{N} \varphi_i(x)\,. \tag{16}$$

Bem: Dass die $u_i$ Funktionswerte sind hat mit unserer speziellen Wahl der Ansatzfunktionen zu tun. Das ist nicht immer so. Es gibt Ansatzfunktionen da beschreiben Koeffizienten etwa Mittelwerte von Integralen.

### 8.1.5 Das Modell in Matrix–Vektor Darstellung

Die Darstellungen (15) und (16) verwenden wir in Gleichung (14). Dann sieht der $j$–te Summand so aus:

$$\sum_{i=0}^{N} u_i \int_I \varphi_i'(x)\varphi_j'(x)\,dx = \int_I f_h(x)\varphi_j(x)\,dx \tag{17}$$

Mit der Definition der Steifigkeitsmatrix $(S_{ij})_{0\le i,j\le N}$ und rechten Seite $(b_j)_{0\le j\le N}$ durch

$$S_{ij} := \int_I \varphi_i'(x)\varphi_j'(x)\,dx$$
$$b_j := \int_I f_h(x)\varphi_j(x)\,dx$$

und den Bezeichnungen

$$\mathbf{u} := (u_0, \ldots, u_N)$$
$$\mathbf{b} := (b_0, \ldots, b_N)$$

und

$$\mathbf{S} := (S_{ij})_{0 \le i,j \le N}$$

können wir das Problem (17) in Matrix–Vektor Darstellung umformulieren zu

Gesucht sind die Koeffizienten $\mathbf{u}$, so dass

$$\mathbf{u}^T \mathbf{S} = \mathbf{b} \qquad (18)$$

+ Randwerte

gilt.

Wir sollten uns hier und da daran erinnern, dass es noch die Randwerte zu verarbeiten gibt. Mathematisch kann man die auch in den Funktionenraum reinstecken und nach Funktionen aus $X_g := \{v \in X \mid v|_{\partial I} = g\}$ suchen, aber die praktische Handhabung von Dirichlet– oder auch Neumannrändern ist so ergreifend einfach, dass wir uns diese Mühe sparen und sie dafür bis kurz vor Schluss mitschleppen.

Bem: Wenn $\mathbf{S}$, wie in unserem Fall gerade symmetrisch ist, so gilt $\mathbf{u}^T\mathbf{S} = \mathbf{b} \Leftrightarrow \mathbf{S}\mathbf{u} = \mathbf{b}$. Aber Achtung! Nur wenn $\mathbf{S}$ symmetrisch ist!

Wenn wir also ein Gleichungssystem der Form (18) haben, untersuchen wir es auf seine Eigenschaften und lassen einen passenden Gleichungssystemlöser darauf los. In unserem Fall wäre das ein vorkonditioniertes (Diagonalvorkonditionierung) CG–Verfahren. Aber soweit sind wir noch nicht. Erst einmal müssen wir das Gleichungssystem haben.

### 8.1.6 Berechnung der Systemmatrix

Zunächst zerlegen wir das Integral über die Teilintervalle $I_k$:

$$\int_I \varphi_i' \varphi_j' \, dx = \sum_{k=1}^{N} \underbrace{\int_{I_k} \varphi_i' \varphi_j' \, dx}_{\text{Dieses Integral transformieren wir auf } \hat{I}. \text{ Dort kennen wir es bereits.}}$$

Jedes Teilintegral

$$\int_{I_k} \varphi_i' \varphi_j' \, dx$$

transformieren wir mittels der Transformation

$$\begin{aligned} \Phi^k : \hat{I} &\to I_k \\ \hat{x} &\mapsto x \end{aligned}$$

auf $\hat{I}$. (Siehe Abbildung 7.) Wir lassen im weiteren den Index $k$ der besseren Übersicht wegen weg.

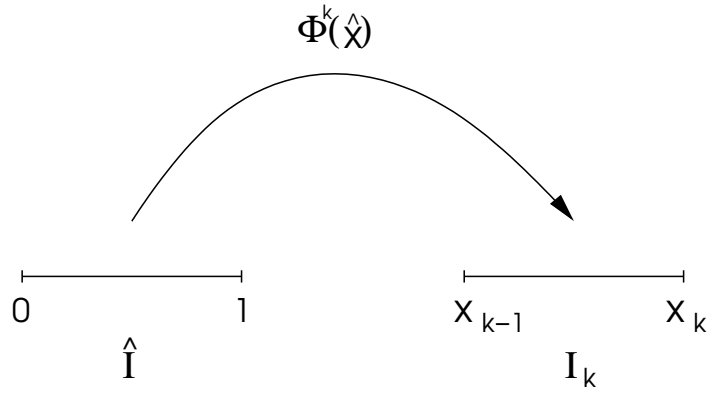



Abbildung 7: Transformation auf das Standardintervall

Für Funktionen

$$\hat{v} : \hat{I} \to \mathbb{R} \qquad \text{und} \qquad v : I_k \to \mathbb{R}$$

gelten die Beziehungen

$$\hat{v}(\hat{x}) = v(\Phi(\hat{x})) \qquad \wedge \qquad v(x) = \hat{v}(\Phi^{-1}(x))\,.$$

Damit erhalten wir die Ableitungen

$$\hat{v}_{\hat{x}}(\hat{x}) = v_x(\Phi(\hat{x}))\Phi_{\hat{x}}(\hat{x}) \qquad \wedge \qquad v_x(\Phi(\hat{x})) = \hat{v}_{\hat{x}}(\hat{x})(\Phi_{\hat{x}}(\hat{x}))^{-1}\,. \tag{19}$$

Die Ableitung des linken Terms ergibt sich direkt aus der Kettenregel. Beim rechten Term kommt noch die Ableitung der Umkehrfunktion dazu. Sicherheitshalber schauen wir uns das noch mal genauer an:

$$\frac{\partial}{\partial x} v(\Phi(\hat{x})) = \frac{\partial}{\partial x} \hat{v}(\Phi^{-1}(x)) = \hat{v}_{\hat{x}}(\Phi^{-1}(x)) \frac{\partial}{\partial x}(\Phi^{-1}(x)) \tag{20}$$

Die Ableitung der Umkehrfunktion $\Phi^{-1}(x)$ geht so:

$$\frac{\partial}{\partial x}\Phi^{-1}(x) = \frac{1}{\Phi_{\hat{x}}(\Phi^{-1}(x))} = \frac{1}{\Phi_{\hat{x}}(\hat{x})} = (\Phi_{\hat{x}}(\hat{x}))^{-1}$$

Das setzen wir in (20) ein und erhalten insgesamt

$$\frac{\partial}{\partial x} v(\Phi(\hat{x})) = \hat{v}_{\hat{x}}(\Phi^{-1}(x))(\Phi_{\hat{x}}(\hat{x}))^{-1} = \hat{v}_{\hat{x}}(\hat{x})(\Phi_{\hat{x}}(\hat{x}))^{-1}$$

Voilá!

Bem: Vorsicht ist geboten beim hochgestellten $-1$, denn $\Phi^{-1}(x)$ bezeichnet die Umkehrfunktion von $\Phi$, wärend $\Phi(x)^{-1}$ den Kehrwert der Funktion $\Phi$, also $\frac{1}{\Phi(x)}$ beschreibt! Jaja. So isses.

Auch ist $\frac{\partial}{\partial x} g(f(x))$ von $g_y(f(x))$ zu unterscheiden. Bei ersterem muss nachdifferenziert werden und beim zweiten Term nicht. ($y = f(x)$!)

Damit das jetzt nicht allzu abstrakt erscheint: Wir können $\Phi$ und seine Umkehrfunktion ganz konkret hinschreiben. Für Intervalle, also eindimensionale Simplizes quasi, kann eine Transformation etwa so aussehen:

$$\Phi(\hat{x}) := (x_k - x_{k-1})\,\hat{x} + x_{k-1} \qquad [0,1] \to [x_{k-1}, x_k]$$
$$\Phi^{-1}(x) := \frac{x - x_{k-1}}{x_k - x_{k-1}} \qquad [x_{k-1}, x_k] \to [0,1]$$

Und damit kennen wir auch die Ableitungen:

$$\Phi_{\hat{x}}(\hat{x}) = x_k - x_{k-1} = |I_k| =: h_k \qquad \Phi_x^{-1}(x) = \frac{1}{x_k - x_{k-1}} = \frac{1}{|I_k|} = \frac{1}{h_k}$$

Die Ableitungen sind konstant, da die Transformation selbst linear ist – man beachte die geradezu bestechende Einfachheit dieses Sachverhaltes; man möchte meinen, dass da was nicht stimmt! Aber nein. Wir dürfen uns getrost zurücklehnen und berechtigt hoffen, dass es sich auch im weitern um eine einfache Aneinanderreihung folgelogischer Trivialitäten handeln wird – und stehen im Zusammenhang mit der Länge des Intervalls, welches sie transformieren. Das ist sinnvoll, denn wird das Intervall $\hat{I}$ auf $I_k$ gestreckt, gilt also $h_k >> 1$, so wird die entsprechende Steigung der abgebildeten Funktion, die ja nun flacher geworden ist, nachkorrigiert. Wir brauchen das, weil wir das Integral der Steifigkeitsmatrix transformieren wollen und da stehen nunmal Ableitungen der Basisfunktionen drin.

(Was ich eigentlich sagen will: Das Transformieren ist gar nicht so schlimm, wie es vielleicht zu Beginn des Kapitel anmutete. Liegt auch sicher an der Wahl des Buchstabens. Da müsste noch ein wenig Didaktik rein. Aber ich finde das $\Phi$ sooo schön..... Genug davon!)

Wir können gerne einmal von einem äquidistanten Gitter auf $I$ ausgehen, das heißt wir nehmen einmal an, dass alle $h_k$ paarweise gleich sind und nennen die Gitterweite der Einfachheit halber $h$. Es kommt dann schon ein wenig was anderes heraus aber ich möchte gerne etwas bestimmtes Zeigen. Wir können später noch von "unstrukturierten" Gittern sprechen.

Bevor wir nun den ganzen Ausdruck der Steifigkeitsmatrix transformieren wollen erinnern wir uns kurz an die Transformationsformel aus der Analysis I–Vorlesung (hehe), die da lautet:

$$\int_{g(a)}^{g(b)} f(x)\,dx = \int_a^b f(g(y))g'(y)\,dy \tag{21}$$

Und weil die Vorlesung schon so lange her ist überzeugen wir uns gerade noch davon. Es sei $F$ eine Stammfunktion von $f$, das heißt $F' = f$. Dann gilt

$$\int_a^b f(g(y))g'(y)\,dy = \int_a^b F'(g(y))g'(y)\,dy = \int_a^b (F(g(y)))'\,dy$$
$$= F(g(b)) - F(g(a)) = \int_{g(a)}^{g(b)} F'(x)\,dx = \int_{g(a)}^{g(b)} f(x)\,dx\,.$$

Schön. Das hätten wir also auch.

Wir ersetzen jetzt einfach $g$ durch $\Phi$, $f$ durch $v$ und $[a,b]$ durch $\hat{I}$. Damit erhalten wir, wenn wir die Transformationsformel, gerade wie sie ist, abschreiben

$$\int_I v(x)\,dx = \int_{\hat{I}} v(\Phi(\hat{x}))\Phi'(\hat{x})\,d\hat{x} = \int_{\hat{I}} \hat{v}(\hat{x})\Phi'(\hat{x})\,d\hat{x}$$

Das gleiche kann man auch mit dem Produkt von Funktionen machen

$$\int_I v(x)\,w(x)\,dx = \int_{\hat{I}} \hat{v}(\hat{x})\,\hat{w}(\hat{x})\Phi'(\hat{x})\,d\hat{x}\,,$$

indem man einfach in der Transformationsformel $f$ durch $v \cdot w$ ersetzt. Aufpassen muss man nur, wenn Ableitungen auftreten. Die Faustregel ist: "Pro auftretende Ableitung im Integranten, minus Eins, muss der Term, der über $\hat{I}$ integriert durch $\Phi'(\hat{x}) = h_k$ geteilt werden."

Wir setzen die Ableitung aus (19)

in die Transformationsformel (21) ein und erhalten

$$\int_I v_x(x)\,dx = \int_{\hat{I}} \hat{v}_{\hat{x}}(\hat{x})\Phi'(\hat{x})^{-1}\Phi'(\hat{x})\,d\hat{x} = \int_{\hat{I}} \hat{v}_{\hat{x}}(\hat{x})\,d\hat{x}\,.$$

Ein Produkt aus zwei differenzierten Funktionen entwickelt sich so

$$\int_I v_x(x)w_x(x)\,dx = \int_{\hat{I}} \hat{v}_{\hat{x}}(\hat{x})\Phi'(\hat{x})^{-1}\hat{w}_{\hat{x}}(\hat{x})\Phi'(\hat{x})^{-1}\Phi'(\hat{x})\,d\hat{x} = \int_{\hat{I}} \hat{v}_{\hat{x}}(\hat{x})\hat{w}_{\hat{x}}(\hat{x})\Phi'(\hat{x})^{-1}\,d\hat{x}$$

und ein Produkt aus einer Funktion mit einer diefferenzierten dann eben so

$$\int_I v(x)w_x(x)\,dx = \int_{\hat{I}} \hat{v}(\hat{x})\hat{w}_{\hat{x}}(\hat{x})\Phi'(\hat{x})^{-1}\Phi'(\hat{x})\,d\hat{x} = \int_{\hat{I}} \hat{v}(\hat{x})\hat{w}_{\hat{x}}(\hat{x})\,d\hat{x}$$

$$\int_I v(x)\ dx = \int_{\hat{I}} \hat{v}(\hat{x})\Phi'(\hat{x})\ d\hat{x}$$

$$\int_I v(x)\, w(x)\ dx = \int_{\hat{I}} \hat{v}(\hat{x})\, \hat{w}(\hat{x})\Phi'(\hat{x})\ d\hat{x}$$

$$\int_I v_x(x)\ dx = \int_{\hat{I}} \hat{v}_{\hat{x}}(\hat{x})\ d\hat{x}$$

$$\int_I v(x)w_x(x)\ dx = \int_{\hat{I}} \hat{v}(\hat{x})\hat{w}_{\hat{x}}(\hat{x})\ d\hat{x}$$

$$\int_I v_x(x)w_x(x)\ dx = \int_{\hat{I}} \hat{v}_{\hat{x}}(\hat{x})\hat{w}_{\hat{x}}(\hat{x})\Phi'(\hat{x})^{-1}\ d\hat{x}$$

$$\int_I v_x(x)w_x(x)z_x(x)\ dx = \int_{\hat{I}} \hat{v}_{\hat{x}}(\hat{x})\hat{w}_{\hat{x}}(\hat{x})\hat{w}_{\hat{x}}(\hat{x})\Phi'(\hat{x})^{-2}\ d\hat{x}$$

Abbildung 8: Transformationen von $\hat{I}$ nach $I$

Der besseren Übersicht wegen fassen wir diese Transformationen noch einmal in Abbildung 8 zusammen und kommen zurück zur Steifigkeitsmatrix.

Unser zu berechnendes Integral sieht so so aus

$$\int_{I_k} \varphi_i'\varphi_j'\ dx = \int_{\hat{I}} \hat{\varphi}_{\hat{i}}'\hat{\varphi}_{\hat{j}}'(\Phi_{\hat{x}}^k)^{-1}\ d\hat{x} = \frac{1}{h_k} \underbrace{\int_0^1 \hat{\varphi}_{\hat{i}}'\hat{\varphi}_{\hat{j}}'\ d\hat{x}}_{=:\hat{S}_{ij}\ \text{(Elementsteifigkeitsmatrix)}} \quad , \hat{i},\hat{j} \in \{0,1\}$$

Abbildung 9 diene dazu, die Beziehungen zwischen $i$ und $\hat{i}$, respektive $\varphi_i$ und $\hat{\varphi}_{\hat{i}}$ zu verdeutlichen.

Die Elementsteifigkeitsmatrix $\hat{\mathbf{S}} \in \mathbb{R}^{2\times 2}$

$$\hat{\mathbf{S}} = \begin{pmatrix} \hat{S}_{00} & \hat{S}_{01} \\ \hat{S}_{10} & \hat{S}_{11} \end{pmatrix}$$

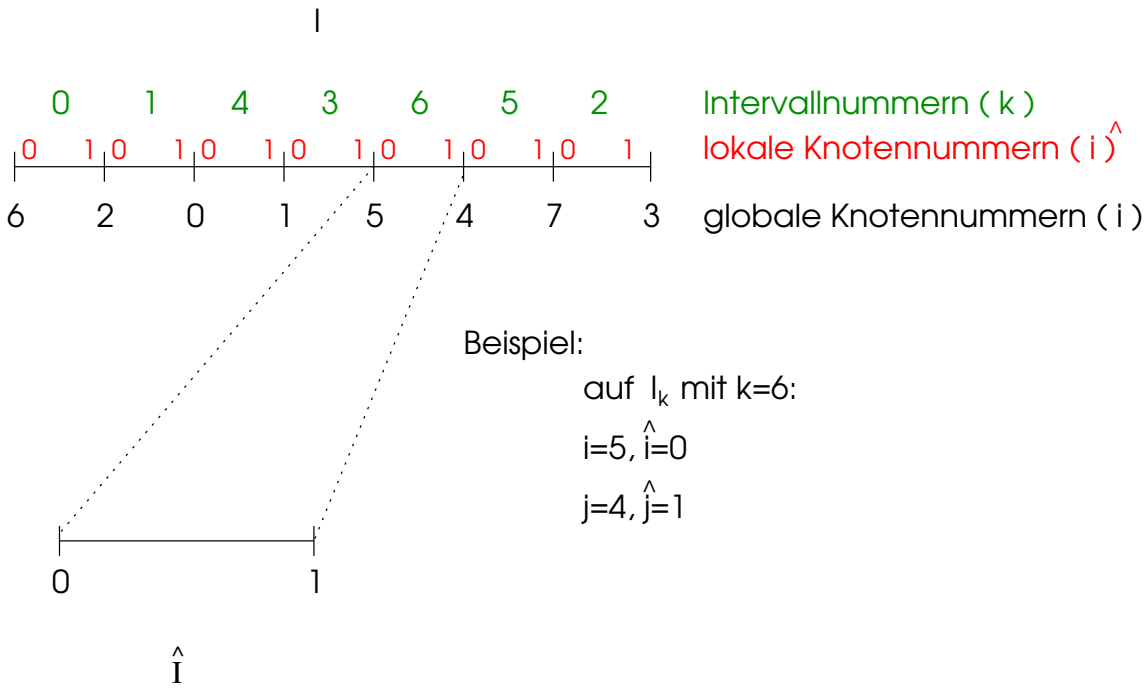



Abbildung 9: Beziehung zwischen globalen Knotennummern $i$ und lokalen Knotennummern $\hat{i}$

berechnet sich so:

$$\hat{S}_{00} = \int_0^1 \hat{\varphi}_0'^2(\hat{x})\, d\hat{x} = \int_0^1 1\, d\hat{x} = 1$$

$$\hat{S}_{01} = \int_0^1 \hat{\varphi}_0'(\hat{x})\hat{\varphi}_1'(\hat{x})\, d\hat{x} = -\int_0^1 1\, d\hat{x} = -1$$

$$\hat{S}_{01} = \hat{S}_{10} = -1$$

$$\hat{S}_{11} = \int_0^1 \hat{\varphi}_1'^2(\hat{x})\, d\hat{x} = \int_0^1 1\, d\hat{x} = 1$$

Damit ergibt sich auf $I_k$ insegesamt:

$$\left( \int_{I_k} \varphi_i'(x)\varphi_j'(x)\, dx \right)_{i,j\in\mathcal{N}_k} = \frac{1}{h_k} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

Dabei ist

$$\mathcal{N}^k := \{i \mid x_i \in \bar{I}_k\}$$

die Menge aller Indizes der Knoten, die zum Intervall $I_k$ gehören.

Beim Aufsummieren auf die Gesamtsteifigkeitsmatrix müssen die Werte, zugehörig der lokalen Knoten, an die entsprechenden Stellen, zugehörig der globalen Knoten, sortiert werden.

Ein anschauliches Beispiel: Wir sind auf dem Intervall der Nummer $3$, also $I_3$. Das Intervall wird durch die globalen Knoten $x_1$ und $x_5$ aufgespannt: $I_3 = [x_1, x_5]$. Wie in Abbildung 9. Dann wird für diesen Anteil folgendes auf die Steifigkeitsmatrix ($\hat{\mathbf{S}} \in \mathbb{R}^{7\times 7}$) aufsummiert (natürlich nachdem auf Null inititalisiert wurde, versteht sich...):

$$\mathbf{S} \quad += \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{h_3}\hat{S}_{00} & 0 & 0 & 0 & \frac{1}{h_3}\hat{S}_{01} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{h_3}\hat{S}_{10} & 0 & 0 & 0 & \frac{1}{h_3}\hat{S}_{11} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

oder anders dargestellt:

$$S_{11} += \frac{1}{h_3}\hat{S}_{00} \qquad S_{15} += \frac{1}{h_3}\hat{S}_{01}$$
$$S_{51} += \frac{1}{h_3}\hat{S}_{10} \qquad S_{55} += \frac{1}{h_3}\hat{S}_{11}$$

Insgesamt sieht $\mathbf{S}$ für dieses Beispiel so aus: (mit $\hat{\mathbf{S}}^k := \frac{1}{h_k}\hat{\mathbf{S}}$)

$$\mathbf{S}$$

$$= \begin{pmatrix}
\hat{S}^1_{11}+\hat{S}^4_{00} & \hat{S}^4_{01} & \hat{S}^1_{10} & 0 & 0 & 0 & 0 & 0 \\
\hat{S}^4_{10} & \hat{S}^4_{11}+\hat{S}^3_{00} & 0 & 0 & 0 & \hat{S}^5_{01} & 0 & 0 \\
\hat{S}^1_{01} & 0 & \hat{S}^0_{11}+\hat{S}^1_{00} & 0 & 0 & 0 & \hat{S}^0_{10} & 0 \\
0 & 0 & 0 & \hat{S}^2_{11} & 0 & 0 & 0 & \hat{S}^2_{10} \\
0 & 0 & 0 & 0 & \hat{S}^6_{11}+\hat{S}^5_{00} & \hat{S}^3_{10} & 0 & \hat{S}^5_{01} \\
0 & \hat{S}^5_{10} & 0 & 0 & \hat{S}^3_{01} & \hat{S}^3_{11}+\hat{S}^6_{00} & 0 & 0 \\
0 & 0 & \hat{S}^0_{01} & 0 & 0 & 0 & \hat{S}^0_{00} & 0 \\
0 & 0 & 0 & \hat{S}^2_{01} & \hat{S}^5_{10} & 0 & 0 & \hat{S}^5_{11}+\hat{S}^2_{00}
\end{pmatrix}$$

$$= \begin{pmatrix}
\frac{1}{h_1}+\frac{1}{h_4} & -\frac{1}{h_4} & -\frac{1}{h_1} & 0 & 0 & 0 & 0 & 0 \\
-\frac{1}{h_4} & \frac{1}{h_4}+\frac{1}{h_3} & 0 & 0 & 0 & -\frac{1}{h_5} & 0 & 0 \\
-\frac{1}{h_1} & 0 & \frac{1}{h_0}+\frac{1}{h_1} & 0 & 0 & 0 & -\frac{1}{h_0} & 0 \\
0 & 0 & 0 & \frac{1}{h_2} & 0 & 0 & 0 & -\frac{1}{h_2} \\
0 & 0 & 0 & 0 & \frac{1}{h_6}+\frac{1}{h_5} & -\frac{1}{h_3} & 0 & -\frac{1}{h_5} \\
0 & -\frac{1}{h_5} & 0 & 0 & -\frac{1}{h_3} & \frac{1}{h_3}+\frac{1}{h_6} & 0 & 0 \\
0 & 0 & -\frac{1}{h_0} & 0 & 0 & 0 & \frac{1}{h_0} & 0 \\
0 & 0 & 0 & -\frac{1}{h_2} & -\frac{1}{h_5} & 0 & 0 & \frac{1}{h_5}+\frac{1}{h_2}
\end{pmatrix}$$

Bei äquidistanten Stützstellen mit $h := h_0 = \cdots h_6$ sähe das dann so aus:

$$= \frac{1}{h}\begin{pmatrix}
2 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 2 & 0 & 0 & 0 & -1 & 0 & 0 \\
-1 & 0 & 2 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 2 & -1 & 0 & -1 \\
0 & -1 & 0 & 0 & -1 & 2 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & -1 & -1 & 0 & 0 & 2
\end{pmatrix}$$

Das sieht dann schon fast nach Finite Differenzen aus. Wir werden sehen, dass wir beim Aufstellen der rechten Seite noch ein $h$ dazubekommen. Es ist wirklich so, dass stückweise lineare Finite Elemente in einer Raumdimension das gleiche Gleichungssystem liefern wie die Finite Differenzen Methode. Da fragt man sich natürlich was der ganze Aufwand soll?

1. Durch das Zulassen unsortierter Knotennummerierung kann leicht ein lokaler Verfeinerungs– und Vergröberungsalgorithmus dazu programmiert werden. Der ist was sehr nützliches, was eine Menge Speicher und Rechenzeit einsparen kann.

2. Wenn man das ganze setting der Gittergenerierung und Datenstrukturen ordentlich beisammen hat kann man relativ einfach die linearen Elemente durch andere ersetzen. Das ist der Vorteil gegenüber der Finite Differenzen Methode.

3. Wenn man mal die Datenstruktur in der 1-D Situation erfaßt hat, kann man die dann direkt in die 2– oder 3–D Situation übertragen. Direkt als erstes ein 3–D Programm zu starten wäre hart.

### 8.1.7 Aufstellen der rechten Seite und Massematrix

Über das Aufstellen der rechten Seite wollen wir mathematisch gar nicht viel Worte verlieren. Eine Variante (die einfachste!) ist $f$ durch $f_h$ zu ersetzen und dann

$$\int_I f_h(x)\varphi_j(x)\,dx = \sum_{i=0}^{N} f_i \underbrace{\int_I \varphi_i(x)\varphi_j(x)\,dx}_{=:M_{ij}\ \text{Massematrix}} = \left(\mathbf{f}^T\mathbf{M}\right)_j$$

Die Massematrix wird genauso berechnet wie die Steifigkeitsmatrix, wozu einmalig die Elementmassematrix auf $\hat{I}$ aufgestellt werden muss:

$$\begin{aligned}
\hat{M}_{00} &= \int_0^1 \hat{\varphi}_0(\hat{x})^2\,d\hat{x} = \int_0^1 (1-\hat{x})^2\,d\hat{x} = \frac{1}{3}\\
\hat{M}_{01} &= \int_0^1 \hat{\varphi}_0(\hat{x})\hat{\varphi}_1(\hat{x})\,d\hat{x} = \int_0^1 \hat{x}(1-\hat{x})\,d\hat{x} = \frac{1}{6}\\
\hat{M}_{10} &= \hat{M}_{01} = \frac{1}{6}\\
\hat{M}_{11} &= \int_0^1 \hat{\varphi}_1(\hat{x})^2\,d\hat{x} = \int_0^1 \hat{x}^2\,d\hat{x} = \frac{1}{3}
\end{aligned}$$

$$\left(\int_{I_k} \varphi_i(x)\varphi_j(x)\,dx\right)_{ij\in\mathcal{N}^k} = \frac{h_k}{6}\begin{pmatrix} 2 & 1\\ 1 & 2\end{pmatrix}$$

Dann ergibt sich in unserem Beispiel genau wie bei der Steifigkeitsmatrix die ganze Massematrix mit $\hat{\mathbf{M}}^k = h_k\hat{\mathbf{M}}$

$$\mathbf{M}$$

$$= \begin{pmatrix} \hat{M}^1_{11} + \hat{M}^4_{00} & \hat{M}^4_{01} & \hat{M}^1_{10} & 0 & 0 & 0 & 0 & 0 \\ \hat{M}^4_{10} & \hat{M}^4_{11} + \hat{M}^3_{00} & 0 & 0 & 0 & \hat{M}^5_{01} & 0 & 0 \\ \hat{M}^1_{01} & 0 & \hat{M}^0_{11} + \hat{M}^1_{00} & 0 & 0 & 0 & \hat{M}^0_{10} & 0 \\ 0 & 0 & 0 & \hat{M}^2_{11} & 0 & 0 & 0 & \hat{M}^2_{10} \\ 0 & 0 & 0 & 0 & \hat{M}^6_{11} + \hat{M}^5_{00} & \hat{M}^3_{10} & 0 & \hat{M}^5_{01} \\ 0 & \hat{M}^5_{10} & 0 & 0 & \hat{M}^3_{01} & \hat{M}^3_{11} + \hat{M}^6_{00} & 0 & 0 \\ 0 & 0 & \hat{M}^0_{01} & 0 & 0 & 0 & \hat{M}^0_{00} & 0 \\ 0 & 0 & 0 & \hat{M}^2_{01} & \hat{M}^5_{10} & 0 & 0 & \hat{M}^5_{11} + \hat{M}^2_{00} \end{pmatrix}$$

$$= \frac{1}{6} \begin{pmatrix} 2(h_1 + h_4) & h_4 & h_1 & 0 & 0 & 0 & 0 & 0 \\ h_4 & 2(h_4 + h_3) & 0 & 0 & 0 & h_5 & 0 & 0 \\ h_1 & 0 & 2(h_0 + h_1) & 0 & 0 & 0 & h_0 & 0 \\ 0 & 0 & 0 & h_2 & 0 & 0 & 0 & h_2 \\ 0 & 0 & 0 & 0 & 2(h_6 + h_5) & h_3 & 0 & h_5 \\ 0 & h_5 & 0 & 0 & h_3 & 2(h_3 + h_6) & 0 & 0 \\ 0 & 0 & h_0 & 0 & 0 & 0 & h_0 & 0 \\ 0 & 0 & 0 & h_2 & h_5 & 0 & 0 & 2(h_5 + h_2) \end{pmatrix}$$

Bei äquidistanten Stützstellen mit $h := h_0 = \cdots h_6$ sähe das dann so aus:

$$= \frac{h}{6} \begin{pmatrix} 4 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 4 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 4 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 4 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 4 \end{pmatrix}$$

Wir schreiben für unser Beispiel mal die Gleichung in (18) auf und nehmen dafür auf $I = [0, 1]$

$$u(x) = x^3 + 1 \qquad f(x) = -6x \qquad u(0) = 1 \qquad u(1) = 2$$

Mit $h = \frac{1}{6}$ erhalten wir den Knotenvektor

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} = \frac{1}{6} \begin{pmatrix} 2 \\ 3 \\ 1 \\ 7 \\ 5 \\ 4 \\ 0 \\ 6 \end{pmatrix} \quad \text{und} \quad \mathbf{u} = \frac{1}{216} \begin{pmatrix} 224 \\ 243 \\ 217 \\ 559 \\ 341 \\ 280 \\ 216 \\ 432 \end{pmatrix} \quad \Rightarrow \quad \mathbf{f} = - \begin{pmatrix} 2 \\ 3 \\ 1 \\ 7 \\ 5 \\ 4 \\ 0 \\ 6 \end{pmatrix}$$

Die rechte Seite:

$$\mathbf{b} = \frac{-1}{36} \begin{pmatrix} 2 & 3 & 1 & 7 & 5 & 4 & 0 & 6 \end{pmatrix} \begin{pmatrix} 4 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 4 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 4 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 4 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 4 \end{pmatrix} = \frac{-1}{36} \begin{pmatrix} 12 \\ 18 \\ 6 \\ 20 \\ 30 \\ 24 \\ 1 \\ 36 \end{pmatrix}$$

Die linke Seite:

$$\mathbf{u}^T \mathbf{S} = \frac{6}{216} \cdot$$

$$\begin{pmatrix} 224 & 243 & 217 & 559 & 341 & 280 & 216 & 432 \end{pmatrix} \begin{pmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 & 0 & -1 & 0 & 0 \\ -1 & 0 & 2 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 2 & -1 & 0 & -1 \\ 0 & -1 & 0 & 0 & -1 & 2 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 2 \end{pmatrix}$$

$$= \frac{1}{36} \begin{pmatrix} -12 \\ -18 \\ -6 \\ 127 \\ -30 \\ -24 \\ -1 \\ -36 \end{pmatrix}$$

Das passt doch prima!

# Index