Vorlesungsmitschrieb

Numerik I

Prof. Dr. Christian Lubich

im Wintersemester 2006/2007
an der Eberhard-Karls-Universtität Tübingen

gesetzt von Markus Klein mit LaTeX

Letzte Änderung: 12. Februar 2010

Vorwort

Dieses Scriptum entstand als Live-Mitschrieb im Wintersemester 2006/2007 bei Prof. Dr. Christian Lubich an der Eberhard-Karls-Universität Tübingen.
Es erhebt keinen Anspruch auf Vollständigkeit oder Richtigkeit. Es ist *nicht* durch Prof. Lubich autorisiert.

Einen besonderen Dank möchte ich an dieser Stelle Frau Patricia Baur und Herrn Jonas Friz aussprechen, die viele Fehler und Unstimmigkeiten verbessert haben.

Bei Fragen, Wünschen oder Verbesserungsvorschlägen freue ich mich über jede E-Mail an
studium@kleiner-markus.de.
Vielen Dank!

Inhaltsverzeichnis

1 Numerische Integration

1.1 Erste Beispiele von Quadraturformeln

1.1 Motivation
Wir möchten $\int_a^b f(x)dx$ berechnen.
Wir unterteilen zunächst das Intervall in Teilintervalle: $[a = x_0, x_1], [x_1, x_2], \ldots, [x_{n-1}, x_n = b]$.
Damit ergibt sich:

$$\int_a^b f(x)dx = \sum_{j=1}^{N} \left(\int_{x_{j-1}}^{x_j} f(x)\, dx \right)$$

Wir bezeichnen im folgenden die Intervallängen als $h_j := x_j - x_{j-1}$.
Dann ist:

$$\int_{x_{j-1}}^{x_j} f(x)dx = \int_{x_{j-1}}^{x_{j-1}+h_j} f(x)dx$$

1.2 Beispiel (Rechtecksregel)
Exemplarisch berechnen wir für ein spezilles x_0:

$$\int_{x_0}^{x_0+h} f(x)dx \approx hf(x_0)$$

1.3 Beispiel (Mittelpunktsregel)
Analog zur Rechtecksregel können wir statt dem Funktionswert am Anfangspunkt des Intervalls auch im Mittelpunkt des Intervalls wählen:

$$\int_{x_0}^{x_0+h} f(x)dx \approx hf\left(x_0 + \frac{h}{2}\right)$$

1.4 Beispiel (Trapezregel)
Wir nähern im Folgenden die Funktion nicht durch Rechtecke, sondern durch Trapeze, wodurch sich die folgende Näherung ergibt:

$$\int_{x_0}^{x_0+h} f(x)dx \approx \frac{h}{2}\left(f(x_0)+f(x_0+h)\right)$$

1.5 Beispiel (Simpsonregel)
Wir können natürlich auch versuchen, eine quadratische Funktion als Näherung anzunehmen und erhalten eine Näherung:

$$\int_{x_0}^{x_0+h} f(x)dx \approx \frac{h}{6}\left(f(x_0)+4\cdot f\left(x_0+\frac{h}{2}\right)+f(x_0+h)\right)$$

Diese Formel ist für quadratische und sogar für kubische Funktionen exakt.

1.6 Definition (Allgemeine Form einer Quadraturformel)
Allgemein ist eine *Quadraturformel* gegeben durch:

$$\int_{x_0}^{x_0+h} f(x)dx \approx h\cdot\sum_{i=1}^{s} b_i f(x_0+c_i h)$$

wobei c_i die Knoten sind und b_i die Gewichte.

1.7 Beobachtung (Betrachtung der bereits bekannten Formeln)
Wir erhalten für die bereits bekannten Formeln folgende Werte:

Name	Anz. Knoten	Gewichte	Knoten
Rechtecksregel	$s=1$	$b_1=1$	$c_1=0$
Mittelpunktsregel	$s=1$	$b_1=1$	$c_1=\frac{1}{2}$
Trapezregel	$s=2$	$b_1=b_2=\frac{1}{2}$	$c_1=0$, $c_2=1$
Simpsonregel	$s=3$	$b_1=b_3=\frac{1}{6}$, $b_2=\frac{2}{3}$	$c_1=0$, $c_2=\frac{1}{2}$, $c_3=1$

1.2 Ordnung einer Qudraturformel

1.8 Motivation
Wir wollen untersuchen, wie „gut“ eine Quadraturformel (QF) ist.

1.9 Definition (Ordnung einer QF)
Eine Quadraturformel ist gegeben durch $(b_i,c_i)_{i=1}^{s}$. Diese hat die *Ordnung* p, falls sie den exakten Integralwert liefert für alle Polynome vom Grad $\leq p-1$.

1.10 Proposition (Charakterisierung der Ordnung)
Eine Quadraturformel hat genau dann Ordnung p, wenn gilt:

$$\sum_{i=1}^{s} b_i c_i^{q-1} = \frac{1}{q} \quad \forall q \leq p$$

Beweis
Es gilt per Definition:

$$\begin{aligned} \text{QF hat Ordnung } p &\iff \text{QF ist exakt für } f(x) = (x - x_0)^{q-1} \quad \forall q \leq p \\ &\iff \int_0^h x^{q-1} dx = h \sum_{i=1}^{s} b_i (c_i h)^{q-1} \end{aligned}$$

Da nun aber gilt

$$\int_0^h x^{q-1} dx = \frac{h^q}{q}, \qquad h \sum_{i=1}^{s} b_i (c_i h)^{q-1} = h^q \sum_{i=1}^{s} b_i c_i^{q-1}$$

ergibt sich die Bedingung:

$$\sum_{i=1}^{s} b_i c_i^{q-1} = \frac{1}{q} \quad \forall q \leq p$$

□

1.11 Beispiel
Für die Rechtecksregel ergibt sich die Ordnung 1, die Mittelpunktsregel ist exakt für lineare Funktionen, hat also Ordnung 2.

1.12 Satz (Eindeutigkeit der QF bei bestimmter Ordnung)
Seien $c_1 < \ldots < c_s$ gegeben. Dann existieren eindeutig $b_1, \ldots, b_s$, sodaß die QF mindestens die Ordnung s hat. Es gilt dabei:

$$b_i = \int_0^1 \ell_i(x) dx \text{ mit } \ell_i(x) = \frac{\prod_{j \neq i}(x - c_j)}{\prod_{j \neq i}(c_i - c_j)}$$

wobei ℓ_i *Lagrange-Polynom* heißt und hat die folgenden Eigenschaft:

$$\ell_i(c_k) = \begin{cases} 1, & i = k \\ 0, & i \neq k \end{cases}$$

und es gilt $\deg \ell_i = s - 1$.

Beweis
Falls die Ordnung $p \geq s$ ist, muß gelten:

$$\int_0^1 \ell_i(x)dx = \sum_{j=1}^{s} b_j \ell_i(c_j) = b_i$$

Damit ist die Eindeutigkeit der b_i festgestellt.

Sind umgekehrt die b_i so definiert, dann werden $\ell_1, \ldots, \ell_s$ durch die Quadraturformel exakt integriert. Da $\ell_1, \ldots, \ell_s$ linear unabhängige Polynome sind, spannen sie den Raum der Polynome bis zum Grad $s-1$ auf, d.h. für die Ordnung p gilt: $p \geq s$. □

1.13 Definition (symmetrische QF)
Eine QF heißt *symmetrisch* genau dann, wenn

1. $c_i = 1 - c_{s+1-i}$ (Knoten symmetrisch zu $\frac{1}{2}$)
2. $b_i = b_{s+1-i}$ (Gewichte sind dann jeweils symmetrisch gleich)

1.14 Beispiel
Die Trapezregel, die Mittelpunktsregel und die Simpsonregel sind symmetrisch.
Die Ordnungen sind jeweils gerade.

1.15 Satz (Eigenschaften von symmetrischen QF)
Die Ordnungen von symmetrischen QF sind immer gerade.

Beweis
Angenommen, die Ordnung ist $2m-1$, d.h. die QF sei für alle Polynome vom Grad $\leq 2m-2 = s-1$ exakt.
Sei f Polynom vom Grad $2m-1$.
Wir können f schreiben als $f(x) = c(x-\frac{1}{2})^{2m-1} + g(x)$, wobei $\deg g \leq 2m-2$.
Es reicht also zu zeigen, daß die QF den ersten Term exakt integriert:

$$\int_0^1 \left(x - \frac{1}{2}\right)^{2m-1} dx = 0$$

Wir betrachten weiterhin:

$$\begin{aligned}
\sum_{i=1}^{s} b_i \left(c_i - \frac{1}{2}\right)^{2m-1} &= \sum_{i=1}^{s} b_{s+1-i} \left(\frac{1}{2} - c_{s+1-i}\right)^{2m-1} \\
&= \sum_{j=1}^{s} b_j \left(\frac{1}{2} - c_j\right)^{2m-1} \\
&= -\sum_{j=1}^{s} b_j \left(c_j - \frac{1}{2}\right)^{2m-1}
\end{aligned}$$

Damit ist diese Summe ebenfalls 0, was zum Widerspruch führt. □

1.3 Untersuchung des Qudraturfehlers

1.16 Motivation
Wir wollen nun die QF für eine beliebige Funktion betrachten, die nicht exakt integriert wird.

1.17 Proposition
Sei $f : [a, b] \to \mathbb{R}$ „genügend oft“ diffbar. Wir betrachten nun den Fehler, also die Differenz von exaktem Integral und QF.
Dabei setzen wir $g_j(t) = f(x_{j-1} + th_j)$ und erhalten:

$$\begin{aligned}\int_a^b f(x)dx - \sum_{j=1}^{N} h_j \sum_{i=1}^{s} b_i f(x_{j-1} + c_i h_j) &= \sum_{j=1}^{N} \left(\int_{x_{j-1}}^{x_j} f(x)dx - h_j \sum_{i=1}^{s} b_i f(x_{j-1} + c_i h_j) \right) \\ &= \sum_{j=1}^{N} \ell_j \left(\int_0^1 g_j(t)dt - \sum_{i=1}^{s} b_i g_j(c_i) \right)\end{aligned}$$

Zur Untersuchung des Fehlers betrachten wir daher den Fehler eines Schrittes der QF auf $[0, 1]$:

$$E(g) = \int_0^1 g(t)dt - \sum_{i=1}^{s} b_i g(c_i)$$

1.18 Hilfssatz (Fehlerterm)
Die QF hat (mindestens) die Ordnung p und sei g p-mal stetig diffbar.
Dann gilt $E(g) = \int_0^1 K_p(\tau) g^{(p)}(\tau)\, d\tau$, wobei $K_p(\tau)$ der sog. *Peano-Kern* ist, der nur von der QF abhängt und explizit gegeben ist durch:

$$K_p(\tau) := E\left(t \mapsto \frac{((t-\tau)_+)^{p-1}}{(p-1)!} \right)$$

wobei $x_+ = \begin{cases} x, & x \geq 0 \\ 0, & \text{sonst} \end{cases}$.
Nach der Definition von E ergibt sich

$$K_p(\tau) = \frac{(1-\tau)^p}{p!} - \sum_{i=1}^{s} b_i \frac{((c_i - \tau)_+)^{p-1}}{(p-1)!}$$

Beweis
Der Beweis funktioniert durch die Taylorreihe mit Restterm in Integralform:

$$g(t) = \underbrace{g(0) + g'(0)\frac{t}{1!} + \ldots + g^{(p-1)}(0)\frac{t^{p-1}}{(p-1)!}}_{q(t)} + \underbrace{\int_0^t \frac{(t-\tau)^{p-1}}{(p-1)!} g^{(p)}(\tau) d\tau}_{} \quad 0 \leq t \leq 1$$

Dabei ist $\deg q(t) \leq p-1$ und der zweite Term ist dabei

$$\int_0^1 \frac{((t-\tau)_+)^{p-1}}{(p-1)!} g^{(p)}(\tau) d\tau$$

Wegen der Linearität ergibt sich:

$$E(g) = \underbrace{E(q)}_{=0} + \int_0^1 \underbrace{E\left(t \mapsto \frac{((t-\tau)_+)^{p-1}}{(p-1)!}\right)}_{K_p(\tau)} g^{(p)}(\tau) d\tau$$

□

1.19 Satz (Anwendung des Fehlerterms)
Wir wollen nun die festgestellte Formel anwenden auf $g(t) = f(x_0 + th)$. Dabei ergibt sich:

$$\int_{x_0}^{x_0+h} f(x)dx - h\sum_{i=1}^{s} b_i f(x_0 + c_i h) = hE(g) = h^{p+1} \int_0^1 K_p(\tau) f^{(p)}(x_0 + \tau h) d\tau$$

Damit können wir den Fehler auf die folgende Weise für ein Teilintervall $[x_0, x_0 + h]$ abschätzen:

$$\left| \int_{x_0}^{x_0+h} f(x)\, dx - h\sum_{i=1}^{s} b_i f(x_0 + c_i h) \right| \leq C_p \cdot h^{p+1} \cdot \max_{x \in [x_0, x_0+h]} |f^{(p)}(x)|$$

mit $C_p = \int_0^1 |K_p(\tau)| d\tau$.

1.20 Beispiel

1. Für die Mittelpunktsregel bei $p = 2$ ergibt sich dabei:

$$K_2(\tau) = \frac{(1-\tau)^2}{2} - \left(\frac{1}{2} - \tau\right)_+ = \begin{cases} \frac{\tau^2}{2}, & 0 \leq \tau \leq \frac{1}{2} \\ \frac{(1-\tau)^2}{2}, & \frac{1}{2} \leq \tau \leq 1 \end{cases}$$

Dabei ergibt sich $C_2 = \int_0^1 |K_2(\tau)|d\tau = \frac{1}{24}$. Damit ergibt sich für den Fehler der Mittelpunktsregel:

$$\left| \int_{x_0}^{x_0+h} f(x)dx - hf(x_0 + \frac{h}{2}) \right| \leq \frac{h^3}{24} \max_{x\in[x_0,x_0+h]} |f''(x)|$$

2. Für die Simpsonregel mit $p = 4$ ergibt sich $C_4 = \frac{1}{2880}$ und ein Fehler, der kleiner ist als

$$\frac{h^5}{2880} \max_{x\in[x_0,x_0+h]} |f^{(4)}(x)|$$

1.21 Proposition (Betrachtung des Fehlers auf dem Gesamtintervall)
Sei $[a, b]$ das Gesamtintervall.
Auf jedem einzelnen Teilintervall gelte

$$\left| \int_{x_{j-1}}^{x_j} f(x)dx - h_j \sum_{i=1}^{s} b_i f(x_{j-1} + c_i h_j) \right| \leq h_j \varepsilon_j$$

wobei $\varepsilon_j = C_p h_j^p \max_{x\in[x_{j-1},x_j]} |f^{(p)}(x)|$.
Mit der Dreiecksungleichung

$$\left| \int_a^b f(x)\,dx - \sum_{j=1}^{N} h_j \sum_{i=1}^{s} b_i f(x_{j-1} + c_i h_j) \right| \leq \underbrace{\sum_{j=1}^{N} h_j}_{b-a} \cdot \max_{j=1\ldots N} \varepsilon_j$$

Falls $h_j = h\ \forall j$ (äquidistante Unterteilung), dann gilt:

$$\left| \int_a^b f(x)\,dx - \sum_{j=1}^{N} h \sum_{i=1}^{s} b_i f(x_{j-1} + c_i h) \right| \leq (b-a) C_p h^p \max_{x\in[a,b]} |f^{(p)}(x)|$$

Eine bessere Strategie ist es, nicht eine äquidistante Unterteilung zu machen, sondern die Fehler auf den Teilintervallen ε_j etwa konstant zu halten.
Damit muß man eine variable Schrittweite h_j (sog. *adaptive* Schrittweiten) anpasst an den Verlauf der Kurve (für große Steigungen kleine Intervalle und umgekehrt).
Wir werden sehen, daß es einen Algorithmus gibt, mit dem sich die Schrittweiten gut bestimmt werden.

1.4 Quadraturformeln mit erhöhter Ordnung

1.22 Motivation
Falls die Knoten $c_1, \ldots, c_s$ beliebig sind, dann existiert dazu Gewicht $b_1, \ldots, b_s$, sodaß die zugehörige QF mit Gewichten $(b_i, c_i)_{i=1}^s$ mindestens die Ordnung s hat.
Wir wollen nun der Frage nachgehen, wie man c_i wählen muß, damit $p = s + m > s$ ist und eine Abschätzung finden, wie groß die Ordnung bezüglich s maximal sein kann.

1.23 Konstruktion (eines geeigneten Polynoms für die Vergrößerung der Ordnung)
Sei f ein Polynom vom Grad $\deg f \leq s + m - 1$. Wir wollen die QF möglichst exakt bestimmen.
Wir dividieren f (mit Rest) durch das folgende Polynom:

$$M(x) = \prod_{i=1}^{s} (x - c_i)$$

Dabei gilt $\deg M(x) = s$

Wir können also schreiben, daß $f(x) = M(x)g(x) + R(x)$, wobei $\deg R \leq s - 1$ und $\deg g \leq m - 1$. Es gilt:

$$\int_0^1 f(x)\,dx = \int_0^1 M(x)g(x)\,dx + \int_0^1 R(x)\,dx$$

Da die Quadraturformel exait sein soll, muß gelten:

$$\sum_{i=1}^{s} b_i f(c_i) = \underbrace{\sum_{i=1}^{s} b_i M(c_i) g(c_i)}_{=0} + \sum_{i=1}^{s} b_i R(c_i)$$

Wir sehen, daß $\int_0^1 R(x)\,dx = \sum_{i=1}^s b_i R(c_i)$ gilt und damit gilt

$$\int_0^1 f(x)\,dx = \int_0^1 R(x)\,dx$$

falls

$$\int_0^1 M(x)g(x)\,dx = 0$$

1.24 Satz (Quadraturformel mit erhöhter Ordnung)
Sei $(b_i, c_i)_{i=1}^s$ eine QF der Ordnung $p \geq s$. Dann gilt:

$$p = s + m \iff \int_0^1 M(x)g(x)\,dx = 0 \quad \forall g, \deg g \leq m-1$$

d.h. M ist orthogonal auf alle Polynome vom Grad $\leq m-1$ bezüglich des Skalarprodukts

$$\langle f, g \rangle = \int_0^1 f(x)g(x)\,dx$$

1.25 Satz (Maximale Ordnung)
Die Ordnung einer QF $(b_i, c_i)_{i=1}^s$ ist höchstens $2 \cdot s$ groß.

Beweis
Angenommen, es gelte $p \geq 2s+1$, d.h. die QF muß exakt sein für alle Polynome vom Grad $\leq 2s$.
Dann muß gelten, daß

$$\int_0^1 M(x)g(x)\,dx = 0 \quad \forall g,\ \deg g \leq s$$

Insbesondere muß dies gelten für $g = M$, also müßte gelten $\int_0^1 M^2(x)\,dx = 0$. Dies kann aber nicht sein, da M nicht die Nullfunktion ist, womit wir einen Widerspruch zur Annahme finden. □

1.26 Bemerkung
Wir werden sehen, daß es möglich ist, QF bis zur Ordnung $p = 2s$ zu konstruieren, die auf Gauss zurückgeht.

1.5 Orthogonale Polynome

1.27 Definition (gewichtetes Skalarprodukt)
Sei $\omega : (a, b) \to \mathbb{R}$ stetig (Gewichtsfunktion) mit

1. $\omega(x) > 0 \quad \forall x \in (a, b)$

2. $\int_a^b \omega(x)|x|^k\,dx < \infty$ für $k \in \mathbb{N}$ bel.

Wir betrachten nun ein *gewichtetes Skalarprodukt*:

$$\langle f, g\rangle = \int_a^b \omega(x) f(x) g(x)\,dx$$

auf dem Vektorraum

$$V = \left\{ f : [a,b] \to \mathbb{R} : f \text{ stetig} \wedge \int_a^b \omega(x)|f(x)|^2\,dx < \infty \right\}$$

Insbesondere sind alle Polynome in V.

1.28 Definition (Orthogonalität)
Wir definieren:

$$f \perp g :\iff \langle f, g\rangle = 0$$

1.29 Satz (Existenz und Eindeutigkeit von orthogonalen Polynomen)
Es existiert eine Folge von Polynomen $p_0, p_1, \ldots$ mit folgenden Eigenschaften:

1. $p_k(x) = 1 \cdot x^k + q(x)$, wobei $\deg q(x) \le k-1$
2. p_k ist orthogonal auf alle Polynome vom Grad kleiner als k.

Diese Polynome können berechnet werden durch folgende Rekursion:

$$p_{k+1}(x) = (x - \beta_{k+1})p_k(x) - \gamma_{k+1}^2 p_{k-1}(x)$$

mit den Startwerten $p_0(x) = 1$ und $p_{-1}(x) = 0$, während die Koeffizienten folgendermaßen gewählt sind:

$$\beta_{k+1} = \frac{\langle x p_k, p_k\rangle}{\langle p_k, p_k\rangle}, \qquad \gamma_{k+1}^2 = \frac{\langle p_k, p_k\rangle}{\langle p_{k-1}, p_{k-1}\rangle}$$

Beweis
Der Beweis geht auf das Gram-Schmidt-Orthogonalisierungsverfahren zurück.

Angenommen, es gibt bereits $p_0, p_1, \ldots, p_k$ konstruiert. Wir bestimmen nun p_{k+1}:

$$p_{k+1}(x) = x p_k(x) + \sum_{j=0}^{k} \alpha_j \cdot p_j(x)$$

Wir berechnen nun α_j mit der Bedingung

$$0 = \langle p_{k+1}, p_k \rangle = \langle xp_k, p_k \rangle + \sum_{j=0}^{k} \alpha_j \underbrace{\langle p_j, p_k \rangle}_{=0 \quad j<k} = \langle xp_k, p_k \rangle + \alpha_k \langle p_k, p_k \rangle$$

also ist $\alpha_k = -\frac{\langle xp_k, p_k \rangle}{\langle p_k, p_k \rangle} = -\beta_{k+1}$.
Für α_{k-1} ergibt sich:

$$0 = \langle p_{k+1}, p_{k-1} \rangle = \langle xp_k, p_{k-1} \rangle + \alpha_{k-1} \langle p_{k-1}, p_{k-1} \rangle$$

Womit sich $\alpha_{k-1} = -\frac{\langle xp_k, p_{k-1} \rangle}{\langle p_{k-1}, p_{k-1} \rangle}$ ergibt.
Man erhält wegen der Bilinearität $\langle xp_k, p_{k-1} \rangle = \langle p_k, xp_{k-1} \rangle = \langle p_k, p_k \rangle$ ergibt.
Wir erhalten

$$\alpha_{k-1} = -\frac{\langle p_k, p_k \rangle}{\langle p_{k-1}, p_{k-1} \rangle} = -\gamma_{k+1}^2$$

Für $j \leq k-2$:

$$0 = \langle p_k, p_j \rangle = \langle xp_k, p_j \rangle + \alpha_j \langle p_j, p_j \rangle, \qquad \langle xp_k, p_j \rangle = \langle p_k, \underbrace{xp_j}_{\deg \leq k-1} \rangle = 0$$

Damit ist $\alpha_j = 0$.
Weil wir nun eine Vorschrift haben, wie man solche Polynome konstruieren kann, die ersten Polynome bereits bestimmt haben, erfüllen wir bereits die Vorschriften für eine solche Polynomfolge, womit die Existenz gezeigt ist. □

1.30 Beobachtung (Wahl der Knoten)
Für eine QF höchstmöglicher Ordnung muß man $c_1, \ldots, c_s$ so wählen, daß

$$M(x) = (x - c_1) \cdot \ldots \cdot (x - c_s) = p_s(x)$$

das orthogonale Polynom bezüglich $(a, b) = (0, 1)$ mit $\omega(x) = 1$ ist.

1.31 Satz (Eigenschaft der Nullstellen des Polynoms)
Sei p_k ein orthogonales Polynom wie oben.
Alle Nullstellen von p_k sind reell, einfach und liegen im Intervall (a, b)

Beweis
Seien $x_1, \ldots, x_n$ jene Nullstellen von p_k, die reell sind, in (a, b) liegen, bei denen p_k das Vorzeichen wechselt (also ungerade Vielfachheit).
Es ist klar, daß $n \leq k$ ist.

Setze nun $g(x) = (x - x_1) \cdot \ldots \cdot (x - x_n)$ und damit gilt

$$\int_a^b \underbrace{\omega(x)}_{>0} p_k(x) g(x) dx = \langle p_k, g \rangle \neq 0$$

Weil kein Vorzeichenwechsel stattfindet, ist $\deg g \geq k$.
Also ist $n \geq k$ und damit $n = k$. □

1.32 Beispiel (für orthogonale Polynome)
Wir können folgende Polynome als Beispiele aufschreiben:

Bezeichnung	Intervall	ω	Name
$p_k^{\alpha,\beta}$	$(-1, 1)$	$(1-\alpha)^\alpha (1+\alpha)^\beta$	Jacobi $(\alpha, \beta > -1)$
p_k	$(-1, 1)$	1	Legendre $(p_k(1) = 1)$
T_k	$(-1, 1)$	$(1 - x^2)^{-\frac{1}{2}}$	Tschebyscheff
$L_k^{(\alpha)}$	$(0, \infty)$	$e^{-x} x^\alpha$	Leguerre $(\alpha > -1)$
H_k	$(-\infty, \infty)$	e^{-x^2}	Hermite

1.6 Gaußsche Quadraturformeln

1.33 Satz (Existenz und Eindeutigkeit der Gaußschen QF)
Es existiert eine eindeutig bestimmte QF $(b_i, c_i)_{i=1}^s$ der Ordnung $p = 2s$.
Dabei ist

1. $c_i = \frac{1}{2}(1 + \gamma_i)$, wobei die γ_i Nullstellen des Legendre-Polynoms (normiert durch $P_k(1) = 1$) P_s, also mit $\deg P_s = s$, sind und wir ordnen die Nullstellen aufsteigend.
2. b_i so bestimmt, daß $p \geq s$ ist.

Beweis
Wegen $p \geq s$ gilt:

$$p = 2s \iff \int_0^1 M(x) g(x)\, dx = 0 \quad \forall g, \deg g \leq s - 1$$

Mit einer Substitution: $t = 2x - 1, [0,1] \to [-1,1]$ ergibt sich:

$$\iff \int_{-1}^{1} M\left(\frac{1+t}{2}\right) f(t)\,dt = 0 \quad \forall f, \deg f \le s-1$$
$$\iff M\left(\frac{1+t}{2}\right) = \text{const} \cdot P_s(t) \iff c_i = \frac{1}{2}(1+\gamma_i)$$

Dabei folgt die Eindeutigkeit der QF aus der Eindeutigkeit des orth. Polynoms. □

1.34 Bemerkung (Ausblick)
Erhalten die GAUSSschen QF die Monotonie des Integrals: $f \ge 0$ auf $[0,1] \Rightarrow \int_0^1 f(x)\,dx \ge 0$?
Wir werden im Folgenden eine Antwort darauf geben können.

1.35 Satz (positive Gewichte der Gaußschen Quadraturformel)
Die Gewichte einer QF der Ordnung $p \ge 2s-1$ sind positiv ($b_i > 0 \quad \forall i$), d.h. insbesondere, daß die Quadraturformel positiv ist, also

$$f \ge 0 \Rightarrow \sum_{i=1}^{s} b_i f(c_i) \ge 0$$

BEWEIS
Betrachte die LAGRANGE-Polynome ℓ_i ($\deg \ell_i = s-1, \ell_i(c_i) = \delta_{ij}$). Daraus folgt dann, daß $\deg \ell_i^2 = 2s-2$.
Wir betrachten nun

$$\int_0^1 \ell_i^2(x)\,dx = \sum_{j=1}^{s} b_j \ell_i^2(c_j) = b_i > 0 \quad \forall i \le s$$

□

1.36 Beispiel (für Gaußsche QF)

1. $s = 1$: $P_1(x)$, also $\gamma_1 = 0, c_1 = \frac{1}{2}, b_1 = 1$, damit

   $$\int_0^1 f(x)dx \approx 1 \cdot f\left(\frac{1}{2}\right)$$

   was die Mittelpunktsregel ergibt

2. $s = 2$: $P_2(x) = \frac{3}{2}x^2 - \frac{1}{2}$, also $\gamma_{1,2} = \pm\frac{\sqrt{3}}{3}, c_{1,2} = \frac{1}{2} \pm \frac{\sqrt{3}}{6}, b_1 = b_2 = \frac{1}{2}$, damit

   $$\int_0^1 f(x)\,dx \approx \frac{1}{2}\left(f\left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right) + f\left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right)\right)$$

3. $s = 3$: $P_3(x) = \frac{5}{2}x^3 - \frac{3}{2}, \gamma_2 = 0, \gamma_{1,3} = \pm\frac{\sqrt{15}}{5}$, damit:

$$\int_0^1 f(x)dx \approx \frac{1}{18}\left(5f\left(\frac{1}{2} - \frac{\sqrt{15}}{10}\right) + 8f\left(\frac{1}{2}\right) + 5f\left(\frac{1}{2} + \frac{\sqrt{15}}{10}\right)\right)$$

1.37 Bemerkung (Ausblick auf Numerik II)

1. Für große s benutzt man die Drei-Term-Rekursion, um ein Eigenwertproblem zu konstruieren, das z.B. durch das sog. QR-Verfahren gelöst werden kann.
2. $b_i = \frac{1-\gamma_i^2}{s^2(P_{s-1}(\gamma_i))^2}$

1.7 Ein adaptives Programm

1.38 Motivation

Sei eine QF (z.B. GAUSS s=15, p=30) gegeben.
Wir wollen auf folgendes Ziel heraus:
Wir möchten eine Funktion „(IN)TEGRAL(Function,a,b,`TOL`)“ (uralte Fortran-Funktion) erhalten, die für eine vorgegebene TOL(Genauigkeitstoleranz) aufomatisch eine Zerlegung (nicht notwendig äquidistant) des Intervalls $[a, b]$ wählt, sodaß folgendes gilt:

$$\left|\int_a^b f(x)\,dx - \text{numer. Resultat}\right| \leq \texttt{TOL} \cdot \int_a^b |f(x)|\,dx$$

1.39 Motivation (Problem)

Wir werden dabei mit folgenden Problemen konfrontiert:

1. Schätzen des Fehlers
2. Wahl der Zerlegung

den wir auf folgende Art wählen:

1. Wir tun so, als hätten wir bereits eine Fehlerabschätzung, konstruieren den Algorithmus und suchen im Nachhinein eine Fehlerabschätzung.
2. Für Teilintervalle $[x_0, x_0 + h] \subseteq [a, b]$ kann man berechnen:

$$\texttt{res}[x_0, x_0+h] := h\sum_{i=1}^{s} b_i f(x_0 + c_i h), \qquad \texttt{resabs}[x_0, x_0+h] := h \cdot \sum_{i=1}^{s} b_i |f(x_0 + c_i h)|$$

1.40 Algorithmus (Intervallzerlegung)

Damit erhalten wir folgenden Algorithmus zur Intervallzerlegung:

1. Berechne $\texttt{res}[a, b], \texttt{resabs}[a, b], \texttt{err}[a, b]$.
   Falls $\texttt{res}[a, b] \leq \texttt{TOL} \cdot \texttt{resabs}[a, b]$,
   dann nehme $\texttt{res}[a, b] \approx \int_a^b f(x)\,dx$,
   sonst:

2. Unterteile $[a,b]$ in $I_1 = [a, \frac{a+b}{2}], I_2 = [\frac{a+b}{2}, b]$.
   Falls $|\texttt{err}_1| + |\texttt{err}_2| \leq \texttt{TOL}|\texttt{resabs}_1 + \texttt{resabs}_2|$,
   dann akzeptiere $\texttt{res}_1 + \texttt{res}_2$ als Lösung,
   sonst:

3. Unterteile das Intervall, für das der Fehler am größten ist:
   Falls $\sum_{i=1}^{3} |\texttt{err}_i| \leq \texttt{TOL} \sum_{i=1}^{3} \texttt{resabs}$,
   dann nehme dies als Lösung oder unterteile das Intervall, bei dem der Fehler am größten ist und fahre analog fort.

Nun müssen wir uns nur noch um die Fehlerabschätzung kümmern.

1.41 Proposition (Fehlerabschätzung)
Wir suchen eine eingebettete QF, d.h. wir nehmen die gleichen Knoten wie vorgegeben und möchte eine neue QF erhalten mit $(\hat{b_i}, c_i)_{i=1}^{s}$ der Ordnung $\hat{p} < p$.
Falls (b_i, c_i) eine GAUSSsche QF ist, so ist ihre Ordnung $\hat{p} \leq s-1$.
Wäre nun $\hat{p} \geq s$, dann ist $\hat{b_i} = \int_0^1 \ell_i(x)\,dx = b_i$, wobei ℓ_i das LAGRANGE-Polynom ist.

Damit erhalten wir:

$$\texttt{ERR}_1 := h \cdot \sum_{i=1}^{s} b_i f(x_i + c_i h) - h \sum_{i=1}^{s} \hat{b_i} f(x_i + c_i h)$$

eine Approximation des Fehlers für die eingebettete QF ist damit:

$$\texttt{ERR}_1 = \underbrace{\left(h \cdot \sum_{i=1}^{s} b_i f(x_i + c_i h) - \int_{x_0}^{x_0+h} f(x)\,dx \right)}_{=Ch^{p+1}+\mathcal{O}(h^{p+2})} + \underbrace{\left(\int_{x_0}^{x_0+h} f(x)\,dx - h \sum_{i=1}^{s} \hat{b_i} f(x_i + ci_h) \right)}_{=\hat{C}h^{\hat{p}+1}+\mathcal{O}(h^{\hat{p}+2})}$$

wobei für kleine h der rechte Fehler einiges größer ist.

Die Grundsätzliche Schwierigkeit besteht darin, daß eine wirkliche Fehlerabschätzung nur für die eingebettete (schlechtere) QF möglich ist. Man möchte aber einen Fehler für die bessere QF schätzen.
Dazu ergeben sich mehrere Möglichkeiten:

1. Wir setzen $\texttt{err}[x_0, x_0+h] = \texttt{ERR}_1$, dies ist aber zu pessimistisch.

2. Wir setzen $\texttt{err}[x_0, x_0+h] = \texttt{ERR}_1^2$.

3. Wir setzen hier im ersten Schritt das Gewicht des mittleren Knoten 0, im nächsten Schritt das Gewicht jedes zweiten gewichteten Knotens ebenfalls 0. Diese Gewichte der neuen QF bezeichnen wir mit $\hat{\hat{b_i}}$. Wir betrachten nun analog den Fehler zur „guten“ QF:

$$\texttt{ERR}_2 := h \sum_{i=1}^{s} b_i f(x_0 + c_i h) - h \sum_{i=1}^{s} \hat{\hat{b_i}} f(x_0 + c_i h) \approx \hat{\hat{C}} h^{\hat{\hat{p}}+1}$$

Damit definieren wir

$$\texttt{err}[x_0, x_0 + h] = \texttt{ERR}_1 \cdot \left(\frac{\texttt{ERR}_1}{\texttt{ERR}_2}\right)^2$$

Wir sehen, daß die Konvergenz dieses Programms recht langsam erfolgt, z.B. wenn f eine Singularität aufweist.

1.8 Konvergenzbeschleunigung mit dem ε-Algorithmus

1.42 Beispiel
Wir wollen z.B. die Funktion $\int_0^1 \sqrt{x}\ln(x)\,dx = s$.

Das Programm TEGRAL(f,0,1,TOL) geht hier wie folgt vor (mit Gauß $s = 15$, $p = 30$):
Der Fehler im linken Intervall ist immer am größten und dadurch wird eben dieses Intervall immer geteilt.

Wenn $s_i - s$ der globale Fehler ist, stellt man fest, daß die Folge (S_i) sehr langsam gegen s konvergiert.

1.43 Frage
Wir möchten die Konvergenz beschleunigen. Wie können wir dies erreichen?

1.44 Beobachtung
Wir sehen, daß sich der Fehler ungefähr wie die geometrische Folge verhält:

$$S_{n+1} - S \approx \varrho(S_n - S) \quad \varrho \in \mathbb{R}$$

Im Beispiel gilt $\varrho \approx 0,37$ und wir sehen: $S_n \approx S + c\varrho^n$

1.45 Bemerkung
Ein ähnliches Verhalten tritt auch bei Fixpunktiterationen auf, d.h. $s_{n+1} = g(s_n)$ für eine diffbare Funktion g.
Dabei sucht man einen Fixpunkt, also $s = g(s)$.
Es gilt wegen des MWS:

$$s_{n+1} - s = g(s_n) - g(s) = g'(\xi_n)(s_n - s) \underbrace{\approx g'(s)}_{=:\varrho}(s_n - s)$$

1.46 Idee (von Aitken)
Man nimmt an, daß s_n, s_{n+1}, s_{n+2} bekannt sind. Dann bestimmt man s, ϱ so, daß folgendes gilt:

$$s_{n+1} - s = \varrho(s_n - s), \qquad s_{n+2} - s = \varrho(s_{n+1} - s)$$

Dabei setzt man $S'_n := s$
d.h. wir konstruieren aus der Folge (s_n) eine neue Folge (s'_n), die schneller konvergiert.

1.47 Konstruktion (Delta-Quadrat-Regel von Aitken)
Durch Subtraktion der Zeilen voneinander erhalten wir, indem wir die Beziehnung $\Delta s_n = s_{n+1} - s_n$ einführen:

$$s_{n+2} - s_{n+1} = \varrho(s_{n+1} - s_n) \iff \Delta s_{n+1} = \varrho \Delta s_n$$

Damit ergibt sich aus den obigen Gleichungen

$$s = s_{n+1} - \frac{\Delta s_{n+1}}{\varrho - 1}$$

Damit folgt: $s = s_{n+1} - \frac{\Delta s_{n+1} \Delta s_n}{\Delta s_{n+1} - \Delta s_n}$.
Dazu setzen wir nun $\Delta^2 s_n = \Delta(\Delta s_n) = \Delta s_{n+1} - \Delta s_n$
Durch die Umformungen erhalten wir die sog. Δ^2-Regel von Aitken:

$$S'_n = S_{n+1} - \frac{\Delta S_n \cdot \Delta S_{n+1}}{\Delta^2 S_n}$$

1.48 Bemerkung
Wir sehen, daß $(s_{n+1} - s) = \varrho_n(s_n - s)$ für $\beta_n \to \varrho \neq 1$.
Man kann zeigen (Beweis: Übungsblatt 3):

$$\frac{s'_n - s}{s_n - s} \to 0$$

1.49 Konstruktion (Erweiterung der Regel)
Konstruiere eine Folge s''_n, die das exakte Ergebnis s für Folgen s_n liefert für $\alpha_0, \alpha_1 \in \mathbb{R}$:

$$s_{n+2} - s = \alpha_1(s_{n+1} - s) + \alpha_0(s_n - s) \iff s_n - s = c_1 \varrho^n + c_2 \varrho_2^n$$

Analog dazu $s_n^{(k)}$ mit $s_n^{(k)} = s$ für folgen mit $k \geq 1$:

$$s_{n+k} - s = \alpha_{k-1}(s_{n+k-1} - s) + \ldots + \alpha_0(s_n - s)$$

Die $s_n^{(k)}$ lassen sich einfach berechnen.

1.50 Algorithmus (Epsilon-Algorithmus von Wynn, 1956)
Seien $s_0, s_1, s_2, \ldots$ gegeben und es gilt:

$$\varepsilon_{-1}^{(n)} = 0, \qquad \varepsilon_0^{(n)} = s_n$$

und

$$\varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + \frac{1}{e_k^{(n+1)} - \varepsilon_k^{(n)}}$$

Es gilt außerdem $\varepsilon_2^{(n)} = s'_n, \varepsilon_4^{(n)} = s''_n, \varepsilon_{2k}^{(n)} = s_k^{(n)}$

BEWEIS
Dies können wir leider nicht beweisen, weil es viel zu schwierig ist. Eine Andeutung zum Beweis:

$$s'_n = \frac{\det \begin{pmatrix} s_n & s_{n+1} \\ s_{n+1} & s_{n+2} \end{pmatrix}}{\det(\Delta^2 s_n)}$$

$$s''_n = \frac{\det \begin{pmatrix} s_n & s_{n+1} & s_{n+2} \\ s_{n+1} & s_{n+2} & s_{n+3} \\ s_{n+2} & s_{n+3} & s_{n+4} \end{pmatrix}}{\det \begin{pmatrix} \Delta^2 s_n & \Delta^2 s_{n+1} \\ \Delta^2 s_{n+1} & \Delta^2 s_{n+2} \end{pmatrix}}$$

usw.
Darin auftretende Matrizen nennt man HANKEL-Matrizen. Man nutzt im Beweis die Eigenschaften der Hankel-Matrix aus. □

BEWEIS
Der Nachweis von $\varepsilon_2^{(n)} = s'_n$ ist leicht zu berechnen:

$$\varepsilon_1^{(n)} = \underbrace{\varepsilon_{-1}^{(n+1)}}_{=0} + \frac{1}{\varepsilon_0^{(n+1)} - \varepsilon_0^{(n)}} = \frac{1}{\Delta s_n}$$

□

Dann ist

$$\begin{aligned} \varepsilon_2^{(n)} &= \varepsilon_0^{(n+1)} + \frac{1}{\varepsilon_1^{(n+1)} - \varepsilon_1^{(n)}} = s_{n+1} + \frac{1}{\frac{1}{\Delta s_{n+1}} - \frac{1}{\Delta s_n}} \\ &= s_{n+1} - \frac{\Delta s_n \cdot \Delta s_{n+1}}{\Delta s_{n+1} - \Delta s_n} = S_{n+1} - \frac{\Delta s_n \cdot \Delta s_{n+1}}{\Delta^2 s_n} = s'_n \end{aligned}$$

2 Interpolation und Approximation

2.1 Motivation
Wir haben folgende Probleme, die wir in diesem Kapitel hinreichend behandeln wollen:

1. Wir wollen zu n paarweise verschiedenen Punkten in $\mathbb{R}^2$ eine Funktion bzw. ein Polynom finden, d.h. wir haben

   $$(x_0, y_0), (x_1, y_1), \dots (x_n, y_n) \in \mathbb{R}^2$$

   und wollen $p(x_i) = y_i$
2. Zu einem gegeben $f : [a, b] \to \mathbb{R}$ sucht man einfach auszuwertende Funktionen (Polynome) $p : [a, b] \to \mathbb{R}$, sodaß $f - p$ klein sein soll.
   Dies kann man beispielsweise über die Supremumsnorm oder eine andere Norm vergleichen.

2.1 Newtonsche Interpolationsformel

2.2 Beispiel
Wir haben $(x_0, y_0), (x_1, y_1), (x_2, y_2) \in \mathbb{R}^2$ mit paarweise verschiedenen x.

Gesucht ist dann $p(x_i) = y_i$. Dazu betrachten wir folgenden Ansatz:

$$p(x) = \underbrace{y_0 + (x - x_0) \cdot \frac{y_1 - y_0}{x_1 - x_0}}_{\text{Gerade}} + \underbrace{a(x - x_0)(x - x_1)}_{=:\tilde{p}(x)}$$

mit $\tilde{p}(x_0) = 0 = \tilde{p}(x_1)$.

Wir wollen nun a so bestimmen, daß $p(x_2) = y_2$:

$$y_2 = y_0 + (x_2 - x_0)\left(\frac{y_1 - y_0}{x_1 - x_0}\right) + a(x_2 - x_0)(x_2 - x_1)$$

Damit ergibt sich

$$a = \frac{1}{x_2 - x_0}\left(\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}\right)$$

2.3 Definition (dividierte Differenzen)
Für $(x_0, y_0), (x_1, y_1), \ldots \in \mathbb{R}^2$ setzen wir:

$$\delta^0 y[x_0] = y_0, \qquad \delta^1 y[x_0, x_1] = \frac{\delta^0 y[x_1] - \delta^0 y[x_0]}{x_1 - x_0} = \frac{y_1 - y_0}{x_1 - x_0}$$

und allgemein definieren wir:

$$\delta^n y[x_0, \ldots, x_n] = \frac{\delta^{n-1} y[x_1, \ldots, x_n] - \delta^{n-1} y[x_0, \ldots, x_{n-1}]}{x_n - x_0}$$

2.4 Satz (Newtonsche Interpolationsformel, 1676)
Seien $(x_0, y_0), \ldots, (x_n, y_n) \in \mathbb{R}^2$ gegeben, x_i alle verschieden.
Dann gibt es genau ein Polynom p vom Grad $\leq n$, das diese Punkte interpoliert, d.h. $p(x_i) = y_i \quad \forall i \leq n$.
Außerdem erhält man das Polynom gegeben über

$$\begin{aligned} p(x) &= y_0 + (x - x_0)\delta^1 y[x_0, x_1] + (x - x_0)(x - x_1)\delta^2 y[x_0, x_1, x_2] + \ldots \\ &\quad + (x - x_0)(x - x_1)\ldots(x - x_{n-1})\delta^n y[x_0, \ldots, x_n] \\ &= \sum_{k=0}^{n} \left(\delta^k y[x_0, \ldots, x_k] \cdot \prod_{j=0}^{k-1} (x - x_j) \right) \end{aligned}$$

Beweis
Wir beweisen diesen Satz durch Induktion, wobei wir bereits gesehen haben, daß der Induktionsanfang gegeben ist.
Wir nehmen an, daß die Behauptung für $n - 1$ wahr ist und wir werden zeigen, daß dies auch für n wahr ist.
Dann ist $p_0(x)$ das Polynom, das durch die ersten $n - 1$ Punkte geht mit

$$p_0(x) = y_0 + (x - x_0)\delta^1 y[x_0, x_1] + \ldots + (x - x_0)(x - x_1)\ldots(x - x_{n-2})\delta^{n-1} y[x_0, \ldots, x_{n-1}]$$

Dann muß $p(x)$ von der Gestalt sein:

$$p(x) = p_0(x) + a \cdot (x - x_0) \cdot \ldots \cdot (x - x_{n-1})$$

Wir bestimmen das a so, daß auch $p(x_n) = y_n$ ist. Damit erhält man die Existenz und Eindeutigkeit von $p(x)$.
Wir müssen also zeigen, daß $a = \delta^n y[x_0, \ldots, x_n]$ gilt.
Sei dazu $p_1(x)$ das Polynom vom Grad $\leq n - 1$ durch $(x_1, y_1), \ldots, (x_n, y_n)$:

$$\begin{aligned} p_1(x) &= y_1 + (x - x_1)\delta y[x_1, x_2] + \ldots + (x - x_1)\ldots(x - x_{n-1})\delta^{n-1} y[x_1, \ldots, x_n] \\ &= x^{n-1} \cdot \delta^{n-1} y[x_1, \ldots, x_n] + \text{Rest} \end{aligned}$$

Dann ist

$$p(x) = \frac{x_n - x}{x_n - x_0} p_0(x) + \frac{x - x_0}{x_n - x_0} p_1(x)$$

ein Polynom vom Grad $\leq n$, das durch alle Punkte geht, was wir nachrechnen können:

$$p(x_0) = \underbrace{\frac{x_n - x_0}{x_n - x_0}}_{=1} p_0(x_0) + \underbrace{\frac{x_0 - x_0}{x_n - x_0}}_{=0} p_1(x_0) = y_0$$
$$p(x_n) = \underbrace{\frac{x_n - x_n}{x_n - x_0}}_{=0} p_0(x_n) + \underbrace{\frac{x_n - x_0}{x_n - x_0}}_{=1} p_1(x_n) = y_n$$

Für die mittleren Punkte gilt:

$$p(x_i) = \frac{x_n - x_i}{x_n - x_0} p_0(x_i) + \frac{x_i - x_0}{x_n - x_0} p_1(x_i) = 1 \cdot y_i = y_i$$

Wir möchten uns nun im Polynom $p(x)$ den Koeffizienten von x^n anschauen:

$$a = -\frac{1}{x_n - x_0} \delta^{n-1} y[x_0, \ldots, x_{n-1}] + \frac{1}{x_n - x_0} \delta^{n-1} y[x_1, \ldots, x_n] = \delta^n y[x_0, \ldots, x_n]$$ □

2.5 Beispiel
Herr Briggs berechnete gerne Logarithmen und hatte bereits folgende Werte berechnet:

1. $\log_{10} 55 = 1,7403627$
2. $\log_{10} 56 = 1,7481880$
3. $\log_{10} 57 = 1,7558749$
4. $\log_{10} 58 = 1,7634280$

Wir möchten die dividierte Differenzen berechnen:

1. $0,0078253$
2. $0,0076869$
3. $0,0075531$

In der zweiten Runde ergeben sich folgende dividierte Differenzen:

1. $-0,0000692$
2. $-0,0000669$

Und als letzte dividierte Differenz ergibt sich: $0,0000008$, womit wir folgende Interpolationsformel erhalten:

$$\begin{aligned} p(x) = {} & 1,7403627 + (x - 55)0,0078253 \\ & + (x - 55)(x - 56)(-0,0000692) \\ & + (x - 55)(x - 56)(x - 57)0,0000008 \end{aligned}$$

2.6 Bemerkung (Geschickte Berechnung mit dem Horner-Schema)
Zur geschickten Berechnung des Interpolationspolynoms $p(x)$ benutzen wir das sog. HORNER-Schema:

$$p(x) = y_0 + (x - x_0)\Big(\delta y[x_0, x_1] + (x - x_1)(\delta^2 y[x_0, x_1, x_2] + (x - x_2)\left(\ldots + (x - x_{n-1})\delta^n y[x_0, \ldots, x_n]\right)\Big)$$

Was zum folgenden Algorithmus führt:

2.7 Algorithmus
Setze

$$s := \delta^n y[x_0, \ldots, x_n]$$

und für $i \leq n$:

$$s := \delta^{n-i} y[x_0, \ldots, x_{n-i}] + s \cdot (x - x_{n-i})$$

und $p(x) := s$

2.2 Fehler bei der Polynominterpolation

2.8 Motivation
Wir wollen folgendes Problem untersuchen:
Wir haben eine Funktion $f : [a, b] \to \mathbb{R}$ und interpolieren f in $x_0, \ldots, x_n \in [a, b]$, wobei $y_i = f(x_i)$.
Für das Interpolationspolynom (IPP) p gilt: $p(x_i) = f(x_i)$ für $i \leq n$.
Was läßt sich über den Interpolationsfehler $f(x) - p(x)$ für $x \in [a, b]$ sagen?

2.9 Hilfssatz (Quasi-Mittelwertsatz)
Sei f k-mal diffbar, dann bezeichnen wir $\delta^k f[x_0, \ldots, x_k] = \delta^k y[x_0, \ldots, x_k]$ mit $y_i = f(x_i)$. Dann gilt:

$$\exists \xi \in (\min_i x_i, \max_i x_i) : \quad \delta^k f[x_0, \ldots, x_k] = \frac{f^{(k)}(\xi)}{k!}$$

BEWEIS
Setze $d(x) = f(x) - p(x)$, wobei p IPP zu f vom Grad $\leq k$ ist.
Wir sehen sofort, daß $d(x_i) = 0$ für $i \leq k$.

Nach dem MWS gilt

$$\exists \xi_0, \xi_1, \ldots, \xi_{k-1} \quad d'(\xi_i) = 0$$

Weiter finden wir wieder mit der Hilfe des MWS

$$\exists \eta_0, \eta_1, \ldots, \eta_{k-1} \quad d''(\eta_i) = 0$$

Dieses Verfahren iteriert man solange, bis man erhält:

$$\exists \xi \quad d^{(k)}(\xi) = 0$$

wobei $f^{(k)}(\xi) = p^{(k)}(\xi) = k!\delta^k f[x_0, \ldots, x_k]$. □

2.10 Bemerkung
für $k = 1$ ergibt sich der bereits bekannte Mittelwertsatz:

$$\frac{f(x_1) - f(x_0)}{x_1 - x_0} = f'(\xi)$$

2.11 Satz (Fehler beim Interpolationspolynom)
Sei $f : [a, b] \to \mathbb{R} \in \mathcal{C}^{n+1}([a, b])$ und sei p das IPP zu f in $x_0, \ldots, x_n \in [a, b]$, wobei $\deg p \leq n$.
$\forall x \in [a, b]\, \exists \xi = \xi(x) \in (a, b)$:

$$f(x) - p(x) = (x - x_0) \cdot \ldots \cdot (x - x_n) \frac{f^{(n+1)}(\xi)}{(n+1)!}$$

Beweis
Sei $\bar{x} \in [a, b]$ beliebig. Zeige, daß die Behauptung für $\bar{x}$ gilt:
Falls $\bar{x} = x_i$ ist, ist die Formel offensichtlich richtig. Sei daher im folgenden $\forall i \leq n \quad \bar{x} \neq x_i$
Sei $\bar{p}$ das IPP zu $x_0, \ldots, x_n, \bar{x}$: $\bar{p}(x_i) = f(x_i)$ und $\bar{p}(\bar{x}) = f(\bar{x})$.
Nach der Newtonschen Interpolationsformel ergibt sich:

$$\bar{p}(x) = p(x) + (x - x_0) \ldots (x - x_n) \cdot \delta^{n+1} f[x_0, \ldots, x_n, \bar{x}]$$

Nach dem Hilfssatz wissen wir, daß

$$\delta^k f[x_0, \ldots, x_k] = \frac{f^{(k)}(\xi)}{k!}$$

gilt, womit gilt:

$$\bar{p}(\bar{x}) = p(\bar{x}) + (\bar{x} - x_0) \cdot \ldots \cdot (\bar{x} - x_n) \frac{f^{(n+1)}(\xi)}{(n+1)!}$$

womit wir durch Umstellung der Gleichung mit dem Beweis fertig sind. □

2.12 Beispiel
Wir greifen auf das Beispiel des vorgehenden Kapitels zurück. Dabei betrachteten wir

$$f(x) = \log_{10} x = \frac{\ln x}{\ln 10}$$

wobei $[a, b] = [55, 58]$. Es gilt außerdem:

$$f'(x) = \frac{1}{x \ln 10}, \qquad f''(x) = -\frac{1}{x^2 \ln 10}, \qquad f'''(x) = \frac{2}{x^3 \ln 10}, \qquad f^{(4)} = -\frac{6}{x^4 \ln 10}$$

Für $x \in [55, 58]$ ist damit

$$\left| f^{(4)}(x) \right| \leq \frac{6}{55^4 \cdot \ln 10}$$

Damit ist

$$|\log_{10}(56,5) - p(56,5)| \leq 1,5 \cdot 0,5 \cdot 0,5 \cdot 1,5 \cdot \frac{6}{55^4 \ln 10} \cdot \frac{1}{4!} \approx 7 \cdot 10^{-9}$$

2.13 Frage (Ausblick)
Wir sehen, daß der Fehler im Wesentlichen von der Funktion sowie von der Wahl der Punkte ab.
Wir fragen uns, wie wir die Punkte wählen müssen, damit der Fehler möglichst klein wird, d.h.

$$\max_{x \in [a,b]} |(x - x_0) \dots (x - x_n)|$$

möglichst klein wird.

Dies führt zur sog. Tschebyscheff-Interpolation, die wir im nächsten Kapitel behandeln werden.

Wir stehen vor einem weiteren Problem:
Wir erhalten für ein Paar nur „gestörte“ Werte: $(x, \tilde{y_i})$. Damit erhalten wir ein neues IPP $\tilde{p}(x)$, d.h. wir wollen $p(x) - \tilde{p}(x)$ berechnen.

2.14 Satz (Lagrange-Interpolationsformel)
Das IPP p durch (x_i, y_i) mit $i = 0, 1, \dots, n$ ist gegeben durch:

$$p(x) = \sum_{i=0}^{n} y_i \ell_i(x), \qquad \ell_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$$

Dabei hat das Lagrange-Polynom ℓ_i die Eigenschaft $\ell_i(x_j) = \begin{cases} 1, & i = j \\ 0, & \text{sonst} \end{cases}$ und $\deg \ell_i = n$

Beweis
Setze $q(x) := \sum_{i=0}^{n} y_i \ell_i(x)$ mit $\deg q \leq n$

$$q(x_k) = \sum_{i=0}^{n} y_i \ell_i(x_k) = y_k$$

Damit ist das q ein IPP vom Grad $\leq n$ und wegen der Eindeutigkeit des IPP gilt: $p = q$. □

2.15 Bemerkung
Wenn $p(x) = \sum_{i=0}^{n} y_i \ell_i(x)$ das richtige IPP ist und $\tilde{p}(x) = \sum_{i=0}^{n} \tilde{y_i} \ell_i(x)$ das IPP für die „gestörten“ Werte.

Wir möchten den Fehler genauer bestimmen und erhalten über die Dreiecksungleichung:

$$|p(x) - \tilde{p}(x)| \leq \sum_{i=0}^{n} |(y_i - \tilde{y_i})| \cdot |\ell_i(x)| \leq \sum_{i=0}^{n} |\ell_i(x)| \cdot \max_{i=0,\ldots,n} |y_i - \tilde{y_i}|$$

2.16 Satz (Lebesgue-Konstante)
Wir bezeichnen mit

$$\Lambda_n := \max_{x \in [a,b]} \sum_{i=0}^{n} |\ell_i(x)|$$

die *Lebesgue-Konstante* zu den Knoten $x_0, \ldots, x_n$ und zum Intervall $[a, b]$ und es gilt:

$$\max_{x \in [a,b]} |p(x) - \tilde{p}(x)| \leq \Lambda_n \cdot \max_{i=0,\ldots,n} |y_i - \tilde{y_i}|$$

Dabei ist Λ_n die kleinstmögliche Konstante in einer solchen Ungleichung.

2.17 Beispiel
Wir betrachten das Intervall $[-1, 1]$ mit einer äquidistanzen Unterteilung, d,h. $x_i = -1 + \frac{2i}{n}$
Wir möchten nun die Lesbegue-Konstanten berechnen:

$$\Lambda_{10} = 40, \qquad \Lambda_{20} \approx 30000, \qquad \Lambda_{40} \approx 10^{10}$$

Das Problem ist, daß das Lagrange-Polynom sehr stark zwischen den Knoten schwankt.
Wir können eine Näherung für große n für die Lesbegue-Konstante angeben:

$$\Lambda_n \approx \frac{2^{n+1}}{e \cdot n \cdot \log n}$$

d.h. die Konstante wächst exponentiell mit n an, d.h. wir müssen bei Polynominterpolation hohen Grades vorsichtig sein.

2.18 Bemerkung
Wir werden später ein Verfahren kennenlernen, bei dem $\Lambda_n \leq 4$ für $n \leq 100$ ist und das uns die Knoten dafür geschickt wählt.

2.19 Frage
Wir fragen uns, ob die Polynominterpolation zur Approximation von Funktionen überhaupt geeignet ist.

2.20 Satz (Bestapproximationsfehler)
Sei $f : [a, b] \to \mathbb{R}$ eine bel. Funktion.
Wenn p ein IPP zu f in $x_0, \ldots, x_n \in [a, b]$ ist, dann gilt:

$$\max_{x\in[a,b]} |f(x) - p(x)| \leq (\Lambda_n + 1) \inf_{\deg q \leq n} \left(\max_{x\in[a,b]} |f(x) - q(x)| \right)$$

Dabei stellt die linke Seite den Interpolationsfehler dar und die rechte Seite den *Bestapproximationsfehler*.

Beweis
Sei q ein bel. Polynom vom Grad $\leq n$. Betrachte

$$f - p = (f - q) + (q - p)$$

Wir wissen, daß q IPP zu sich selbst ist in $x_0, \ldots, x_n$.
Wir wissen, daß wenn $y_i = f(x_i)$ und $\tilde{y}_i = q(x_i)$ gilt:

$$\max_{x\in[a,b]} |p(x) - q(x)| \leq \Lambda_n \max_{i=0,\ldots,n} |f(x_i) - q(x_i)| \leq \Lambda_n \max_{x\in[a,b]} |f(x) - q(x)|$$

Somit ergibt sich mit der Dreiecksungleichung:

$$\max_{x\in[a,b]} |f(x) - p(x)| \leq (1 + \Lambda_n) \max_{x\in[a,b]} |f(x) - q(x)|$$

□

2.3 Tschebyscheff-Interpolation

2.21 Motivation
Wir wollen im Folgenden $f : [a, b] \to \mathbb{R}$ durch Polynominterpolation mit „günstigen“ Stützstellen betrachten.
Dazu betrachten wir o.B.d.A. statt $[a, b]$ das Intervall $[-1, 1]$.

2.22 Definition (Tschebyscheff-Polynom)
Wir definieren als n-tes *Tschebyscheff-Polynom*:

$$T_n(x) = \cos(n \arccos(x)), \qquad x \in [-1, 1]$$

2.23 Hilfssatz (Eigenschaften der Tschebyscheff-Polynome)
Das n-te Tschebyscheff-Polynom ist ein Polynom vom Grad n mit folgenden Eigenschaften:

1. $T_0(x) = 1, T_1(x) = x$ und allgemein gilt für $n \geq 1$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \qquad T_n(x) = 2^{n-1}x^n + \mathcal{O}(x^{n-1})$$
2. $|T_n(x)| \leq 1$ und für $k \in \mathbb{N}$ gilt:
$$T_n\left(\cos\frac{k\pi}{n}\right) = \cos(k\pi) = (-1)^k$$
3. Für $k \in \mathbb{N}$ gilt:
$$T_n\left(\cos\frac{(2k+1)\pi}{2n}\right) = \cos\left(\frac{(2k+1)\pi}{2}\right) = 0$$

Beweis
1. Sei $x = \cos\varphi$. Dann ist $\cos((n+1)\varphi) + \cos((n-1)\varphi) = 2\cos\varphi \cdot \cos(n\varphi)$, womit die Rekursion gezeigt ist und man damit sieht, daß dies Polynome sind.
2. Betrachte Beschränktheit des Kosinus bzw. Extrempunktverhalten
3. klar □

2.24 Beispiel
Wir erhalten folgende Polynome:

1. $T_2(x) = 2x^2 - 1$
2. $T_3(x) = 2x(2x^2 - 1) - x = 4x^3 - 3x$
3. $T_4(x) = 8x^4 - 8x^2 - 1$

2.25 Hilfssatz (Abschätzung der Maximalwerte)
Sei q ein Polynom vom Grad n, $q(x) = 2^{n-1}x^n + \ldots$ und $q \neq T_n$. Dann gilt:

$$\max_{x\in[-1,1]} |q(x)| > \max_{x\in[-1,1]} |T_n(x)| = 1$$

Beweis
Angenommen, es gilt $|q(x)| \leq 1$. Wir wissen, daß $T_n(1) = 1$ und $T_n\left(\cos\frac{\pi}{n}\right) = -1$ ist.
Nach dem ZWS hat $q - T_n$ eine Nullstelle in $[\cos\frac{\pi}{n}, 1]$. Falls $\cos\frac{\pi}{n}$ eine Nullstelle ist, dann ist sie eine doppelte. Ebenso muß es in $[\cos\frac{2\pi}{n}, \cos\pi n]$ eine Nullstelle geben.
Diesen Gedankengang kann man fortsetzen.
Also hat $q - T_n$ n Nullstellen in $[-1, 1]$. Andererseits wissen wir, daß $\deg(q - T_n) \leq n - 1$, also muß $q = T_n$ sein, was aber im Widerspruch steht zu $q \neq T_n$. □

2.26 Satz (Nullstellen des Tschebyscheff-Polynoms)
Unter allen $(x_0, \ldots, x_n) \in \mathbb{R}^{n+1}$ wird

$$\max_{x \in [-1,1]} |(x - x_0) \cdot \ldots \cdot (x - x_n)|$$

minimal und zwar $\frac{1}{2^n}$ für $x_k = \cos\left(\frac{2k+1}{n+1}\frac{\pi}{2}\right)$ und die x_k sind die Nullstellen von T_{n+1}

Beweis
Die Behauptung ergibt sich unmittelbar durch den vorgehenden Hilfssatz durch Multiplikation mit 2^{n+1} und Anwendung der Eigenschaften eines Tschebyscheff-Polynoms. □

2.27 Satz (Lebesgue-Konstante beim Tschebyscheff-Polynom)
Die Lebesgue-Konstanten zu den Tschebyscheff-Knoten $x_k = \cos\left(\frac{2k+1}{n+1}\frac{\pi}{2}\right)$ für $k = 0, \ldots, n$ zum Intervall $[-1, 1]$ erfüllen:

$$\begin{aligned} \Lambda_n &\leq 3 && n \leq 20 \\ \Lambda_n &\leq 4 && n \leq 100 \\ \Lambda_n &\approx \frac{2}{\pi} \log n && n \to \infty \end{aligned}$$

Dies ist ein sehr viel besserer Fehler als bei der äquidistanten Unterteilung.

Beweis
Direkte Rechnung, die Approximation wird nicht bewiesen. □

2.28 Bemerkung
Die Tschebyscheff-Knoten liefern eine *fast-optimale* Polynomapproximation an f.
Wir werden im Folgenden noch weitere Eigenschaften sehen, die die Berechnung von Interpolationspolynomen zu Tschebyscheff-Knoten vereinfachen.

2.29 Hilfssatz (Orthogonalitätsrelation)
Die Tschebyscheff-Polynome sind orthogonal bezüglich des Skalarprodukts:

$$\langle f, g \rangle = \int_{-1}^{1} \frac{1}{\sqrt{1 - x^2}} f(x) g(x)\, dx$$

Beweis
Es gilt:

$$\begin{aligned}\langle T_k, T_j\rangle &= \int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} T_k(x) T_j(x)\, dx \\ &= -\int_{\pi}^{0} \cos(k\varphi)\cos(j\varphi)\, d\varphi\end{aligned}$$

mit Additionstheoremen:

$$\begin{aligned}&= \int_{0}^{\pi} \frac{1}{2}(\cos(k-j)\varphi + \cos(k+j)\varphi)\, d\varphi \\ &= \begin{cases} 0, & k \neq j \\ \frac{\pi}{2}, & k = j \neq 0 \\ \pi, & k = j = 0 \end{cases}\end{aligned}$$

womit die Behauptung gezeigt ist. □

2.30 Satz (Orthogonalität der Tschebyscheff-Polynome)
Die Tschbyscheff-Polynome $T_0, \ldots, T_n$ sind orthogonal aufeinander bezüglich des Skalarprodukts auf dem Vektorraum der Polynome vom Grad $\leq n$:

$$\langle f, g\rangle = \sum_{l=0}^{n} f(x_l) g(x_l)$$

wobei $x_0, \ldots, x_n$ die Nullstellen von T_{n+1} sind.

Beweis
Daß es sich dabei um ein Skalarprodukt handelt, ist klar. Betrachte nun

$$\langle T_k, T_j\rangle = \sum_{l=0}^{n} T_k(x_l) T_j(x_l), \qquad x_l = \cos\left(\frac{2l+1}{n+1}\frac{\pi}{2}\right)$$

Wegen der rekursiven Formel gilt:

$$T_k(x_l) = \cos(k \cdot \arccos x_l) = \cos\left(k\left(l+\frac{1}{2}\right)h\right), \qquad h := \frac{\pi}{n+1}$$

Damit ergibt sich dann für das Skalarprodukt:

$$\begin{aligned}
\langle T_k, T_j \rangle &= \sum_{l=0}^{n} \cos\left(k\left(l+\frac{1}{2}\right)h\right)\cos\left(j\left(l+\frac{1}{2}\right)h\right) \\
&= \frac{1}{2}\sum_{l=0}^{n}\left(\cos\left((k-j)\left(l+\frac{1}{2}\right)h\right)+\cos\left((k+j)\left(l+\frac{1}{2}\right)h\right)\right) \\
&= \frac{1}{2}\,\mathrm{Re}\left(\sum_{l=0}^{n}\exp(i(k-j)\left(l+\frac{1}{2}\right)h)+\sum_{l=0}^{n}\exp(i(k+j)\left(l+\frac{1}{2}\right)h)\right) \\
&= \begin{cases} 0, & k \neq j \\ \frac{1}{2}(n+1), & k=j\neq 0 \\ n+1, & k=j=0 \end{cases}
\end{aligned}$$

womit die Behauptung gezeigt ist. □

2.31 Satz (Aussehen der IPP mit Tschebyscheff-Knoten)
Das IPP zu f in den Tschebyscheff-Knoten $x_0, \ldots, x_n$ (Nullstellen von T_{n+1}) ist gegeben durch

$$p(x) = \frac{1}{2}c_0 + c_1 T_1(x) + \ldots + c_n T_n(x)$$

wobei man die Koeffiezienten leicht über die folgende Formel berechnen kann:

$$c_k = \frac{2}{n+1}\sum_{l=0}^{n} f\underbrace{\left(\cos\left(\frac{2l+1}{n+1}\frac{\pi}{2}\right)\right)}_{=x_l}\cos\left(k\frac{2l+1}{n+1}\frac{\pi}{2}\right)$$

Beweis
Betrachte das Skalarprodukt von p und T_k:

$$\langle p, T_k\rangle = \frac{1}{2}c_0\langle T_0, T_k\rangle + c_1\langle T_1, T_k\rangle + \ldots + c_k\langle T_n, T_k\rangle$$

wegen Orthogonalität gilt

$$\langle p, T_k\rangle = c_k\frac{n+1}{2}$$

und damit:

$$\begin{aligned} c_k = \frac{2}{n+1}\langle p, T_k\rangle &= \frac{2}{n+1}\sum_{l=0}^{n} p(x_l)T_k(x_l) \\ &= \frac{2}{n+1}\sum_{l=0}^{n} f(x_l)T_k(x_l) \\ &= \frac{2}{n+1}\sum_{l=0}^{n} f(x_l)\cos\left(k\frac{2l+1}{n+1}\frac{\pi}{2}\right) \end{aligned}$$

womit die Behauptung folgt. □

2.32 Satz (Clenshaw-Algorithmus)
Sei p ein bel. Polynom vom Grad $\leq n$ für das gilt:

$$p(x) = \frac{1}{2}c_0 + c_1T_1(x) + \ldots + c_nT_n(x)$$

Sei $d_{n+2} = d_{n+1} = 0$. Setze dann

$$d_k = c_k + 2x \cdot d_{k+1} - d_{k+2} \quad \text{für } k = n, n-1, \ldots, 0$$

Dann ist

$$p(x) = \frac{1}{2}(d_0 - d_2)$$

Beweis
Wir verwenden für den Beweis die Rekursionsformel für die Tschebyscheff-Polynome:

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x)$$

und damit

$$p(x) = \frac{1}{2}c_0 + c_1T_1(x) + \ldots + \underbrace{c_n}_{=d_n}T_n(x)$$

und für $k = n-1$ ergibt sich:

$$p(x) = \frac{1}{2}c_0 + c_1T_1(x) + \ldots + \underbrace{(c_{n-1} + 2x \cdot d_n)}_{=d_{n-1}}T_{n-1}(x)$$

und für $k = n-2$ ergibt sich:

$$p(x) = \frac{1}{2}c_0 + c_1 T_1(x) + \ldots + (c_{n-3} - d_{n-1})T_{n-3}(x) + \underbrace{(c_{n-2} - d_n + 2xd_{n-1})}_{=d_{n-2}} T_{n-2}(x)$$

Dieses Verfahren wiederholen wir solange, bis wir auf die folgende Gleichung kommen:

$$\begin{aligned} p(x) &= \left(\frac{1}{2}c_0 - d_2\right) + \underbrace{c_1 - d_3 + 2xd_2}_{=d_1}\underbrace{T_1(x)}_{=x} \\ &= \frac{1}{2}c_0 - d_2 + xd_1 \\ &= \frac{1}{2}\left(\underbrace{(c_0 + 2xd_1 - d_2)}_{=d_0} - d_2\right) \\ &= \frac{1}{2}(d_0 - d_2) \end{aligned}$$

womit die Behauptung per Induktion folgt. □

2.33 Bemerkung (Allgemeine Bemerkung über Rekursionen)
Bei der Verwendung von Rekursionen ist es wichtig, wie sich Fehler (z.B. Rundungsfehler) fortpflanzen, z.B.

Sei $x_{n+1} = 10x_n - 9$ mit $x_0 = 1$, damit ist $x_n = 1\ \forall n$. Hätte man nun als Anfangswert $\tilde{x_0} = 1 + \varepsilon$, dann wäre $\tilde{x_n} = 1 + 10^n\varepsilon$.

Dies wäre eine *instabile* Rekursion.

2.34 Hilfssatz (Stabilität des Clenshaw-Algorithmus)
Betrachte zum Clanshaw-Algorithmus gestörte Rekursion

$$\tilde{d_k} = c_k + 2x\tilde{d}_{k+1} - \tilde{d}_{k+2} + \varepsilon_k$$

wobei die ε_k Rundungsfehler im k-ten Schritt sind.
und wir betrachten als Startwerte $\tilde{d}_{n+1} = \tilde{d}_{n+2} = 0$ und wir setzen $\tilde{p}(x) = \frac{1}{2}(\tilde{d}_0 - \tilde{d}_2)$.
Dann gilt:

$$|\tilde{p}(x) - p(x)| \leq \sum_{j=0}^{n} |\varepsilon_j|$$

Beweis
Setze $e_k = \tilde{d}_k - d_k$ wegen Lineartität der Rekursion gilt:

$$e_k = \varepsilon_k + 2xe_{k+1} - e_{k+2}$$

mit $e_{n+2} = e_{n+1} = 0$. Somit gilt nach dem Clenshaw-Algorithmus:

$$\frac{1}{2}(e_0 - e_2) = \frac{1}{2}\varepsilon_0 + \varepsilon_1 T_1(x) + \ldots + \varepsilon_n T_n(x)$$

Damit ergibt sich, daß

$$\frac{1}{2}(e_0 - e_2) = \frac{1}{2}(\tilde{d}_0 - \tilde{d}_2) - \frac{1}{2}(d_0 - d_2) = \tilde{p}(x) - p(x)$$

wegen $|T_k(x)| \leq 1$ für $|x| \leq 1$ erhalten wir:

$$|\tilde{p}(x) - p(x)| \leq \frac{1}{2}|\varepsilon_0| + |\varepsilon_1| + \ldots + |\varepsilon_n|$$

□

2.35 Bemerkung
Approximationen durch Linearkombinationen von Tschebyscheff-Polynomen wird im Taschenrechner zur Berechnung von Funktionen wie der Logarithmus, Exponentialfunktion, Sinus und Kosinus verwendet.

2.36 Beispiel (zur Approximation im Taschenrechner)
Wir wollen im Taschenrechner den Logarithmus berechnen.
Im Taschenrechner liegen alle darstellbaren Zahlen zwischen $0 < x_{\min} \leq x \leq x_{\max}$.
Der Taschenrechner arbeitet mit der Gleitpunktdarstellung: $x = [1b_1b_2 \ldots b_M]_2 \cdot 2^N$.
Der Ausdruck im Binärsystem ist dabei $1+b_1 2^{-1}+b_2 2^{-2}+\ldots+b_M 2^{-M}$ die sog. *Mantisse* und N der Exponent. Daher können wir auch im „normalen“ Zahlensystem auch $x = (1+t) \cdot 2^N$ schreiben.

Wir können daher den Logarithmus folgendermaßen darstellen:

$$\log x = \log(1+t) + N \cdot \log 2$$

wobei wir $t \in [0,1]$ beschränken können.
Durch die Tschebyscheff-Approximation erhalten wir:

$$[-1,1] \to [0,1] : x \mapsto t = \frac{1+x}{2} \iff x = 2t - 1$$

Der Interpolationsfehler ergibt sich dann durch

$$\log(1 + \frac{1+x}{2}) - p(x) = (x - x_0) \ldots (x - x_n) \frac{1}{(n+1)!} \frac{(-1)^{n-1}(n-1)!}{\left(1 + \frac{1+\xi}{2}\right)^n} \left(\frac{1}{2}\right)^n$$

und damit

$$|\log(1 + \frac{1+x}{2}) - p(x)| \leq 2^{-n} \cdot \frac{1}{(n+1)!} \cdot (n-1)! \cdot 2^{-n} = \frac{1}{(n+1)n} 4^{-n}$$

Es ergibt sich z.B. dadurch für $n = 15$, daß der Fehler kleiner als 10^{-11} wird.
Die Koeffizienten für die c_k erhält man nach dem vorgehenden Satz und wir sehen (was sogar allgemein für glatte Funktionen gilt), daß diese Koeffizienten c_k sehr rasch gegen 0 gehen.

Um also eine Genauigkeit von 10^{-8} zu erzielen, braucht man $c_0, \ldots, c_9$. Zur Auswertung des IPP mit dem Clenshaw-Algorithmus sind etwa 10 Multiplikationen notwendig.

2.37 Bemerkung (Vergleich mit der Taylorreihe)
Wir wollen die Taylorreihe mit dem Tschebyscheff-Polynom vergleichen und sehen, daß bei der Taylorentwicklung

$$\log(1+t) = \sum_{k=1}^{\infty} \frac{(-1)^k}{k} t^k$$

die Koeffizienten sehr langsam abfallen. Damit gibt sich für die Genauigkeit von 10^{-8} etwa 10^8 Terme.

2.4 Hermite-Interpolation

2.38 Motivation
Wir betrachten nun folgendes Problem:
Wir haben (x_i, y_i, y_i') gegeben für $i = 0, 1, \ldots, n$ und möchten ein IPP finden, für das gilt

1. $p(x_i) = y_i$
2. $p'(x_i) = y_i'$

2.39 Idee
Dazu betrachten wir folgende Idee:
Wir bilden die dividierten Differenzen aus folgenden Werten: (x_0, y_0) und $(x_0 + \varepsilon, y_0 + \varepsilon y_0')$ (x_1, y_1) und $(x_1 + \varepsilon, y_1 + \varepsilon y_1')$ usw.
Danach werden wir $\varepsilon \to 0$ laufen lassen.

Mit der Newtonschen Interpolationsformel erhalten wir folgendes IPP:

$$\begin{aligned} p_\varepsilon(x) = {} & y_0 + (x - x_0) y_0' + (x - x_0)(x - x_0 - \varepsilon)\delta^2[x_0, x_0 + \varepsilon, x_1] + \ldots \\ & + (x - x_0)(x - x_0 - \varepsilon)\ldots \\ & \cdot \ldots (x - x_{n-1})(x - x_{n-1} - \varepsilon)\delta^{2n} y[x_0, x_0 + \varepsilon, \ldots, x_{n-1}, x_{n-1} + \varepsilon, x_n] \\ & + (x - x_0)(x - x_0 - \varepsilon)\ldots(x - x_{n-1})(x - x_{n-1} - \varepsilon) \\ & \cdot (x - x_n)\delta y^{2n+1} y[x_0, x_0 + \varepsilon, \ldots, x_n, x_n + \varepsilon] \end{aligned}$$

Wir definieren $\delta^2 y[x_0, x_0, x_1] = \lim\limits_{\varepsilon \to 0} \delta^2 y[x_0, x_0 + \varepsilon, x_1]$ entsprechend $\delta^3 y[x_0, x_0, x_1, x_1]$, usw.
Wir setzen nun $p(x) = \lim\limits_{\varepsilon \to 0} p_\varepsilon(x)$ und erhalten damit:

$$\begin{aligned} p(x) = {} & y_0 + (x - x_0) y_0' + (x - x_0)(x - x_0)\delta^2[x_0, x_0, x_1] + \ldots \\ & + (x - x_0)(x - x_0)\ldots(x - x_{n-1})(x - x_{n-1})\delta^{2n} y[x_0, x_0, \ldots, x_{n-1}, x_{n-1}, x_n] \\ & + (x - x_0)^2 \ldots (x - x_{n-1})^2 (x - x_n)\delta^{2n+1} y[x_0, x_0, \ldots, x_n, x_n] \end{aligned}$$

Wir erhalten damit ein Polynom vom Grad $\leq 2n+1$ und müssen noch zeigen, daß $p(x_i) = y_i$ und $p'(x_i) = y_i'$ gilt.
Es ist außerdem klar, daß

$$p(x_i) = \lim_{\varepsilon \to 0} p_\varepsilon(x_i) = y_i$$

und

$$p_\varepsilon(x_i + \varepsilon) = y_i + \varepsilon y_i', \qquad p_\varepsilon(x_i) = y_i$$

Damit ist

$$y_i' = \frac{p_\varepsilon(x_i + \varepsilon) - p_\varepsilon(x_i)}{\varepsilon} = p_\varepsilon'(x_i + \varepsilon\xi_\varepsilon)$$

wobei sich das zweite Gleichheitszeichen nach dem MWS ergibt und daraus ergibt sich, daß

$$p'(x_i) = \lim_{\varepsilon \to 0} p_\varepsilon'(x_i + \varepsilon\xi_\varepsilon) = y_i'$$

2.40 Satz (Eindeutigkeit der Hermiteschen Polynome)
Die Hermiteschen Polynome sind eindeutig.

Beweis
Sei q ein weiteres Polynom vom Grad $\leq 2n+1$ mit $q(x_i) = y_i$ und $q'(x_i) = y_i'$.
Also ist $p - q$ ein Polynom vom Grad $\leq 2n+1$ mit doppelten Nullstellen x_i:

$$p(x) - q(x) = c \cdot \prod_{i=0}^{n} (x - x_i)^2$$

Der linke Term hat dabei den Grad $\leq 2n+1$ und der rechte hat den Grad $2n+2$. Also ist $c = 0$ und damit $p = q$. □

2.41 Satz (Hermite-Interpolation)
Zu (x_i, y_i, y_i') für $i = 0, \ldots, n$ für verschiedene x_i existiert ein eindeutiges Polynom vom Grad $\leq 2n+1$ mit $p(x_i) = y_i$ und $p'(x_i) = y_i'$ für $i = 0, \ldots, n$.

Es kann mithilfe der Newton'schen Interpolationsformel mit doppelt geschriebenen Knoten berechnet werden:

$$\begin{aligned} p(x) = {} & y_0 + (x - x_0)y_0' + (x - x_0)^2 \delta^2 y[x_0, x_0, x_1] + \ldots \\ & + (x - x_0)^2 \cdot \ldots \cdot (x - x_{n-1})^2 \cdot (x - x_n)\delta^{2n+1} y[x_0, x_0, \ldots, x_n, x_n] \end{aligned}$$

2.42 Satz (Fehler bei Hermite-Interpolation)
Sei $f : [a, b] \to \mathbb{R}$ mind. $(2n+2)$-mal stetig diffbar und $x_0, \ldots, x_n \in [a, b]$ alle verschieden und p das Hermite IPP zu $(x_i, f(x_i), f'(x_i))$. Dann kann man den Fehler folgendermaßen abschätzen:

$$\forall x \in [a,b] \quad \exists \xi = \xi(x) \in (a,b) \quad f(x) - p(x) = (x - x_0)^2 \ldots (x - x_n)^2 \frac{f^{(2n+2)}(\xi)}{(2n+2)!}$$

Beweis
Wir kennen bereits den Fehler für das Newtonsche und leiten dies völlig analog und einfach her. □

2.5 Spline-Interpolation

2.43 Motivation
Spline bedeutet eine Metal- oder Holzfeder, die zwischen Punkten eingespannt werden soll. Wir suchen also eine Glatte Funktion, die durch vorgegebene Punkte geht, d.h. wir wollen die zweite Ableitung minimal bekommen.

2.44 Konstruktion
Wir suchen eine Funktion so, daß $s(x_i) = y_i$ und $\int_a^b s''(x)^2 dx$ minimal wird, d.h.

$$\int_a^b s''(x)^2 \, dx \leq \int_a^b \left(s''(x) + \varepsilon h''(x)\right)^2 \, dx$$

Wenn $\varepsilon \in \mathbb{R} \quad \forall h : [a, b] \to \mathbb{R}$ ist zwei mal diffbar mit $h(x_i) = 0$. Damit:

$$\int_a^b s''(x)^2 \, dx + 2\varepsilon \int_a^b s''(x) h''(x) \, dx + \underbrace{\varepsilon^2 \int_a^b h''(x)^2 \, dx}_{>0}$$

Wir sehen, daß die obige Ungleichung erfüllt ist, wenn der mittlere Term Null ist, d.h.

$$\int_a^b s''(x) h''(x) \, dx = 0 \quad \forall h, h(x_i) = 0$$

durch partielle Integration erhalten wir:

$$\int_a^b s''(x) h''(x) \, dx = \underbrace{[s''(x) h'(x)]_a^b}_{=0} - \int_a^b s'''(x) h'(x) \, dx$$

Wir nehmen nun an, daß $s'''(x)$ stückweise konstant ist, d.h. $s''' = \alpha_i$ und wir erhalten:

$$\sum_{i=1}^{n} \alpha_i \int_{x_{i-1}}^{x_i} h'(x)\,dx = \sum_{i=1}^{n} \alpha_i \left(h(x_i) - h(x_{i-1})\right) = 0$$

2.45 Definition ((kubischer) Spline)
Ein (kubischer) *Spline* zu Stützstellen $a = x_0 < x_1 < \ldots < x_n = b$ ist eine Funktion $s : [a, b] \to \mathbb{R}$ mit

1. $s_i(x) := s|_{[x_{i-1},x_i]}$ ist ein Polynom vom Grad ≤ 3
2. $s : [a, b] \to \mathbb{R} \in \mathcal{C}^2([a, b])$

2.46 Satz
Sei $f : [a, b] \to \mathbb{R} \in \mathcal{C}^2([a, b])$ und sei s der interpolierende kubische Spline zu f, d.h. $s(x_i) = f(x_i)$ und zudem $s'(a) = f'(a)$ sowie $s'(b) = f'(b)$ (ein sog. *eingespannter Spline*), dann gilt:

$$\int_a^b s''(x)^2\,dx \leq \int_a^b f''(x)^2\,dx$$

Beweis
Der Beweis ergibt sich sofort aus der Betrachtung, daß $\varepsilon = 1$ ist. □

2.47 Konstruktion (des Polynoms)
Seien (x_i, y_i) für $i \leq n$ gegeben und $a = x_0 < x_1 < \ldots < x_n = b$ konstruieren einen interpolierenden Spline s, wobei wir dies in lokale Polynome vom Grad ≤ 3 aufteilen, die wir s_i nennen:

$$s_i(x_i) = y_i, \qquad s_i(x_{i-1}) = y_{i-1}$$

und

$$s_i'(x_i) = v_i, \qquad s_i'(x_{i-1}) = v_{i-1}$$

Wobei wir die v_i noch nicht kennen, aber berechnen können über:

$$s_i(x) = y_{i-1} + (x - x_{i-1})\delta y[x_{i-1}, x_i] + (x - x_{i-1})(x - x_i)\left(\alpha_i(x - x_{i-1}) + \beta_i(x - x_i)\right)$$

Damit ist $s_i'(x_{i-1}) = \delta y[x_{i-1}, x_i] + h_i^2\beta_i = v_{i-1}$ und $s_i'(x_i) = \delta y[x_{i-1}, x_i] + h_i^2\alpha_i = v_i$.
Hiermit erhalten wir für die Formel für den Spline, wenn wir die Ableitungen kennen:

$$\begin{aligned} s_i(x) = y_{i-1} &+ (x - x_{i-1})\delta y[x_{i-1}, x_i] \\ &+ \frac{(x - x_i)(x - x_{i-1})}{h_i^2}\Big((v_i - \delta y[x_{i-1}, x_i])(x - x_{i-1}) \\ &+ (v_{i-1} - \delta y[x_{i-1}, x_i])(x - x_i)\Big) \end{aligned}$$

Für bel. $v_0, \ldots, v_n$ erhalten wir also $s : [a, b] \to \mathbb{R}$ stückweise kubische Polynom, welches mindestens einmal diffbar ist und $s(x_i) = y_i$.
Außerdem soll gelten, daß $v_0 = s'(a) = f'(a)$ und $v_n = s'(b) = f'(b)$.

Wir müssen nun $v_1, \ldots, v_{n-1}$ so bestimmen, daß $s : [a, b] \to \mathbb{R}$ wenigstens zwei mal stetig diffbar ist, d.h. $s_i''(x_i) = s_{i+1}''(x_i)$.
Dies sind $n-1$ Bedingungen für $n-1$ Unbekannte, was uns sehr freudig stimmt, wenn diese noch unabhängig sind.

Wir erhalten aus der der Formel für den Spline folgende Bedingung:

$$s_i''(x_i) = \frac{2}{h_i}\left(2v_i + v_{i-1} - 3\delta y[x_{i-1}, x_i]\right)$$

und

$$s_{i+1}''(x_i) = -\frac{2}{h_{i+1}}\left(v_{i+1} + 2v_i - 3\delta y[x_i, x_{i+1}]\right)$$

Wir fordern also, daß beide Ableitungen gleich groß sind und erhalten folgendes LGS:

$$\frac{v_{i-1}}{h_i} + 2\left(\frac{1}{h_i} + \frac{1}{h_{i+1}}\right)v_i + \frac{v_{i+1}}{h_{i+1}} = 3\left(\frac{\delta y[x_{i-1}, x_i]}{h_i} + \frac{\delta y[x_i, x_{i+1}]}{h_{i+1}}\right)$$

Dies können wir als Matrix auffassen und sehen sofort, daß sie tridiagonal, symmetrisch und diagonaldominant ist:

$$\begin{pmatrix} 2\left(\frac{1}{h_1} + \frac{1}{h_2}\right) & \frac{1}{h_2} & & \\ \frac{1}{h_2} & 2\left(\frac{1}{h_2} + \frac{1}{h_3}\right) & \ddots & \\ & \ddots & \ddots & \frac{1}{h_{n-1}} \\ & & \frac{1}{h_{n-1}} & 2\left(\frac{1}{h_{n-1}} + \frac{1}{h_n}\right) \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_{n-1} \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_{n-1} \end{pmatrix}$$

wobei b_i die folgenden Werte annimmt:

$$\begin{aligned} b_1 &= 3\left(\frac{\delta y[x_0, x_1]}{h_1} + \frac{\delta y[x_1, x_2]}{h_2}\right) - \frac{v_0}{h_1} \\ b_i &= 3\left(\frac{\delta y[x_{i-1}, x_i]}{h_i} + \frac{\delta y[x_i, x_{i+1}]}{h_{i+1}}\right) \\ b_{n-1} &= 3\left(\frac{\delta y[x_{n-2}, x_{n-1}]}{h_{n-1}} + \frac{\delta y[x_{n-1}, x_n]}{h_n}\right) - \frac{v_n}{h_n} \end{aligned}$$

2.48 Satz (Beschränkung der Lösungsvektoren)
Sei $Av = b$ das oben beschriebene LGS, dann gilt

$$\max_i |v_i| \leq \frac{h}{2} \max_i |b_i|, \qquad h = \max_i h_i$$

2.49 Satz (Eindeutigkeit und Existenz der interpolierenden Splines)
Der eingespannte interpolierende Spline existiert auf eine eindeutige Art und Weise durch Lösen des oben angeführten LGS.

2.50 Bemerkung
Für eine äquivistante Unterteilung sieht A folgendermaßen aus:

$$\frac{1}{h}\begin{pmatrix} 4 & 1 & & \\ 1 & 4 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & 4 \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_{n-1} \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_{n-1} \end{pmatrix}$$

Dabei gilt:

$$\begin{aligned} b_1 &= \frac{3}{h^2}(y_2 - y_0) - hv_0 \\ b_k &= \frac{3}{h^2}(y_{k+1} - y_{k-1}) \qquad\qquad k = 2, \ldots, n-2 \\ b_{n-1} &= \frac{3}{h^2}(y_n - y_{n-2}) - hv_n \end{aligned}$$

Durch das Lösen von $Av = b$ mit dem Gauß-Algorithmus ergibt sich dann die folgende Matrix:

$$\begin{pmatrix} d_1 & 1 & & & \\ 1 & d_2 & 1 & & \\ & 1 & d_3 & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & d_{n-1} \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_{n-1} \end{pmatrix} = \begin{pmatrix} c_1 \\ \vdots \\ c_{n-1} \end{pmatrix}$$

wobei

$$\begin{aligned} d_1 &= 4 \\ d_{k+1} &= 4 - \frac{1}{d_k} && k \leq n-2 \\ c_1 &= b_1 \\ c_{k+1} &= b_{k+1} - \frac{c_k}{d_k} && k \leq n-2 \end{aligned}$$

Damit lassen sich ebenfalls rekursiv die v_k berechnen:

$$\begin{aligned} v_{n-1} &= \frac{c_{n-1}}{d_{n-1}} \\ v_k &= \frac{c_k - v_{k+1}}{d_k} \end{aligned}$$

2.6 Fehler bei der Spline-Interpolation

2.51 Motivation
Sei s der eingespannte interpolierende Spline und $f : [a, b] \to \mathbb{R}$.
Wir wollen nun den folgenden Fragen nachgehen:

1. Wie groß ist der Fehler, also gibt es eine Abschätzung für $|f(x) - s(x)|$?
2. Wenn wir leicht gestörte Werte haben, also statt (x_i, y_i) haben wir $(x_i, \tilde{y}_i)$. Wie groß ist die Abweichung des Splines, also $|s(x) - \tilde{s}(x)|$?

2.52 Bemerkung (Vorgehensweise für die Fehlerabschätzung)
Wir betrachten zur Abschätzung des Fehlers $|f(x) - s(x)|$ folgende 3 Schritte:

1. $|f'(x_i) - v_i| \leq ?$
2. Sei p_i das Hermite-IPP zu f in x_{i-1} und x_i und s_i der Spline mit Grad $s_i \leq 3$. Dadurch erhalten wir einen Fehler für $|p_i(x) - s_i(x)|$?
3. Für die Hermite-Interpolation kennen wir bereits den Fehler für $|f(x) - p_i(x)|$.

2.53 Satz (Fehler bei der Spline-Interpolation)
Sei $f : [a, b] \to \mathbb{R}$ 4-mal stetig diffbar und s das interpolierende kubische Spline zu f in $x_0, \ldots, x_n \in [a, b]$ mit
$s'(a) = f'(a)$ und $s'(b) = f'(b)$, $h = \max_i h_i$ und $h_i = x_i - x_{i-1}$. Dann gilt:

$$|f(x) - s(x)| \leq \frac{5}{384} h^4 \max_{\xi \in [a,b]} |f^{(4)}(\xi)|$$

Beweis
Um dies zu beweisen, betrachten wir die folgenden Hilfssätze. □

2.54 Hilfssatz (Abschätzung der Steigungen der Splines)
Unter der Voraussetzungen des Satzes gilt:

$$|f'(x_i) - v_i| \leq \frac{h^3}{24} \max_{\xi \in [a,b]} |f^{(4)}(\xi)|$$

Beweis
Wir führen den Beweis der Einfachkeit halber nur für den äquidistanten Fall.
v_i sind definiert durch

$$\frac{1}{h}\left(v_{i-1} + 4v_i + v_{i+1}\right) - \frac{3}{h^2}\left(f(x_{i+1}) - f(x_{i-1})\right) = 0$$

Wir untersuchen zunächst den Defekt, wenn man hier v_i durch $f'(x_i)$ ersetzt:

$$\frac{1}{h}\left(f'(x_{i-1}) + 4f'(x_i) + f'(x_{i+1})\right) - \frac{3}{h^2}\left(f(x_{i+1}) - f(x_{i-1})\right) =: \delta_i$$

Mit der Taylorentwicklung ergibt sich:

$$f(x_{i+1}) = f(x_i) + hf'(x_i) + \frac{h^2}{2!}f''(x_i) + \frac{h^3}{3!}f'''(x_i) + h^4 \int_0^1 \frac{(1-t)^3}{3!} f^{(4)}(x_i + h)\, dt$$

und analog

$$f'(x_{i+1}) = f'(x_i) + hf''(x_i) + \frac{h^2}{2!}f'''(x_i) + h^3 \int_0^1 \frac{(1-t)^2}{2!} f^{(4)}(x_i + th)\, dt$$

Durch Einsetzen in die obige Bedingung ergibt sich:

$$\delta_i = h^2 \int_0^1 \underbrace{\left(\frac{(1-t)^2}{2!} - 3\frac{(1-t)^2}{3!}\right)}_{\geq 0} f^{(4)}(x_i + th)\, dt$$
$$+ h^2 \int_0^1 \underbrace{\left(\frac{(1-t)^2}{2!} - 3\frac{(1-t)^2}{3!}\right)}_{\geq 0} f^{(4)}(x_i - th)\, dt$$

Durch den Mittelwertsatz ergibt sich:

$$\delta_i = \frac{h^2}{24} f^{(4)}(\xi) + \frac{h^2}{24} f^{(4)}(\psi)$$

Damit ergibt sich:

$$|\delta_i| \leq \frac{h^2}{12} \max_{x\in[a,b]} |f^{(4)}(x)|$$

Die Differenzen $f'(x_i) - v_i =: e_i$ erfüllen ein LGS, was sich mittels folgender Gleichung lösen läßt:

$$|e_i| \leq \frac{h}{2} \max_j |\delta_j| \leq \frac{h^3}{24} \max_{x\in[a,b]} |f^{(4)}(x)|$$

□

2.55 Bemerkung
Sei s_i das auf $[x_{i-1}, x_i]$ eingeschränkte Polynom vom Grad ≤ 3 mit $s(x_{i-1}) = f(x_{i-1}), s(x_i) = f(x_i)$ und $s'(x_{i-1}) = v_{i-1}$ und $s'(x_i) = v_i.$, dann ist $s_i(x)$:

$$\begin{aligned} s_i(x) = f(x_{i-1}) &+ (x - x_{i-1})\delta f[x_{i-1}, x_i] \\ &+ \frac{(x - x_i)(x - x_{i-1})}{h_i^2}\Big((v_i - \delta f[x_{i-1}, x_i])(x - x_{i-1}) \\ &+ (v_{i-1} - \delta f[x_i, x_{i+1}])(x - x_i)\Big) \end{aligned}$$

Betrachte nun zusätzlich die Hermite IPP p_i vom Grad $p \leq 3$, wobei gilt

$$p_i(x_{i-1}) = f(x_{i-1}), \qquad p_i(x_i) = f(x_i), \qquad p_i'(x_{i-1}) = f'(x_{i-1}), \qquad p_i'(x_i) = f'(x_i)$$

und hat die gleiche Form wie $s_i(x)$, nur daß v_{i-1} durch $f'(x_{i-1})$ und v_i durch $f'(x_i)$ ersetzt wird.

2.56 Hilfssatz (Abschätzung von Spline-Interpolation und Hermite Interpolation)
Unter den Voraussetzungen vom Satz und mit der Beziehung zwischen Hermite IPP und Spline-Interpolation für $x \in [x_{i-1}, x_i]$ gilt:

$$|s_i(x) - p_i(x)| \leq \frac{h^4}{96} \max_{\xi\in[a,b]} |f^{(4)}(\xi)|$$

Beweis
Es gilt:

$$s_i(x) - p_i(x) = \underbrace{\frac{(x - x_i)(x - x_{i-1})}{h_i^2}}_{\leq \frac{1}{4}} \left((v_i - f'(x_i))(x - x_{i-1}) + (v_{i-1} - f'(x_{i-1}))(x - x_i)\right)$$

Damit gilt für den Betrag:

$$|s_i(x) - p_i(x)| \leq \frac{1}{4}\frac{h^3}{24} \max |f^{(4)}| \cdot h \leq \frac{h^4}{96} \cdot \max |f^{(4)}|$$

□

2.57 Hilfssatz
Für die gleichen Voraussetzungen wie oben und $x \in [x_{i-1}, x_i]$ gilt:

$$|f(x) - p_i(x)| \leq \frac{h^4}{384} \max_{\xi \in [a,b]} |f^{(4)}(\xi)|$$

Beweis (des Hauptsatzes)
Wir können nun einfach den Hauptsatz beweisen. Nach der Dreiecksungleichung gilt:

$$|f(x) - s_i(x)| \leq |f(x) - p_i(x)| + |p_i(x) - s_i(x)|$$

und damit folgt mit Hilfe der Hilfssätze die Behauptung. □

2.58 Bemerkung (Vorgehensweise zur Abschätzung des Fehlers)
Sei s der Spline zu (x_i, y_i) und $\tilde{s}$ der Spline zu $(x_i, \tilde{y}_i)$. Wir wollen die Differenz von diesen Splines wissen und sehen:

$$s(x) - \tilde{s}(x) = \sum_{i=0}^{n} (y_i - \tilde{y}_i) \cdot \ell_i(x)$$

wobei $\ell_i(x)$ die sog. *Lagrange-Splines* sind mit

$$\ell_i(x_j) = \begin{cases} 1, & i = j \\ 0, & \text{sonst} \end{cases}, \qquad \ell_i'(a) = \ell_i'(b) = 0$$

Im Gegensatz zur Polynominterpolation oszillieren die ℓ_i nicht über alle Grenzen hinweg, sondern sie fallen weg vom Hochpunkt $\ell_i(x_i)$ ab. Dies können wir zeigen:

2.59 Proposition (Abfallen der Lagrange-Splines)
Sei $v_j = \ell_i'(x_j)$ mit i fest für äquidistante Stützstellen erhalten wir ein LGS:
Die Matrix hat auf der Hauptdiagonalen überall 4 und ober- und unterhalb Einsen. Der Lösungsvektor hat an der $i-1$. Stelle $\frac{3}{h}$ stehen und an der $i+1$. Stelle $-\frac{3}{h}$ stehen.

Mit der Gauß-Elimination erhalten wir dann: Für $k < i$ ergibt sich: $v_k = -\frac{1}{d_k} v_{k+1}$ und für $k > i$ ergibt sich: $v_k = -\frac{1}{d_k} v_{k-1}$.
Damit nimmt $|v_k|$ rasch ab, also bleiben Störungen nur lokal und nicht global wirksam.

2.60 Proposition (Genauere Abschätzung für die Lagrange-Splines)
Wir betrachten nun

$$|s(x) - \tilde{s}(x)| \leq \Lambda_n \max_{i \leq n} |y_i - \tilde{y}_i|$$

wobei wir Λ_n analog zu früher als *Lebesgue-Konstante* bezeichnen:

$$\Lambda_n = \max_{x \in [a,b]} \sum_{i=0}^{n} |\ell_i(x)|$$

2.61 Satz (Lebesgue-Konstante bei äquidistanter Unterteilung)
Mit einer geeigneten Rechnung kann man zeigen, daß die Lebesgue-Konstante $\leq 2\ \forall n \in \mathbb{N}$

2.62 Zusammenfassung (Vergleich der Verfahren an einem Beispiel)
Sei $f(x) = \sqrt{|x|}$ für $x \in [-1, 1]$.
Wir betrachten nun die Interpolation für verschiedene Interpolationsverfahren

1. Die Polynom-Interpolation mit äquidistanter Unterteilung für n Punkte konvergiert für $n \to \infty$ nur in $-1, 0$ und 1.
2. Die Polynom-Interpolation mit den Tschebyscheff-Knoten konvergiert gleichmäßig auf $[-1, 1]$. Wobei der Fehler etwa mit $\frac{C(x)}{\sqrt{n}}$ geht.
3. Bei der Spline-Interpolation mit äquidistanter Unterteilung konvergiert gleichmäßig auf $[-1, 1]$ und der maximale Fehler geht mit $\frac{C}{\sqrt{n}}$, außerhalb von $(-\varepsilon, \varepsilon)$ geht der Fehler mit $\frac{C_\varepsilon}{n^4}$.
4. Bei der Spline-Interpolation mit bei 0 verfeinertem Gitter, d.h. $x_i = \left(\frac{i}{n}\right)^8$ geht der maximale Fehler bei $[0, 1]$ mit $\frac{C}{n^8}$.

Für glatte Funktionen ist für die Forderung einer hohen Genauigkeit die Tschebyscheff-Interpolation die beste Wahl, sonst ist die Spline-Interpolation am besten.

2.7 Numerische Differentiation

2.63 Motivation
Die Problemstellung dieses Abschnittes: Wir haben eine Funktion $f : (a, b) \to \mathbb{R}$, wobei $f \in \mathcal{C}^1([a, b])$, gegeben und wir möchten näherungsweise $f'(x)$ für einzelne Punkte oder gar die Funktion f' berechnen.

2.64 Konstruktion (Erste Näherung mit dem Differenzenquotienten)
Da der Differentialquotienten mit

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

definiert ist, ergibt sich für den Fehler über die Taylorentwicklung, falls $f \in \mathcal{C}^2([a, b])$:

$$\exists \xi \in [x_0, x_0 + h] : \ \frac{f(x+h) - f(x)}{h} = f'(x) + \frac{h}{2} f''(\xi) \iff \text{Fehler} = \frac{h}{2} f''(\xi)$$

Wobei es eine Schwierigkeit ist, daß $h \to 0$ auf Computern schwer zu realisieren ist. Daher müssen wir mit einem Rundungsfehler rechnen.

2.65 Bemerkung (Rundungsfehler)
Wir berechnen also statt dem genäherten Differentialquotienten nun $\frac{f(x+h)-f(x)+\varepsilon}{h}$, wobei ε den Rundungsfehler bezeichnet.
Wir würden aber für $h \to 0$ ein $\frac{\varepsilon}{h} \to \infty$ beobachten (am Computer sicherlich eher gegen einen konstanten Wert)

Wir nehmen im Folgenden an, daß `eps` $\approx \varepsilon$ ist.
Dabei ist `eps` die kleinste Maschinenezahl mit $[1 + \texttt{eps}]_{\text{gerundet}} \neq 1$.
Es ergibt sich für unser Problem, daß der kleinste Fehler dann bei $h = \sqrt{\texttt{eps}}$, bzw. für große x auch $h \approx \sqrt{\texttt{eps}(1 + |x|)}$.
Bei C++ liegt bspw. die Maschinengenauigkeit für `double` bei etwa 10^{-16}.

2.66 Definition (Zentraler Differenzenquotient)
Wir betrachten nun für $f \in \mathcal{C}^3([a, b])$ einen *zentralen Differenzenquotienten* und erhalten mit der Taylorentwicklung:

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + \frac{h^2}{6} \cdot f'''(\xi) \quad \xi \in [x-h, x+h]$$

Wir erreichen hierbei die selbe Genauigkeit mit einem größeren h, d.h.

$$\frac{\varepsilon}{h} \approx h^2 \;\Rightarrow\; h \approx \sqrt[3]{\texttt{eps}}$$

2.67 Bemerkung
Für die partiellen Ableitungen $\frac{\partial f}{\partial x_i}$ von $f(x_1, \ldots, x_n)$ benötigen wir bei normalen Differenzennquotienten $n + 1$ Auswertungen, denn

$$\frac{\partial f}{\partial x_i}(x_1, \ldots, x_n) \approx \frac{f(x_1, \ldots, x_i + h, \ldots, x_n) - f(x_1, \ldots, x_n)}{h}$$

Bei zentralen Differenzenquotienten benötigen wir daher allerdings $2n$ Funktionsauswertungen.
Daher wird bei partiellen Ableitungen öfter der normale Differenzenquotient benutzt.

2.68 Motivation (Approximation einer Funktion auf einem Intervall)
Wir möchten im Folgenden $f : (a, b) \to \mathbb{R}$ approximieren. Dazu interpolieren wir f durch Polynome oder einen Spline p und nehmen p' als Approximation von f'.

2.69 Proposition (Ableitung des Interpolationspolynom)
Wir wissen, daß $p(x) = x^n \cdot \delta^n y[x_0, \ldots, x_n] + \ldots$. Leiten wir dies n-mal ab, so ergibt sich:
$p^{(n)}(x) = n!\delta^n y[x_0, \ldots, x_n] = \text{const}$.
Die Auswertung des Polynoms an einer Stelle x ergibt sich mit dem Horner-Schema. Dies entspricht dem Hinzufügen eines weiteren Knotens, x, im dividierten Differenzen-Schema, d.h.

$$b_0 := p(x), \qquad b_k := \delta^k p[x, x_0, \ldots, x_{k-1}]$$

insbesondere gilt:

$$b_n = \delta^n p[x, x_0, \ldots, x_{n-1}] = \frac{p^{(n)}(\xi)}{n!} = \delta^n y[x_0, \ldots, x_n]$$

Damit können wir rekursiv rückwärts alle divierten Differenzen berechnen:

$$b_{k+1} = \frac{b_k - \delta^k y[x_0, \ldots, x_k]}{x - x_k} \iff b_k = \delta^k y[x_0, \ldots, x_k] + (x - x_k) \cdot b_{k+1}$$

Mit der gleichen Idee erhalten wir die Ableitungen, d.h. wir fügen $x - \varepsilon$ als weiteren Knoten hinzu und betrachten anschließend $\varepsilon \to 0$.

Wir erhalten als div. Differenzen für diesen Knoten c_i, wobei $c_n = b_n$ völlig analog zu oben ist und wir erhalten:

$$c_k = \delta^k p[x, x, x_0, \ldots, x_{k-2}]$$

2.70 Algorithmus (zur Berechnung von der Ableitung des Interpolationspolynoms)
Wir gehen dabei von rechts nach links:

1. $c_n = b_n$
2. $c_k = b_k + (x - x_{k-1})c_{k+1}$
3. $p'(x) = c_1 \cdot 1!$

Damit erhält man auch die höheren Ableitungen, z.B.

1. $d_n = c_n$
2. $d_k = c_k + (x - x_{k-2})d_{k+1}$
3. $d_2 = p''(x) \cdot 2!$.

2.71 Satz (Fehleruntersuchung)
Sei $f : [a, b] \to \mathbb{R}$ und $f \in \mathcal{C}^{n+2}([a, b])$ und sei p das IPP zu f in $x_0, \ldots, x_n \in [a, b]$ mit $\deg p \leq n$. Dann gilt $\forall x \in [a, b]$, daß es $\xi, \xi' \in (a, b)$ gibt, mit:

$$f'(x) - p'(x) = \sum_{i=0}^{n} \left(\prod_{j \neq i} (x - x_j) \frac{f^{(n+1)}(\xi)}{(n+1)!} \right) + \prod_{j=0}^{n} (x - x_j) \cdot \frac{f^{(n+2)}(\xi')}{(n+2)!}$$

insbesondere gilt dabei:

$$f'(x_i) - p'(x_i) = \prod_{i \neq j} (x_i - x_j) \frac{f^{(n+1)}(\xi)}{(n+1)!}$$

Beweis
Der Beweis verläuft fast analog zum Beweis für die allg. Fehlerfortpflanzung. Dabei zeigen wir das Ergebnis für $x = \bar{x}$.
Sei $\bar{p}(x)$ das Hermite IPP mit:

$$\forall i \quad \bar{p}(x_i) = f(x_i), \qquad \bar{p}(\bar{x}) = f(\bar{x}), \qquad \bar{p}'(\bar{x}) = f'(\bar{x})$$

Das Hermite IPP sieht nun folgendermaßen aus:

$$\begin{aligned}\bar{p}(x) = p(x) &+ (x - x_0) \cdot \ldots \cdot (x - x_n) \underbrace{\delta^{n+1} f[x_0, \ldots, x_n, \bar{x}]}_{= \frac{f^{(n+1)}(\xi)}{(n+1)!}} \\ &+ (x - x_0) \cdot \ldots \cdot (x - x_n)(x - \bar{x}) \underbrace{\delta^{n+2} f[x_0, \ldots, x_n, \bar{x}, \bar{x}]}_{= \frac{f^{(n+2)}(\xi)}{(n+2)!}}\end{aligned}$$

Nun setzen wir in die diese Formel $\bar{p}'(\bar{x}) = f'(\bar{x})$ ein und sind fertig. □

2.72 Algorithmus (Clenshaw-Algorithmus zur Berechnung von Ableitungen)
Analog können wir ein Polynom mit Tschebyscheff-Knoten mit dem Clenshaw-Algorithmus berechnen, wodurch sich als Ableitung die auf dem Übungsblatt zu beweisende Formel ergibt.

2.73 Proposition (Abschätzung des Fehlers bei der Spline-Interpolation)
Es gilt:

$$|v_i - f'(x_i)| \leq \frac{h^3}{24} \max |f^{(4)}|$$

Wir berechnen nun $s'(x)$ zwischen den Knoten x_{i-1}, x_i, indem wir dies ebenfalls übers Hermite IPP machen. Damit ergibt sich auf ähnliche Weise:

$$\begin{aligned}|f'(x) - s'(x)| &\leq \frac{h^3}{24} \max |f^{(4)}| \\ |f''(x) - s''(x)| &\leq \frac{3}{8} h^2 \max |f^{(4)}| \\ |f'''(x) - s'''(x)| &\leq \frac{1}{2} h \max |f^{(4)}|\end{aligned}$$

2.8 Extrapolationsverfahren

2.74 Motivation
Man möchte die Ableitung $f'(t)$ und $f''(t)$ mit hoher Genauigkeit berechnen, z.B. $f''(t)$.
Dies geschieht über den im letzten Kapitel gelernten Algorithmus:

$$\begin{aligned}\frac{f(t-h) - 2f(t) + f(t+h)}{h^2} = f''(t) &+ \frac{2f^{(4)}(t)}{4!} h^2 + \frac{2f^{(6)}(t)}{6!} h^4 \\ &+ \ldots + \frac{2f^{(2N)}(t)}{(2N)!} h^{2N-2} + \mathcal{O}(h^{(2N)})\end{aligned}$$

Wegen Rundungsfehlern kann man h nicht bel. klein wählen.
Wir sehen aber, daß sich der Ausdruck rechts wie ein Polynom über h^2 verhält (bis auf den Restterm der Ordnung $2N+2$).

2.75 Idee (von Richardson, 1920)
Wir berechnen nun die linke Seite der obigen Gleichung für $h_1 > h_2 > h_3$ und wollen dann aufgrund der Werte der linken Seite auf ein Polynom extrapolieren und erhalten folgende Näherungen:

$$y_1 = \frac{f(t+h_1) - 2f(t) + f(t-h_1)}{h_1^2}$$
$$y_2 = \frac{f(t+h_2) - 2f(t) + f(t-h_2)}{h_2^2}$$
$$y_3 = \frac{f(t+h_3) - 2f(t) + f(t-h_3)}{h_3^2}$$

Wir suchen nun das Interpolationspolynom p durch (h_i^2, y_i) und nehmen $p(0)$ als (bessere) Näherung an $f''(t)$.
Dies ist die *Extrapolation* auf $h^2 = 0$.

2.76 Hilfssatz
Seien $(x_0, y_0), \ldots, (x_n, y_n)$ gegeben und x_i alle verschieden.
Sei p_0 das IPP, welches durch $(x_0, y_0), \ldots, (x_{n-1}, y_{n-1})$ geht und p_1 sei das IPP durch die Punkte $(x_1, y_1), \ldots, (x_n, y_n)$. Wir kennen bereits ein Polynom, das durch alle $n+1$ Punkte geht, und zwar:

$$p(x) = \frac{(x - x_0)p_1(x) + (x_n - x)p_0(x)}{x_n - x_0}$$

2.77 Algorithmus (H-Quadrat-Extrapolation)
Wir setzen nun

$$T_{j1} = y_j = \frac{f(t+h_j) - 2f(t) + f(t-h_j)}{h_j^2}$$

Und $T_{j,k+1}$ sei der Wert in $x = 0$ des IPP durch $(h_j^2, y_j), (h_{j-1}^2, y_{j-1}), \ldots, (h_{j-k}^2, y_{j-k})$ für $j \geq k+1$.
Nach dem Hilfssatz berechnet man $T_{j,k+1}$ aus $T_{j,k}$ und $T_{j-1,k}$ mit, dabei gilt $x = 0$ und $x_0 = h_j^2$ und $x_n = h_{j-k}^2$, $n = k$ und $j \geq k+1$:

$$\begin{aligned} T_{j,k+1} &= \frac{-h_j^2 T_{j,k} + h_{j-k}^2 T_{j-1,k}}{h_{j-k}^2 - h_j^2} \\ &= T_{j,k} + \frac{T_{j,k} - T_{j-1,k}}{\frac{h_{j-k}^2}{h_j^2} - 1} \end{aligned}$$

Dabei ist es üblich, daß man $h_j := \frac{H}{n_j}$ wählt und $n_j = j$. Mit dieser Wahl ergibt sich:

$$T_{j,k+1} = T_{j,k} + \frac{T_{j,k} - T_{j-1,k}}{\left(\frac{n_j}{n_{j-k}}\right)^2 - 1} \quad k+1 \leq j$$

2.78 Bemerkung (Rechenaufwand und Fehler)
Der Rechenaufwand wird im Wesentlichen durch die Anzahl der Funkiontsauswertungen bestimmt, der sich durch die Anzahl der verwendeten Zeilen wiederspiegelt.
Wir fragen uns, wieviele Zeilen zur Berechnung sinnvoll sind.

Wir schätzen den Fehler ab durch:

$$e_j = T_{jj} - T_{j,j-1}$$

Wir gehen nun soweit, bis $|e_j| \leq \texttt{TOL}$, wobei `TOL` die Toleranz ist;
oder bis $|e_j| > |e_{j-1}|$ ist (dies kann aus Rundungsfehlern z.B. auftreten).

2.79 Proposition (Theoretische Betrachtung des Fehlers)
Sei $y(h) = y(0) + c_1 h^2 + c_2 h^4 + \ldots + c_N h^{2N} + \mathcal{O}(h^{2N+2})$
Wir gehen davon aus, daß wir bereits $y_i = y(h_i)$ berechnet haben, als z.B.

$$y(h) = \frac{1}{h^2}\left(f(t+h) - 2f(t) + f(t+h)\right) \quad \text{oder} \quad y(h) = \frac{1}{2h}\left(f(t+h) - f(t-h)\right)$$

Wir interessieren uns für den Fehler $y(0) - T_{j,k}$.
Setze nun $\tilde{y}(h) = y(0) + c_1 h^2 + \ldots + c_N h^{2N}$ als abbrechende Entwicklung und $\tilde{y}_i = \tilde{y}(h_i)$ für die Fehlerentwicklung, die nicht berechnet werden.
Wir betrachten nun die IPP:

$$p(x) - \tilde{p}(x) = \sum_i (y_i - \tilde{y}_i)\ell_i(x)$$

wobei

$$\ell_i(x) = \frac{\prod_{j \neq i}(x - h_j^2)}{\prod_{j \neq i}(h_i^2 - h_j^2)}$$

Damit ergibt sich für $x = 0$:

$$p(0) - \tilde{p}(0) = \sum (y_i - \tilde{y}_i)\ell_i(0)$$

und es gilt, weil wir eine Verteilung $h_i = \frac{H}{n_i}$ annehmen:

$$\ell_i(0) = \prod_{j \neq i} \frac{1}{1 - \frac{h_i^2}{h_j^2}} = \prod_{i \neq j} \frac{1}{1 - \frac{n_j^2}{n_i^2}} = \mathcal{O}(1)$$

Somit ist $T_{j,k} - \tilde{T}_{j,k} = \mathcal{O}(H^{2N+2})$, womit wir diesen Term in Zukunft ignorieren können.
Wir können nun den Fehler der Interpolation auf die abbrechende Entwicklung anwenden und erhalten:

$$\begin{aligned} y(0) - \tilde{T}_{j,k} &= (0 - h_j^2) \cdot \ldots \cdot (0 - h_{j-k+1}^2) \frac{1}{k!} \frac{d^k \tilde{y}}{(dh^2)^k}(\xi) \\ &= \frac{(-1)^k}{n_j^2 \cdot \ldots \cdot n_{j-k+1}^2} H^{2k} \cdot c_k + \mathcal{O}(H^{2k+2}) \end{aligned}$$

2.80 Satz (Fehler bei der Extrapolation)
Sei $y(h) = y(0) + c_1 h^2 + c_2 h^4 + \ldots + c_N h^{2N} + \mathcal{O}(h^{2N+2})$ eine asymptotische h^2-Entwicklung. Wenn die $T_{j,k}$ aus den $y_i = y(h_i)$ wie oben berechnet sind und $h_i = \frac{H}{n_i}$, dann gilt für $k \leq N$:

$$y(0) - T_{j,k} = \frac{(-1)^k}{n_j^2 \cdot \ldots \cdot n_{j-k+1}^2} c_k H^{2k} + \mathcal{O}(H^{2n+2})$$

2.81 Bemerkung
Für einen asymmetrischen Differenzenquotienten ergibt sich eine h-Entwicklung:

$$\frac{f(t+h) - f(t)}{h} = f'(t) + h \frac{f''(t)}{2!} + h^2 \frac{f'''(t)}{3!} + \ldots$$

Dafür kann man analog die gleiche Rechnung durchführen, was jedoch zu einem größeren Fehler führt.

2.82 Bemerkung (Ausblick)
Die Extrapolation, egal ob h- oder h^2-Extrapolation, kann man vielseitig anwenden, bspw. bei der numerischen Integration:

$$\int_{x_0}^{x_0+H} f(x)\, dx$$

Mit der Trapezregel und $h = \frac{H}{n}$ ergibt sich, falls $f \in \mathcal{C}^{2N+2}$ ist:

$$\begin{aligned} y(h) &= \frac{h}{2} f(x_0) + h f(x_0 + h) + \ldots + h f(x_0 + (n-1)h) + \frac{h}{2} f(x_0 + nh) \\ &= \int_{x_0}^{x_0+H} f(x)\, dx + c_1 h^2 + c_2 h^4 + \ldots + c_N h^{2N} + \mathcal{O}(h^{2N+2}) \end{aligned}$$

Dies ist die Euler-MacLaurinsche Summenformel, die 1745 entstanden ist und wir ohne Beweis annehmen können.
Hiermit können wir analog das oben berechnete Verfahren anwenden.
Weitere Anwendungen können wir bei den Differentialgleichungen sehen.

3 Lineare Gleichungssysteme und lineare Ausgleichsrechnungen

3.1 Gauß-Elimination

3.1 Motivation
Wir möchten im Folgenden $Ax = b$ lösen, wobei A eine reelle, invertierbare $n \times n$-Matrix mit $A \in \mathbb{R}^{n\times n}$ ist und $b \in \mathbb{R}^n$ ein Spaltenvektor ist, d.h.

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ \qquad\qquad \vdots & \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{cases}$$

Wir nehmen an, daß $a_{11} \neq 0$ (ansonsten Zeilentausch, falls A invertierbar).
Für $i = 2, \ldots, n$ ersetzen wir die i-te Zeile durch:

$$(\text{Zeile i}) - \ell_i(\text{Zeile 1})$$

Damit erhalten wir das äquivalente LGS:

$$\begin{array}{llll} a_{11}^{(1)}x_1 & +a_{12}^{(1)}x_2 & +\ldots & +a_{1n}^{(1)}x_n = b_1^{(1)} \\ a_{21}^{(1)}x_1 & +a_{22}^{(1)}x_2 & +\ldots & +a_{2n}^{(1)}x_n = b_2^{(1)} \\ \vdots & \vdots & \vdots & \vdots \quad\quad \vdots \\ a_{n1}^{(1)}x_1 & +a_{n2}^{(1)}x_2 & +\ldots & +a_{nn}^{(1)}x_n = b_n^{(1)} \end{array}$$

mit $a_{1j}^{(1)} = a_{1j}$, $b_1^{(1)} = b_1$ und

$$a_{ij}^{(1)} = a_{ij} - l_{1i}a_{1j}, \qquad b_i^{(1)} = b_i - \ell_{i1} \cdot b_1$$

Man wiederholt nun den Vorgang mit der Untermatrix der Dimension $(n-1) \times (n-1)$ von $A^{(1)}$, die wiederrum invertierbar ist, weil das LGS eindeutig lösbar ist und vertausche die Zeilen so, daß $a_{22}^{(1)} \neq 0$. Für $i = 3, \ldots, n$ setzen wir dann:

$$\ell_{i2} = \frac{a_{i2}^{(1)}}{a_{22}^{(1)}}$$

und die i. Zeile wie oben.
Es ergibt sich folgendes Schema:

$$(A, b) \to (A^{(1)}, b^{(1)}) \to (A^{(2)}, b^{(2)}) \to \ldots \to (A^{(n-1)}, b^{(n-1)}) = (R, c)$$

Dabei ist R eine obere Dreiecksmatrix mit $r_{ii} \neq 0$ und $r_{11} = a_{11}, r_{22}, \ldots, r_{nn} = a_{nn}^{(n-1)}$. Das LGS ist nun $Rx = c$.
Für die Lösung von diesem LGS erhält man dann:

$$x_n = \frac{c_n}{r_{11}}, \qquad x_i = \left(c_i - \sum_{j=i+1}^{n} r_{ij} x_j \right) \cdot \frac{1}{r_{ii}}$$

3.2 Satz (LR-Zerlegung)
Sei $A \in \mathbb{R}^{n \times n}$ invertierbar. Die Gaußelemination liefert:

$$P \cdot A = L \cdot R$$

mit L eine untere Dreiecksmatrix und R eine obere Dreiecksmatrix und P ist Permutationmatrix.

Beweis
Wir nehmen an, daß die notwendigen Zeilenumtaschungen für die Gauß-Elimination bereits zu Beginn durchgeführt wurden, d.h. wir ersetzen A durch PA.
Wir setzen:

$$L_i = \begin{cases} 1, & i = j \\ -\ell_{ik} & j = k \wedge i > k \\ 0, & \text{sonst} \end{cases}$$

Damit ist $A^{(1)} = L_1 PA$ und $A^{(k)} = L_k A^{(k-1)}$. Damit erhalten wir:

$$R = A^{(n-1)} = L_{n-1} A^{(n-2)} = \ldots = L_{n-1} L_{n-2} \cdot \ldots \cdot L_1 PA$$

Zu zeigen bleibt noch, daß $L_1^{-1} \cdot \ldots \cdot L_{n-1}^{-1} = L$ ist.
Wir setzen hierzu

$$V_i = \begin{cases} \ell_{ik} & j = k \wedge i > k \\ 0, & \text{sonst} \end{cases}$$

Damit $L_i = I - V_i$ und wir sehen, daß $V_i V_k = 0$ für $i \leq k$.
Damit ergibt sich $(I + V_i)(I - V_i) = I + V_i - V_i - V_i V_i = I$. Damit ist $(I + V_i)$ ist die Inverse zu L_i.

$I + V_i$ ist eine untere Dreiecksmatrix und untere Dreiecksmatrizen sind unter Multiplikation abgeschlossen, bzw. (wenn man viel Rechnerei mag):

$$L_1^{-1} \cdot \ldots \cdot L_{n-1}^{-1} = (I + V_1) \cdot \ldots \cdot (I + V_{n-1}) = I + V_1 + \ldots + V_{n-1} + V_1 V2 + \ldots = L \quad \square$$

3.3 Algorithmus
Dabei ergibt sich der folgende Algorithmus:
Setze $A^{(0)} := A$. Für $k = 1, \ldots, n-1$:

1. Setze
$$\ell_{ik} = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}$$
2. Berechnung der Koeffizienten von $A^{(k)}$ für $j = 1, \ldots, n$:
$$\begin{aligned} a_{ij}^{(k)} &= a_{ij}^{(k-1)} && i = 1, \ldots, k \\ a_{ij}^{(k)} &= a_{ij}^{(k-1)} - \ell_{ik} \cdot a_{kj}^{(k-1)} && i = k+1, \ldots, n \end{aligned}$$

3.4 Bemerkung (zur Permutationsmatrix)
Eine Permutationmatrix ist eine Matrix, deren Spalten aus Einheitsvektoren besteht, z.B:

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

3.5 Bemerkung (Determinantenberechnung)
Die Beziehung $PA = LR$ ist besonders praktisch, um die Determinante einer Matrix zu berechnen. Denn $\det(PA) = \det(LR) \iff \det(P)\det(A) = \det(L)\det(R) \iff \pm\det A = \det(L)\det(R) = 1 \cdot \det(R)$ und damit

$$\det A = \pm \prod_{k \in n} r_{kk} = r_{11} \cdot \ldots \cdot r_{nn}$$

3.6 Algorithmus (zur Lösung eines LGS mit LR-Zerlegung)
Ist die LR-Zerlegung von A bekannt, ist ist das LGS $Ax = b$ wie folgt zu lösen.

$$PAx = L \underbrace{Rx}_{y} = Pb$$

1. Wir lösen zuerst $Ly = Pb$, dies ist einfach, da L eine untere Dreiecksmatrix ist von oben nach unten
von $y_1 \to y_n$.
2. Dann lösen wir $Rx = y$, was eine obere Dreiecksmatrix ist von unten nach oben, also von $x_n, x_{n-1} \to x_1$.

3.7 Proposition (Betrachtung des Rechenaufwands)

- Für $A \to A^{(1)}$ benötigt man $n-1$ Divisionen. Man benötigt die gleiche Anzahl von Multiplikationen, d.h. man benötigt etwa n^2 Operationen.

  Für die Berechnung von $A \to R, L$ benötigen wir insgesamt $n^2 + (n-1)^2 + (n-2)^2 + \ldots + 2^2 + 1 \approx \frac{n^3}{3}$ Operationen.

- Die Berechnung von y aus $Ly = b$ benötigen wir etwa $\frac{n^2}{2}$ Operationen und für die Berechnung von x aus $Rx = y$ benötigt ebenfalls mit $\frac{n^2}{2}$ Operationen.
  Damit liegt der Hauptaufwand in der LR-Zerlegung.

3.2 Pivotwahl und Implementierung der Gauß-Elimination

3.8 Motivation
Wir wollen wissen, wie sich Rundungsfehler auf die Berechnung auswirkt. Dazu betrachten wir zunächst ein Beispiel als Motivation:

3.9 Beispiel
Sei folgendes LGS gegeben:

$$\begin{array}{lll} 10^{-4}x_1 & +x_2 & = 1 \\ x_1 & +x_2 & = 2 \end{array}$$

Die exakte Lösung ist $x_1 = 1,000100010001\ldots$ und $x_2 = 0,99989998\ldots$.
Bei einer 3-stelligen Gleichpunktrechung erhält man:

$$\begin{array}{lll} 0,100 \cdot 10^{-3}x_1 & +0,100 \cdot 10^{1}x_2 & = 0,100 \cdot 10^{1} \\ 0,100 \cdot 10^{1}x_1 & +0,100 \cdot 10^{1}x_2 & = 0,200 \cdot 10^{1} \end{array}$$

Wir erhalten wegen $a_{11} = 10^{-4}$ die Pivots

1. $\ell_{21} = \frac{a_{21}}{a_{11}} = 10^4$.
   Mit 3-stelliger Genauigkeit ergibt sich: $x_2 = 1$, aber $x_1 = 0$, was völlig falsch ist.
2. Durch den Zeilenumtausch erhalten wir wegen $a_{21} = 1$ den Pivot $\ell_{21} = \frac{a_{11}}{a_{21}} = 10^{-4}$.
   Mit 3-stelliger Genauigkeit ergibt sich damit: $x_2 = 1$ und $x_1 = 2 - x_1 = 2 - 1 = 1$, was nötig ist.

3.10 Bemerkung (Abstrakte Erklärung des Beispiels)
Fals $|\ell_{21}|$ groß ist, dann gilt:

$$a_{22}^{(1)} = a_{21} - \ell_{21}a_{12} \approx -\ell_{21}a_{12}, \qquad b_2^{(1)} = b_2 - \ell_{21}b_1 \approx -\ell_{21}b_1$$

Damit erhalten wir:

$$x_2 = \frac{b_2^{(1)}}{a_{22}^{(12)}} \approx \frac{b_1}{a_{12}}$$

Aber bei Berechnung von x_1 ergibt sich eine Stellenauslöschung:

$$x_1 = \frac{b_1 - a_{12}x_2}{a_{11}} \approx 0$$

Der Ausweg davon ist der Zeilentausch: $\rightarrow |a_{21}| \leq |a_{11}|$ und damit $|\ell_{21}| \leq 1$

3.11 Bemerkung (Spaltenpivotsuche)
Wir nehmen das Pivot im $(k+1)$-ten Schritt:

$$a^{(k)}_{j,k+1} \quad \text{mit} \quad \left|a^{(k)}_{j,k+1}\right| = \max_{i:k+1\leq i\leq n} \left|a^{(k)}_{i,k+1}\right|$$

Im 1. Schritt also $|a_{j1}| \geq |a_{i1}| \quad \forall i = 1, \ldots, n$
Wir erhalten damit dann, daß $|\ell_{ij}| \leq 1$

Die Schwierigkeit ist, daß durch Multiplikation der Zeilen des LGS mit bel. Zahlen die Pivotwahl bel. geändert wird.

3.12 Algorithmus (Zeilenäquilibierung)
Wir multiplizieren die i-te Zeile von A mit

$$s_i = \frac{1}{\sum_{\ell=1}^{n} |a_{i\ell}|}$$

wobei der Nenner die Zeilensumme der Beträge ist.
Damit ergibt sich als Kriterium für die Pivotwahl:

1. 1. Schritt: $|a_{j1}| \cdot s_j \geq |a_{i1}s_i| \quad \forall i$
2. $k+1$. Schritt: $\left|a^{(k)}_{j,k+1}\right| s_j \geq \left|a^{(k)}_{i,k+1}\right| s_i$

3.13 Bemerkung (Speicherung der Variablen)
Man speichert nun von der Matrix A alle Variablen, dann die ℓ_{i1}, usw. bis man eine Matrix erhält, auf der man auf der unteren Dreiecksmatrix das L speichert, allerdings ohne die Diagonale und überall darüber die Matrix R als obere Dreiecksmatrix.
Außerdem merkt man sich die Zeilenvertauschungen mit $p(k) = m$, falls im k-ten Schritt mit m-ten Zeile vertauscht.

3.3 Cholesky-Verfahren für symmetrische positiv definite Matrizen

3.14 Definition (symmetrische positive Definitheit)

1. A ist genau dann symmetrisch, wenn $A = A^T$, d.h. $a_{ij} = a_{ji}$
2. A ist genau dann positiv definit, wenn $x^T Ax > 0 \quad \forall 0 \neq x \in \mathbb{R}^n$

3.15 Satz (Existenz der Cholesky-Zerlegung)
Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch positiv definit. Dann gilt:

1. Die Gauß-Elimination kann *ohne* Zeilenvertauschungen durchgeführt werden.
2. Für die Zerlegung $A = LR$ gilt $R = DL^T$ mit D eine Diagonalmatrix mit echt positiven Elementen und damit $A = LDL^T$

Beweis

1. $A = (a_{ij})_{i,j=1}^n$. Dann können wir $a_{11} = e_1^T A e_1 > 0$ schreiben. Damit können wir a_{11} als Pivot-Element nehmen.
   Damit ergibt sich für A folgende Gestalt, wobei $z \in \mathbb{R}^{n-1}$ und C eine symmetrische positiv definite Matrix aus $\mathbb{R}^{n-1\times n-1}$ ist:
   $$A = \begin{pmatrix} a_{11} & z^T \\ z & C \end{pmatrix}, \qquad A^{(1)} = \begin{pmatrix} a_{11} & z^T \\ 0 & C^{(1)} \end{pmatrix}$$
   Hierbei ist $C^{(1)}$ symmetrisch, denn
   $$c_{ij}^{(1)} = a_{ij} - \underbrace{\frac{a_{i1}}{a_{11}}}_{\ell_{i1}} a_{1j} = a_{ji} - \frac{a_{j1}}{a_{11}} a_{1i} = c_{ji}^{(1)}$$
   Außerdem ist $C^{(1)}$ positiv definit. Für $y \in \mathbb{R}^{n-1}$ gilt:
   $$0 < \begin{pmatrix} x_1 \\ y \end{pmatrix}^T A \begin{pmatrix} x_1 \\ y \end{pmatrix} = a_{11}x_1^2 + 2x_1 z^T + y^T C y > 0 \quad \forall \begin{pmatrix} x_1 \\ y \end{pmatrix} \neq 0$$
   Aber $C^{(1)} = C - \frac{1}{a_{11}} z z^T$, aber $z \cdot z^T = (a_{i1}a_{ij})_{i,j=2}^n$.
   Nun zeigen wir, daß $C^{(1)}$ positiv definit ist:
   $$y^T C^{(1)} y = y^T C y - \frac{1}{a_{11}} (y^T z)^2$$
   Mit $x_i = -\frac{y^T z}{a_{11}}$ ist
   $$y^T C^{(1)} y = \begin{pmatrix} x_1 \\ y \end{pmatrix}^T A \begin{pmatrix} x_1 \\ y \end{pmatrix} > 0 \quad \forall y \neq 0$$
   Diesen Schritt wiederholt man nun solange rekursiv und damit kann man den Gauß-Algorithmus benutzen, ohne Zeilen zu vertauschen.

2. Betrachte $\ell_{i1} = \frac{a_{i1}}{a_{11}} = \frac{a_{i1}}{r_{11}}$ mit $r_{1i} = a_{1i} = a_{i1}$
   Damit $\ell_{i2} = \frac{a_{i2}^{(1)}}{a_{22}^{(1)}} = \frac{a_{i2}^{(1)}}{r_{22}}$ mit $r_{2i} = a_{2i}^{(1)} = a_{i2}^{(1)}$
   Im Allgemeinen gilt dann:
   $\ell_{ij} = \frac{r_{ji}}{r_{jj}} \quad \forall i, j$. Dies kann man aber auch schreiben als: $r_{ji} = r_{jj} \cdot \ell_{ij}$
   Und $r_{jj} = a_{jj}^{(j-1)} > 0$ wegen der positiven Definitheit. Und damit ergibt sich sich die Behauptung. □

3.16 Bemerkung
Bei positiv definiten Matrizen wird die Gauß-Elimination ohne Zeilenvertauschungen durchgeführt, da sonst die Symmetrie der Matrix zerstört werden würde.

3.17 Bemerkung (Cholesky-Zerlegung)
Wegen $d_i = r_{ii} > 0$ ist $D = D^{1/2} \cdot D^{1/2}$ mit den Wurzeleinträgen auf der Hauptdiagonalen.
Damit erhält man mit $\tilde{L} = LD^{1/2}$:

$$A = LDL^T = LD^{1/2} \cdot D^{1/2}L^T = \tilde{L} \cdot \tilde{L}^T$$

$\tilde{L}$ ist eine untere Dreiecksmatrix mit den Diagonalelementen $\ell_{ii}\sqrt{d_i}$.
Dies nennt man die *Cholesky-Zerlegung* der Matrix A.

3.18 Algorithmus (Cholesky-Zerlegung)
Wir bezeichnen im folgenden die Einträge in $\tilde{L}$ mit ℓ_{ij}.
Wir können also die Matrix A schreiben als:

$$\begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix} = \begin{pmatrix} \ell_{11} & \dots & 0 \\ \vdots & \ddots & 0 \\ \ell_{n1} & \dots & \ell_{nn} \end{pmatrix} \begin{pmatrix} \ell_{11} & \dots & \ell_{n1} \\ 0 & \ddots & \vdots \\ 0 & \dots & \ell_{nn} \end{pmatrix}$$

Damit erhalten wir in der 1. Spalte:

- $i = 1$: $0 < a_{11} = \ell_{11}^2 \Rightarrow \ell_{11} = \sqrt{a_{11}}$
- $i > 1$: $a_{i1} = \ell_{i1}\ell_{11} \Rightarrow \ell_{i1} = \frac{a_{i1}}{\ell_{11}}$

Allgemein gilt für die k-te Spalte:

- $i = k$: $a_{kk} = \ell_{k1}^2 + \ell_{k2}^2 + \ldots + \ell_{k,k-1}^2 + \ell_{kk}^2$. Damit erhält man

$$\ell_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} \ell_{kj}^2}$$

- $i > k$: $a_{ik} = \ell_{i1}\ell_{k1} + \ldots + \ell_{i,k-1}\ell_{k,k-1} + \ell_{ik}\ell_{kk}$. Damit erhält man

$$\ell_{ik} = \frac{a_{ik} - \sum_{j=1}^{k-1} \ell_{ij}\ell_{kj}}{\ell_{kk}}$$

Für die Speicherung überschreibt L dann die Ursprungsmatrix A.

3.19 Proposition (Rechenaufwand)
Wir haben n Wurzeln, was aber vernachlässigbar ist.
Dann haben wir

$$\sum_{k=1}^{n} (k + (n-k) + (k-1)(n-k))$$

Multiplikationen oder Divisionen und ebensoviele Additionen, das sind ungefähr:

$$\sum_{k=1}^{n} k(n-k+1) = n^3 \frac{1}{n} \sum_{k=1}^{n} \frac{k}{n} \left(1 - \frac{k-1}{n}\right) \approx n^3 \int_0^1 x(1-x)dx \approx \frac{n^3}{6}$$

Dies sind etwa halb so viele Rechenoperationen wie bei der Gauß-Elimination.

3.20 Bemerkung (Lösen des LGS)
Wir lösen dann $Ax = b$ folgendermaßen:

1. $A = \tilde{L}\tilde{L}^T$ mit Cholesky mit $\frac{n^3}{6}$ Operationen
2. $\tilde{L}y = b$ mit $\frac{n^2}{2}$ Operationen
3. $\tilde{L}^T x = y$ mit $\frac{n^2}{2}$ Operationen

3.21 Frage

1. Wie wirken sich Störungen in A und b auf die Lösung von $Ax = b$ aus? (Kondition des Problems)
2. Wie wirken sich Rundungsfehler auf die berechnte Lösung aus? (Stabilität des Algorithmus)

Dazu benötigen wir einige Hilfsmittel.

3.4 Einschub: Matrixnormen

3.22 Definition (Matrixnorm)
Sei $\|.\|$ eine Norm auf $\mathbb{R}^n$ bzw. $\mathbb{R}^m$ und sei $A \in \mathbb{R}^{n \times n}$. Dann ist allgemein

$$\|A\| = \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|}$$

3.23 Beispiel (Spezialfälle)
Wir können speziell $\|.\|_p$ für $p = 1, 2, \infty$ definieren:

$$\|A\|_p = \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}$$

3.24 Satz (Matrixnormen)
Für $A \in \mathbb{R}^{m\times n}$ ist

$$\|A\|_1 = \max_{j=1,\dots,n} \sum_{i=1}^{m} |a_{ij}| \qquad \text{max. Spaltenbetragssumme}$$

$$\|A\|_\infty = \max_{i=1,\dots,m} \sum_{j=1}^{n} |a_{ij}| \qquad \text{max. Zeilenbetragssumme}$$

$$\|A\|_2 = \sqrt{\text{größter EW von } A^TA} \qquad \text{Spektralnorm}$$

Beweis

1. Analysis 2
2. ebenso
3. A^TA ist symmetrisch und positiv semidefinit.
   $x^TA^TAx = (Ax)^T(Ax) = \|Ax\|_2^2 \geq 2$, d.h. es gibt eine orthogonale Matrix Q mit $Q^TA^TAQ =$ Diagonalmatrix mit den EW auf der Diagonalen.
   Damit gilt: $\|Ax\|_2^2 = x^TA^TAx$ und für $x = Qx$ ergibt sich:

$$\begin{aligned}\|Ax\|_2^2 = y^TQ^TA^TAQy = \operatorname{diag}(\lambda_i) = \sum_{i=1}^{n} \lambda_i y_i^2 \leq \lambda_{\max} \sum_{i=1}^{n} y_i^2 \\ = \lambda_{\max} y^Ty = \lambda_{\max} x^TQQ^Tx = \lambda_{\max} x^Tx = \lambda_{\max}\|x\|_2^2\end{aligned}$$

   Und damit

$$\frac{\|Ax\|_2}{\|x\|_2} \leq \sqrt{\lambda_{\max}} \quad \forall x \in \mathbb{R}^n$$

   Damit ist

$$\|A\|_2 = \sup_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2} \leq \sqrt{\lambda_{\max}}$$

   Doch wie bekommen wir eine Gleichheit? Wir wählen y so, daß in der Abschätzung eine Gleichheit steht, d.h. y wird so gewählt, daß überall 0 und an einer Stelle eine 1 steht. □

3.5 Kondition einer Matrix

3.25 Motivation
Wir haben ein LGS $Ax = b$ und ein LGS mit gestörten Werten $\tilde{A}\tilde{x} = \tilde{b}$.
Wie können wir

$$\frac{\|x-\tilde{x}\|}{\|x\|} \leq \dots \left(\frac{\|A-\tilde{A}\|}{\|A\|} + \frac{\|b-\tilde{b}\|}{\|b\|} \right)$$

abschätzen, d.h. wie sehr wirken sich Störungen in den Werten auf die Lösung aus?

3.26 Satz (Kondition)
Sei A invertierbar, $Ax = b$ und $\tilde{A}\tilde{x} = \tilde{b}$. Seien

$$\frac{\|\tilde{A} - A\|}{\|A\|} \leq \varepsilon_A, \qquad \frac{\|\tilde{b} - b\|}{\|b\|} \leq \varepsilon_b$$

Dann gilt:

$$\frac{\|x - \tilde{x}\|}{\|x\|} \leq \frac{\operatorname{cond}(A)}{1 - \varepsilon_A \cdot \operatorname{cond}(A)} \cdot (\varepsilon_A + \varepsilon_b)$$

wobei $\operatorname{cond}(A) := \|A\| \cdot \|A^{-1}\|$ die *Konditionszahl* der Matrix A ist und sofern $\varepsilon_A \operatorname{cond}(A) < 1$ ist.

Beweis
Gelte $Ax = b$ und $\tilde{A}\tilde{x} = \tilde{b}$. Damit erhalten wir durch Subtraktion:

$$Ax(-A\tilde{x} + A\tilde{x}) - \tilde{A}\tilde{x} = b - \tilde{b} \iff A(x - \tilde{x}) + (A - \tilde{A})\tilde{x} = b - \tilde{b}$$

Aufgelöst ergibt sich:

$$x - \tilde{x} = -A^{-1}\left((A - \tilde{A})\tilde{x} - (b - \tilde{b})\right)$$

Wegen $\|A^{-1}v\| \leq \|A^{-1}\| \cdot \|v\|$ gilt:

$$\begin{aligned} \|x - \tilde{x}\| &\leq \|A^{-1}\| \cdot (\|A - \tilde{A}\| \cdot \|\tilde{x}\| + \|b - \tilde{b}\|) \\ &\leq (\varepsilon_A \cdot \|A\|)(\|x\| + \|x - \tilde{x}\|) + \underbrace{\varepsilon_b \|b\|}_{=\varepsilon_b \|Ax\|} \end{aligned}$$

$$(1 - \varepsilon_A \operatorname{cond}(A))\|x - \tilde{x}\| \leq \operatorname{cond}(A) \cdot \|x\|(\varepsilon_A + \varepsilon_b)$$

womit die Behauptung gilt. □

3.27 Proposition (Eigenschaften der Konditionszahl)
Da die Störung der Lösung von der Konditionszahl der Matrix abhängt, sind uns einige Eigenschaften sehr wichtig:

1. $\operatorname{cond}(A) \geq 1$
2. $\operatorname{cond}(\alpha A) = \operatorname{cond}(A) \quad \forall \alpha \in \mathbb{R}^{\bullet}$

3. Es gilt:

$$\operatorname{cond}(A) = \frac{\max_{\|y\|=1} \|Ay\|}{\min_{\|z\|=1} \|Az\|}$$

BEWEIS

1. Betrachte

$$1 = \|I\| = \|AA^{-1}\| \leq \|A\| \cdot \|A^{-1}\|$$

2. Klar

3. Ohne Beweis, s. Übungsaufgabe □

3.28 Beispiel (Konditionen bekannter Matrizen)

1. Sei $A = \left(\frac{1}{i+j-1}\right)_{(i,j)\in(n\times n)}$, d.h.

$$A = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \dots \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \dots \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \dots \\ \vdots & & & & \end{pmatrix}$$

Wir erhalten für diese Matrix für versch. n folgende Konditionen:

n	1	2	3	4	...	10
$\operatorname{cond}(A)$	1	27	748	2300	...	$3,5 \cdot 10^{13}$

Dies ist eine sog. HILBERT-Matrix

2. $\operatorname{cond}(I) = \|I\| \cdot \|I\| = 1 \cdot 1 = 1$

3. Sei Q eine orthogonale Matrix, d.h. $Q^TQ = QQ^T = I$. Damit gilt:

$$\|Qx\|_2^2 = (Qx)^T(Qx) = x^TQ^TQx = x^Tx = \|x\|_2^2$$

und damit

$$\operatorname{cond}_2(Q) = \|Q\|_2 \cdot \|Q^{-1}\|_2 = \|Q\|_2 \cdot \|Q^T\|_2 = 1 \cdot 1 = 1$$

4. Betrachte die Matrix für die Splines bei äquidistanter Unterteilung. Die Kondition ist unabhängig von der Schrittweite h.
Es gilt $\|A\| = 6$ und $A = 4(I+N)$ mit $\|N\| = \frac{1}{2} < 1$ und damit
$A^{-1} = \frac{1}{4}(I+N)^{-1} = \frac{1}{4}\left(I - N + N^2 - N^3 + \dots\right)$ wegen der geometrischen Reihe:
$\frac{1}{1+x} = \sum_{n=0}^{\infty}(-1)^n x^n$ für $|x| < 1$

Damit erhalten wir:

$$\|A^{-1}\| \leq \frac{1}{4}\left(\|I\| + \|N\| + \|N^2\| + \|N^3\|\right) \leq \frac{1}{4}\frac{1}{1-\|N\|} = \frac{1}{4}\frac{1}{1-\frac{1}{2}} = \frac{1}{2}$$

und damit $\operatorname{cond}(A) = \|A\| \cdot \|A^{-1}\| \leq 3$

3.6 Stabilität der Gauß-Elimination

3.29 Definition (Numerische Stabilität)
Sei x die exakte Lösung (hier $Ax = b$) und $\hat{x}$ die mit dem Algorithmus berechnete Lösung (inkl. allen Rundungsfehlern).
Der Algorithmus heißt *numerisch stabil*

1. im Sinne der *Vorwärtsanalyse*, falls

   $$\frac{\|\hat{x} - x\|}{\|x\|} \leq C \cdot \mathrm{cond}(A) \cdot \mathtt{eps}$$

   mit nicht allzu großem C, d.h. der Einfluß von Rundungsfehlern während der Rechnung ist nicht viel größer als der zu erwartende Einfluß von Rundunfsfehlern in den Daten $(A, b) \to x$.

2. im Sinne der *Rückwärtsanalyse*, falls das numerische Ergebnis $\hat{x}$ interpretiert werden kann als exakte Lösung einer Gleichung mit gestörten Daten $\hat{A}\hat{x} = \hat{b}$ mit

   $$\frac{\|\hat{A} - A\|}{\|A\|} \leq C \cdot \mathtt{eps}, \qquad \frac{\|\hat{b} - b\|}{\|b\|} \leq C \cdot \mathtt{eps}$$

   mit nicht allzu großen C.

Man braucht dazu nicht $\mathrm{cond}(A)$ zu kennen.

3.30 Notation
Wir bezeichnen im Folgenden $A = (a_{ij})$ und $B = (b_{ij})$
Dann gilt $A \leq B \iff a_{ij} \leq b_{ij} \quad \forall i, j$ und $|A| = (|a_{ij}|)$

3.31 Satz (Rundungsfehler beim Gauß-Algorithmus)
Wir untersuchen den Rundungsfehler bei der Zerlegung $PA = LR$ und nehmen an, daß die Zeilenvertauschungen bereits zu Beginn durchgeführt wurden und ersetzen A durch PA.
Sei $A \in \mathbb{R}^{n \times n}$ eine Matrix von Gleitpunktzahlen.
Falls bei der LR-Zerlegung von A keine Nullpivots auftreten, dann gilt für die in Gleitpunktrechnung durch Gauss-Elimination erhaltenen gestörten Matrizen $\hat{L}$ und $\hat{R}$:

$$|A - \hat{L}\hat{R}| \leq (n+3)|\hat{L}| \cdot |\hat{R}| \cdot \mathtt{eps} + \mathcal{O}(\mathtt{eps}^2)$$

sofern $n|\hat{L}| \cdot |\hat{R}| \ll \frac{1}{\mathrm{eps}}$ ist.

Beweis
Induktion über n für $n = 1$ ist der Fall klar.
Sei die Behauptung wahr für $(n-1) \times (n-1)$ als Gleitpunktmatrix.
Dann ist $A = \begin{pmatrix} \alpha & w^T \\ v & B \end{pmatrix}$

Die Gauß-Elimination berechnet $z = \frac{v}{\alpha}$ und $A_1 = B - zw^T$
Bei der Gleitpunktrechnung erhalten wir die Werte $\hat{z}$ und $\hat{A}_1$.
$\hat{i} = \frac{v_i}{\alpha}(1+\varepsilon_i)$ mit $|\varepsilon_i| \leq \texttt{eps}$ und damit erhalten wir

$$|\hat{z}_i - z_i| \leq |\varepsilon_i| \cdot |z_i| \quad \forall i, \qquad |\hat{z} - z| \leq \texttt{eps} \cdot |z|$$

Damit ergibt sich

$$(\hat{A}_1)_{ij} = (b_{ij} - \hat{z}_i \cdot w_j(1+\varepsilon_{ij}))\,(1+\varepsilon'_{ij}), \qquad |\varepsilon_{ij}|, |\varepsilon'_{ij}| \leq \texttt{eps}$$

und damit

$$\left|(\hat{A}_1)_{ij} - (A_1)_{ij}\right| \leq \texttt{eps} \cdot (|b_{ij}| + 3 \cdot |z_i| \cdot |w_j|) + \mathcal{O}(\texttt{eps}^2)$$

d.h.

$$|\hat{A}_1 - A_1| \leq \texttt{eps}\left(|B| \cdot |B| \cdot |z| \cdot |w^T|\right) + \mathcal{O}(\texttt{eps}^2)$$

und mit $B = A_1 + zw^T$ ergibt sich:

$$|\hat{A}_1 - A_1| \leq \texttt{eps} \cdot \left(|A_1| + 4|z||w^T|\right) + \mathcal{O}(\texttt{eps}^2)$$

mit der Induktionsannahme für $\hat{A}_1$ ergibt sich:

$$|\hat{A}_1 - \hat{L}_1\hat{R}_1| \leq (n+2)|\hat{L}_1| \cdot |\hat{R}_1| + \mathcal{O}(\texttt{eps}^2)$$

Wir haben also:

$$\hat{L}\hat{R} = \begin{pmatrix} 1 & 0 \\ \hat{z} & \hat{L}_1 \end{pmatrix} \begin{pmatrix} \alpha & w^T \\ 0 & \hat{R}_1 \end{pmatrix} = \begin{pmatrix} \alpha & w^T \\ \alpha\hat{z} & \hat{z}w^T + \hat{L}_1\hat{R}_1 \end{pmatrix}$$

und für die ursprüngliche Matrix ergibt sich:

$$A = LR = \begin{pmatrix} 1 & 0 \\ z & L_1 \end{pmatrix} \begin{pmatrix} \alpha & w^T \\ 0 & R_1 \end{pmatrix} = \begin{pmatrix} \alpha & w^T \\ \alpha z & zw^T + L_1R_1 \end{pmatrix}$$

Für die Differenz ergibt sich:

$$A - \hat{L}\hat{R} = \begin{pmatrix} 0 & 0 \\ \alpha(z-\hat{z}) & (z-\hat{z})w^T + (A_1 - \hat{A}_1) + (\hat{A}_1 - \hat{L}_1\hat{R}_1) \end{pmatrix}$$

mit den obigen Abschätzungen ergibt sich:

$$\begin{aligned}|A - \hat{L}\hat{R}| &\leq \texttt{eps} \cdot \begin{pmatrix} 0 & 0 \\ |\alpha| \cdot |z| & |z||w^T| + |A_1| + 4|z||w^T| + (n+2)|\hat{L}_1||\hat{R}_1| \end{pmatrix} + \mathcal{O}(\texttt{eps}^2) \\ &\leq (n+3)\texttt{eps} \underbrace{\begin{pmatrix} |\alpha| & |w^T| \\ |\alpha||z| & |z|w^T| + |\hat{L}_1|\hat{R}_1| \end{pmatrix}}_{=|\hat{L}|\cdot|\hat{R}|} + \mathcal{O}(\texttt{eps}^2)\end{aligned}$$

womit die Abschätzung bewiesen ist. □

3.32 Bemerkung
Können $\hat{L}$ und $\hat{R}$ groß werden? Wir nehmen an, daß bei Spaltenpivotsuche $|\ell_{ij}| \leq 1 \; \forall i, j$ ist, dann gilt:

$$\max_{i,j} |r_{ij}| \leq 2^{n-1} \max_{i,j} |a_{ij}|$$

für gewisse konstruierte Beispiele.
Für zufällige gewählte Matrizen A beobachten wir dann $\max_{i,j} |r_{ij}| \approx n \cdot \max_{i,j} |a_{ij}|$.

3.33 Satz (Fehlerabschätzung beim Lösung der LGS)
Seien L, R eine untere bzw. obere Dreiecksmatrix von Gleitpunktzahlen und b, c seien Vektoren von Gleitpunktzahlen.
Die in Gleitpunktrechnung erhaltenen Ergebnisse $\hat{x}$ und $\hat{y}$ für die linearen Gleichungssysteme $Ly = b, Rx = c$ sind die exakten Lösungen von gestörten Gleichungen $\hat{L}\hat{y} = b$ und $\hat{R}\hat{x} = c$ mit

$$|\hat{L} - L| \leq n|L| \cdot \texttt{eps}, \qquad |R - \hat{R}| \leq n \cdot |R| \cdot \texttt{eps}$$

Beweis
s. Übungsblatt □

3.34 Satz (Fehler des Lösungsvektors bei der LR-Zerlegung)
Seien $\hat{L}$ und $\hat{R}$ wie im vorherigen Satz erhalten.
Das in Gleichpunktrechnung erhalte Ergebnis $\hat{\hat{x}}$ von $\hat{L}\hat{c} = b$ und $\hat{R}\hat{x} = \hat{c}$ gilt $\hat{\hat{A}}\hat{\hat{x}} = b$ mit

$$|A - \hat{\hat{A}}| \leq 3(n+2)|L| \cdot |R| \cdot \texttt{eps} + \mathcal{O}(\texttt{eps}^2)$$

Beweis
Ohne Rundungsfehler erhalten wir: $A = LR$, $Lc = b$ und $Rx = c$ und damit die Lösung des LGS $Ax = b$.

Statt L, R haben wir allerdings gestörte Matrizen $\hat{L}, \hat{R}$.
Wir erhalten $\hat{\hat{x}}$ als exakte Lösung von gestörten Gleichungssystemen:

$$\hat{\hat{L}}\hat{c} = b \qquad \text{mit} \quad |\hat{\hat{L}} - \hat{L}| \leq n \cdot \texttt{eps}|\hat{L}|$$

und

$$\hat{\hat{R}}\hat{\hat{x}} = \hat{c} \qquad \text{mit} \quad |\hat{\hat{R}} - \hat{R}| \leq n \cdot \texttt{eps}|\hat{R}|$$

Wir setzen nun $\hat{\hat{A}} = \hat{\hat{L}}\hat{\hat{R}}$ und damit $\hat{\hat{A}}\hat{\hat{x}} = b$ und damit:

$$\begin{aligned}
|\hat{\hat{A}} - A| &= \left|\hat{\hat{L}}\hat{\hat{R}} - \hat{\hat{L}}\hat{R} + \hat{\hat{L}}\hat{R} - \hat{L}\hat{R} + \hat{L}\hat{R} - LR\right| \\
&\leq |\hat{\hat{L}}| \cdot |\hat{\hat{R}} - \hat{R}| + |\hat{\hat{L}} - \hat{L}| \cdot |\hat{R}| + |\hat{L}\hat{R} - LR| \\
&\leq |\hat{L}| n \cdot \text{eps}|\hat{R}| + n \cdot \texttt{eps}|\hat{L}||\hat{R}| + (n+3)\texttt{eps}|\hat{L}||\hat{R}| + \mathcal{O}(\texttt{eps}^2) \\
&\leq 3(n+1)\texttt{eps}|\hat{L}||\hat{R}| + \mathcal{O}(\texttt{eps}^2)
\end{aligned}$$

womit die Abschätzung gilt. □

3.35 Bemerkung
Für A symmetrisch und positiv definit erhalten wir für die Cholesky-Zerlegung $A = LL^T$ auf ähnliche Weise:

$$|\hat{L}\hat{L}^T - A| \leq (n+3)\texttt{eps}|\hat{L}| \cdot |\hat{L}^T| + \mathcal{O}(\texttt{eps}^2)$$

Wir sehen sogar, daß $|\hat{L}|$ nicht groß werden kann, denn mit $A = LL^T$ ergibt sich:

$$a_{ii} = \sum_{k=1}^{i} \ell_{ik}^2 \geq \ell_{ij}^2 \quad \forall i, j$$

Damit gilt für $a = \max_i a_{ii} = \max_{i,j} |a_{ij}|$:

$$|\ell_{ij}| \leq \sqrt{a}$$

3.7 QR-Zerlegung mittels Householder-Transformation

3.36 Motivation
Im Folgenden sei $A \in \mathbb{R}^{m \times n}$ mit $m \geq n$ und wir wollen eine Zerlegung der Art

$$A = QR$$

konstruieren mit $Q \in \mathbb{R}^{m \times m}$ orthogonal, d.h. $Q^{-1} = Q^T$ und

$$R = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}$$

und $\tilde{R}$ eine obere Dreiecksmatrix.

3.37 Bemerkung (Anwendungen der QR-Zerlegung)

1. Für $m = n$ kann man das LGS $Ax = b$ betrachten. Dann gilt für $A = QR$:

   $$Qc = b, \qquad Rx = c$$

   Wir lösen hierbei die erste Gleichung mit $c = Q^T b$ auf und die zweite wie gehabt durch Substitution von unten nach oben.

   Dies liefert einen besonders stabilen Algorithmus für das lineare Gleichungssystem, aber doppelt so rechenaufwändig wie die Gauß-Elimination.

2. Für $m > n$ haben wir statt $Ax = b$ (m Gleichungen in $n < m$ Unbekannten), d.h. wir möchten $\|Ax - b\|_2$ minimal erhalten.

   Diese Art eines LGS nennt man lineare Ausgleichgsrechnung, auf die im nächsten Kapitel eingegangen wird.

3. Der sog. QR-Algorithmus zur Berechnung von EW.

3.38 Konstruktion (Householder-Transformation)
Nehme Matrizen der Form $Q = I - 2uu^T$ mit $u \in \mathbb{R}^m$ und Dabei ist $u^T u = \|u\|_2^2 = 1$.
Diese Matrix ist eine HOUSEHOLDER-Transformation und hat folgende Eigenschaften:

1. Q ist eine Spiegelung an der Hyperebene $\{x | u^T x = 0\}$, denn:

   Sei $x = \alpha u + v$ mit $\alpha \in \mathbb{R}, u^T v = 0$.
   Dann ist $Qx = \alpha u + v - 2uu^T u\alpha - 2uu^T v = -\alpha u + v$.

2. Q ist symmetrisch, denn $Q^T = Q$

3. Q ist orthogonal, denn $\forall x \quad Q^T Qx = Q^2 x = x$. Damit ist $Q^T Q = I$

3.39 Konstruktion (Erster Schritt der QR-Zerlegung)
Sei

$$a_1 = \begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{pmatrix}$$

die erste Spalte von A.
Wir suchen eine Householder-Matrix $Q_1 = I - 2u_1 u_1^T$, sodaß gilt:

$$Q_1 a_1 = \begin{pmatrix} \alpha_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Weil Q_1 orthogonal, muß gelten:

$$\underbrace{\|Q_1 a_1\|_2^2}_{\alpha_1^2} = \|a_1\|_2^2$$

Damit muß gelten:

$$\alpha_1 = \pm\|a_1\|_2$$

Wir möchten, daß $Q_1 a_1 = \alpha_1 e_1$ und damit erhalten wir:

$$(I - 2u_1 u_1^T)a_1 = a_1 - 2u_1 \underbrace{u_1^T a_1}_{\in\mathbb{R}}$$

Damit muß u_1 ein Vielfaches sein von $a_1 - \alpha_1 e_1$. Andererseits muß ja gelten, daß $\|u_1\|_2 = 1$. Wir erhalten damit:

$$u_1 = \frac{a_1 - \alpha_1 e_1}{\|a_1 - \alpha_1 e_1\|_2}$$

mit

$$\|a_1 - \alpha_1 e_1\|_2^2 = \underbrace{\|a_1\|_2^2}_{\alpha_1^2} - 2\alpha_1 a_{11} + \alpha_1^2 = 2\alpha_1(\alpha_1 - a_{11})$$

Das Vorzeichen von α_1 ist so zu wählen, daß es genau das entgegengesetzte Vorzeichen von a_{11} hat. Damit gibt es keine Stellenauslöschung bei der Berechnung dieser Differenz.
Damit:

$$\alpha_1 = -\operatorname{sgn}(a_{11}) \cdot \|a_1\|_2$$

Es ergibt sich damit durch $Q_1 A = A'$ eine Matrix, die in der ersten Spalte $\begin{pmatrix} \alpha_1 \\ 0 \\ \vdots \end{pmatrix}$ hat in der ersten Zeile bel. Einträge und eine Restmatrix $A^{(1)}$.
Wir bezeichnen mit a_j bzw. $a_j^{(1)}$ die j-te Spalte von A bzw. $A^{(1)}$ mit $v_1 = a_1 - \alpha_1 e_1$ gilt dann für $j \geq 2$:

$$\alpha_j' = Q_1 \alpha_j = \alpha_j - \underbrace{\frac{2v_1^T a_j}{v_1^T v_1}}_{\in\mathbb{R}} v_1$$

wobei zu beachten ist, daß

$$\frac{v_1^T v_1}{2} = \frac{1}{2}\|a_1 - \alpha_1 e_1\|^2 = \alpha_1(\alpha_1 - a_{11})$$

Um $Q_1 A$ zu berechnen, braucht man nur v_1 und α_1 zu kennen. Man muß nicht die Matrix $Q_1 = I - 2u_1 u_1^T$ berechnen.

3.40 Konstruktion (Wiederholte Householder-Transformation)
Wir erhalten dann eine Matrix

$$Q_2 = \begin{pmatrix} 1 & 0 \\ 0 & \bar{Q}_2 \end{pmatrix}$$

und wir bestimmen $\bar{Q}_2 = I_{m-1} - 2u_2u_2^T$ und $u_2 \in \mathbb{R}^{m-1}$ mit $u_2^T u_2 = 1$ so, daß

$$\bar{Q}_2 a_1^{(1)} = \begin{pmatrix} \alpha_2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Wir erhalten damit eine Restmatrix:

$$A'' = Q_2 A' = \begin{pmatrix} \alpha_1 & * & \dots & * \\ 0 & \alpha_2 & * & * \\ 0 & 0 & A^{(2)} & \end{pmatrix}$$

im k-ten Schritt erhält man dann für

$$Q_k = \begin{pmatrix} I_{k-1} & 0 \\ 0 & \bar{Q}_k \end{pmatrix}$$

und die Bedingung, daß

$$\bar{Q}_k a_1^{(k-1)} = \begin{pmatrix} \alpha_k & \\ 0 & \vdots \\ 0 & \end{pmatrix}$$

Das Ergebnis der Umformung nach dem n-ten Schritt ist dann:

$$\underbrace{Q_n \cdot \ldots \cdot Q_2 Q_1}_{=:Q^T} A = R$$

Das Produkt von orthogonalen Matrizen $Q_{n-1} \cdot \ldots \cdot Q_1$ ist wieder orthogonal.
Damit ist $A = QR$, wobei Q orthogonal ist und R ist die obere Dreiecksmatrix.

3.41 Algorithmus (QR-Zerlegung)
Wir können $A = QR$ zerlegen mit $Q = Q_{n-1} \cdot \ldots \cdot Q_1$, A hat die Dimension $m \times n$ mit $m \geq n$.

Für die Speicherung benötigt man für A ein (m, n)-Feld.
Zudem braucht man einen u-Vektor für $\alpha_1, \ldots, \alpha_n$.

Man fängt bei der Matrix A an, überschreibt die erste Spalte mit dem Vektor v_1 und speichert zusätzlich das α_1.
Dann überschreibt man in Treppenform die zweite Spalte (d.h. ohne das erste Element) mit v_2, usw.
Man erhält dann eine untere Dreiecksmatrix mit den v_i und die obere Dreiecksmatrix R ohne die Digaonale, die aber in den α_i gespeichert ist.

3.42 Proposition (Rechenaufwand)

1. Man berechnet die Norm von A mit $\|a_1\|_2$ für $j = 2, \ldots, n$ die $a_j^{(1)}$.
   Dies sind $m + (n-1)(2m+1) \approx 2mn$ Operationen.

   Man erhält als Gesamtaufwand an Rechenoperationen:

   $$2mn + 2(m-1)(n-1) + \ldots + 2(m-n+1) \cdot 1$$

   Für den Spezialfall: $m = n$ ist der Aufwand etwa $\frac{2}{3}n^3$ Schritte. (Dies ist der doppelte Aufwand wie bei der Gauß-Elimination, jedoch ist dieser Algorithmus sehr stabil).
   Für den Spezialfall $m \gg n$ erhält man ungefähr $2m(n + (n-1) + \ldots + 1) \approx mn^2$ Operationen.

3.43 Bemerkung (Stabilität und Rundungsfehleranalyse)
Da $A = QR$ und Q ein orthogonale Matrix ist, ist $\|A\|_2 = \|QR\|_2 = \|R\|_2$, d.h. die Einträge in R können nicht groß werden.
Für das berechnte $\hat{Q}$ gilt:

$$\|\hat{Q}^T\hat{Q} - I\|_2 \leq c_{mn} \cdot \texttt{eps}$$

d.h. $\hat{Q}$ ist fast orthogonal.
Es gilt daher:

$$\|A - \hat{Q}\hat{R}\|_2 \leq c'_{mn}\|A\|_2 \cdot \texttt{eps}$$

3.8 Lineare Ausgleichsrechnung

3.44 Motivation
Gegeben ist eine Matrix $A \in \mathbb{R}^{m \times n}$ mit $m > n$ (oft ist $m \gg n$) und $b \in \mathbb{R}^m$.

Das LGS $Ax = b$ mit m Gleichungen in n Unbekannten ist somit überbestimmt.
Wir möchten stattdessen $Ax - b$ „möglichst klein“ machen, genauer:

Suche ein $x \in \mathbb{R}^n$, sodaß $\|Ax - b\|_2$ minimal ist.
Dies ist die lineare Ausgleichsrechnung mit der euklidischen Norm.

3.45 Konstruktion (Methode der kleinsten Fehlerqudrate)
Ein typisches Beispiel sind Messdaten (t_j, y_j) mit $j = 1, \ldots, m$, sodaß man z.B. eine Gerade durchlegen kann.

Konkret suchen wir eine Funktion

$$y = f(t) = \sum_{i=1}^{n} x_i \varphi_i(t)$$

mit $\varphi_1, \ldots, \varphi_n$ gegebene Funktionen sind (z.B. $\varphi_1(t) = 1, \varphi_2(t) = t, \varphi_3(t) = t^3$) mit den unbekannten Koeffizienten $x_1, \ldots, x_n$ unbekannt so bestimmen, daß $y_j \approx f(t_j)$.
Genauer definiert man den Fehler $e_j = y_j - f(t_j)$ erfülle

$$\left\| \begin{pmatrix} e_1 \\ \vdots \\ e_m \end{pmatrix} \right\| \text{ minimal}$$

d.h. $\sum\limits_{j=1}^{m} e_j^2$ muß minimal werden.
Dies ist die *Methode der kleinsten Fehlerquadrate*, die auf Gauss (1801) zurückgeht.

3.46 Bemerkung (Formulierung in der Matrixschreibweise)
Wir schreiben:

$$A = \begin{pmatrix} \varphi_1(t_1) & \ldots & \varphi_n(t_1) \\ \vdots & & \vdots \\ \varphi_1(t_m) & \ldots & \varphi_n(t_m) \end{pmatrix} \in \mathbb{R}^{m \times n}, \qquad b = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$$

und für die Fehler ergibt sich:

$$e_j = y_j - f(t_j) = y_j - \sum_{i=1}^{n} x_i \underbrace{\varphi_i(t_j)}_{a_{ji}} = (b - Ax)_j$$

Wir sehen also, daß gilt:

$$\sum_{j=1}^{m} e_j^2 \text{ minimal} \iff \|Ax - b\|_2 \text{ minimal}$$

3.47 Satz (Gaußsche Normalgleichungen)
Wir möchten ein $x \in \mathbb{R}^m$ finden, sodaß $\|Ax - b\|_2$ minimal ist.
Dann gilt:

$$x \text{ ist die gesuchte Lösung} \iff A^T A x = A^T b$$

3.48 Bemerkung
Die geometrische Interpretation zu $A^T(Ax-b)=0$ ist:
Sei $V = \text{Bild}(A) = \{Ax | x \in \mathbb{R}^m\}$ ein Unterraum des $\mathbb{R}^m$. Ist b ein Vektor, dann ist Ax so gewählt, daß es die orthogonale Projektion von b auf den Unterraum V ist, d.h. $v^T A^T(Ax-b)=0 \quad \forall v \in \mathbb{R}^n$

Beweis
Es gilt $\|Ax-b\|_2$ minimal $\iff$ $Ax-b$ orthogonal auf Bild A, denn:

$$\|A(x+y)-b\|_2^2 \geq \|Ax-b\|_2^2 \quad \forall y \in \mathbb{R}^n$$

Es gilt aber auch:

$$\|A(x+y)-b\|_2^2 = \|Ax-b\|_2^2 + 2(Ay)^T(Ax-b) + \underbrace{\|Ay\|_2^2}_{\geq 0}$$
$$\iff (Ay)^T(Ax-b) = y^T A^T(Ax-b) = 0 \; \forall y \in \mathbb{R}^n$$

und dies ist äquivalent zu $A^T(Ax-b)=0$, womit wir die Behauptung bewiesen haben. □

3.49 Bemerkung (Bemerkung zur Lösung der neuen Bedingung)
$A^TA \in \mathbb{R}^{n\times n}$ ist eine symmetrische, positive semidefinite Matrix und es gilt:
A^TA positiv definit $\iff$ $\text{rg}(A) = n$.

Beweis
$x^TA^TAx = 0 \iff \|Ax\|0 \iff Ax = 0$. Daher gilt:

$$\begin{aligned} A^TA \text{ positiv definit} &\iff (x^TA^TAx = 0 \Rightarrow x = 0) \\ &\iff (Ax = 0 \Rightarrow x = 0) \iff \text{rg}(A) = n \end{aligned}$$

womit die Behauptung bewiesen ist. □

3.50 Zusammenfassung (Rechenaufwand bei der Berechnung)
Falls $\text{rg}(A) = n$, gibt es verschiedene Algorithmen, um das Ergebnis zu berechnen:

1. Man berechnet A^TA (mit $\frac{1}{2}n^2m$ Operationen) und A^Tb (mit $n \cdot m$ Operationen) und löst $A^TAx = A^Tb$ mit der Cholesky-Zerlegung in $\frac{1}{6}n^3$ Operationen
2. mit der QR-Zerlegung benötigt man mn^2 Operationen.
   Wir betrachten

$$\begin{aligned} \|Ax-b\|_2^2 &= \|QRx-b\|_2^2 = \|Q(Rx-Q^{-1}b)\|_2^2 = \|Rx-Q^Tb\|_2^2 \\ &= \left\| \begin{pmatrix} \tilde{R}x \\ 0 \end{pmatrix} - \begin{pmatrix} c \\ d \end{pmatrix} \right\|_2^2 = \left\| \begin{pmatrix} \tilde{R}x-c \\ -d \end{pmatrix} \right\|_2^2 = \|\tilde{R}x-c\|_2^2 + \|d\|_2^2 \end{aligned}$$

   und damit ist $\text{rg}(A) = \text{rg}(R) = \text{rg}(\tilde{R})$ und daher ist $\tilde{R}$ invertierbar. Somit gilt:
   $\|Ax-b\|_2^2 = \min \iff \tilde{R}x = c$.

Wir sehen, daß die Berechnung mit der Cholesky-Zerlegung zwar weniger Schritte benötigt, aber die QR-Zerlegung wieder stabiler ist.

Die QR-Zerlegung benötigt etwa doppelt so viele Operationen wie die Berechnung von A^TA, aber der zweite Algorithmus ist stabiler.

3.51 Beispiel
Seien

$$A = \begin{pmatrix} 1 & 1 \\ \varepsilon & 0 \\ 0 & \varepsilon \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

Wir erhalten

$$A^TA = \begin{pmatrix} 1+\varepsilon^2 & 1 \\ 1 & 1+\varepsilon^2 \end{pmatrix}, \qquad A^Tb = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

die exakte Lösung hierdavon ist $x_1 = x_2 = \frac{1}{2+\varepsilon^2} \approx \frac{1}{2}$.
In Gleitpunktrechnung ist aber $[1+\varepsilon^2] = 1$. Damit ist A^TA singulär und der erste Algorithmus gar nicht durchführbar.

Mit der Householder-Transformation ist dann $\alpha_1 \approx -1$, $v_1 = \begin{pmatrix} 2 \\ \varepsilon \\ 0 \end{pmatrix}$ und damit:

$$R = \begin{pmatrix} -1 & -1 \\ 0 & \sqrt{2}\varepsilon \\ 0 & 0 \end{pmatrix}, \qquad Q^Tb = \begin{pmatrix} -1 \\ -\frac{\varepsilon}{\sqrt{2}} \\ -\frac{\varepsilon}{\sqrt{2}} \end{pmatrix}$$

Damit ist $\tilde{R}x = c$; die exakte Lösung $x_1 = x_2 = \frac{1}{2}$.

3.52 Motivation (für die bessere Stabilität)
Beim ersten Algorithmus lösen wir $A^TAx = A^Tb$. Dabei ist dann

$$\mathrm{cond}_2(A^TA) = \mathrm{cond}_2(A)^2 \gg \mathrm{cond}_2(A)$$

Mit dem zweiten Algorithmus löst man $\tilde{R}x = c$. Dann ist

$$\mathrm{cond}_2(\tilde{R}) = \mathrm{cond}_2(R) = \mathrm{cond}_2(QR) = \mathrm{cond}_2(A)$$

3.53 Algorithmus (falls der Rang nicht voll ist)
Für $\operatorname{rg}(A) < n$ bricht der erste Algorithmus zusammen, da A nicht invertierbar ist. Wir betrachten daher nur die QR-Zerlegung mit Spaltentausch:
In jedem Schritt der QR-Zerlegung vertauscht man die Spalten so, daß die Spalte mit der größten Norm vorne steht.
Damit erhalten wir: $AP = QR$.

Wir partitionieren auch $Px = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ und $Q^T b = \begin{pmatrix} c \\ d \end{pmatrix}$ und damit ist

$$\|Ax - b\|_2^2 = \|R_1 x_1 + R_2 x_2 - c\|_2^2 + \|d\|_2^2$$

Dann muß gelten: $R_1 x_1 + R_2 x_2 = c$, wobei R_1 invertierbar ist.
d.h. $x_2 \in \mathbb{R}^{n-k}$ ist beliebig. Daher ist die Lösung nicht eindeutig.

4 Nichtlineare Gleichungssysteme

4.1 Motivation
Wir stehen in diesem Kapitel vor folgendem Problem:
Zu einer Funktion $f : D \subseteq \mathbb{R}^n \to \mathbb{R}^n$ sucht man ein x mit $f(x) = 0$, d.h. man erhält

$$\begin{aligned} f_1(x_1, \ldots, x_n) &= 0 \\ \ldots &= 0 \\ f_n(x_1, \ldots, x_n) &= 0 \end{aligned}$$

diese n Gleichungen mit n Variablen. Hier gibt es verschiedene Möglichkeiten:

$f(x) = e^x$	keine Lösung
$f(x) = x^2 - 1$	evtl. mehrere Lösungen
$f(x) = \sin x$	unendlich viele lokal eindeutige Lösungen
$f(x) = x \cdot \sin \frac{1}{x}$	

4.1 Newton-Verfahren

4.2 Algorithmus (Prinzipielle Vorgehensweise)
Das Newton-Verfahren ist ein iteratives Verfahren, bei dem man wie folgt vorgeht:

1. Wähle einen Startwert x_0
2. Ersetze die Funktion durch die Tangente in $(x_0, f(x_0))$
3. Die Nullstelle der Tangente ist x_1

Danach werden die Schritte wiederholt.
Wir ersetzen also ein nichtlineares Gleichungssystem durch eine Folgen von linearen Gleichungssystemen.

4.3 Bemerkung (Konvergenzbetrachtung)
Das Newton-Verfahren muß nicht konvergieren.
Falls der Startpunkt *schlecht* gewählt ist (z.B. bei nicht-konvexen Funktionen), kann das Verfahren auch divergieren.

4.4 Algorithmus (gewöhnliches Newton-Verfahren)
Sei x_0 in der Nähe einer Nullstelle x^*.
Wir entwickeln nun die Funktion nach Taylor:

$$0 = f(x^*) = f(x_0 + (x^* - x_0)) = f(x_0) + f'(x_0) \cdot (x^* - x_0) + \mathcal{O}((x^* - x_0)^2)$$

Wir ersetzen diese Gleichung durch ein lineares Gleichungsyssystem:

$$f'(x_0) \cdot \underbrace{(x_1 - x_0)}_{=\Delta x_0} = -f(x_0)$$

Damit erhalten wir folgenden allg. Algorithmus:

1. Wähle x_0 „geschickt“.
2. für $k = 0, 1, 2, \ldots$:
   Löse $f'(x_k) \cdot \Delta x_k = -f(x_k)$ und setze $x_{k+1} = x_k + \Delta x_k$

4.5 Satz (Konvergenzverhalten)
Sei $f \in \mathcal{C}^3$ und $f(x^*) = 0$ und die Folge (x_k) sei durch das Newton-Verfahren definiert. Dann gilt für den Fehler $e_k := x_k - x^*$:

$$e_{k+1} = \frac{1}{2} f'(x_k)^{-1} \cdot f''(x_k)(e_k, e_k) + \mathcal{O}(\|e_k\|^3)$$

wobei

$$f''(x)(u, v) = \sum_{i,j=1}^{n} \frac{\partial^2 f}{\partial x_i \partial x_j}(x) u_i v_j$$

Das Newton-Verfahren konvergiert *quadratisch*, falls e_k genügend klein ist.

Beweis
Es gilt:

$$\begin{aligned} 0 = f(x^*) &= f(x_k - e_k) \\ &= \underbrace{f(x_k)} - f'(x_k) e_k + \frac{1}{2} f''(x_k)(e_k, e_k) + \mathcal{O}(\|e_k\|^3) \end{aligned}$$

Wegen des Newton-Verfahrens gilt:

$$f(x_k) = -f'(x_k)(x_{k+1} - x_k) = -f'(x_k)(x_{k+1} - x^* + x^* - x_k) = -f'(x_k)(e_{k+1} - e_k)$$

Damit ist:

$$0 = -f'(x_k) e_{k+1} + \frac{1}{2} f''(x_k)(e_k, e_k) + \mathcal{O}(\|e_k\|^3)$$

Durch Umstellen der Gleichung und Invertieren mit $f'(x_k)$ ergibt sich die Behauptung. □

4.6 Motivation
Wir möchten nun wissen, wann e_0 (und damit alle weiteren e_k) genügend klein ist, damit das Verfahren konvergiert.

4.7 Satz (Newton-Mysovskii)
Sei $D \subseteq \mathbb{R}^n$ offen, $f : D \to \mathbb{R}^n \in \mathcal{C}^1(D)$ und $f'(x)$ sei invertierbar für alle $x \in D$.
Sei $x_0 \in D$ und gelte:

1. $\|\Delta x_0\| \leq \alpha \in \mathbb{R}$ (Diese Bedingung definiert α)
2. $\|f'(x)^{-1}(f'(y) - f'(z))(y - z)\| \leq \omega \cdot \|y - z\|^2 \quad \forall x, y, z \in D$ mit y auf der Verbindungsstrecke zwischen x und z. (Diese Bedingung definiert ω).
3. $\gamma := \frac{1}{2}\alpha\omega < 1$
4. Für $\varrho := \frac{\alpha}{1-\gamma}$ sei $B_\varrho(x_0) = \{x \in \mathbb{R}^n | \|x - x_0\| < \varrho\} \subseteq D$

Dann bleibt die Folge (x_k) des Newton-Verfahrens in $B_\varrho(x_0)$ und konvergiert gegen eine Lösung x^* der Gleichung von $f(x^*) = 0$.
Außerdem gilt:

$$\|x_{k+1} - x_k\| \leq \frac{\omega}{2}\|x_k - x_{k-1}\|^2$$

und

$$\|x_k - x^*\| \leq \frac{2}{\omega}\frac{\gamma^{2^k}}{1 - \gamma^{2^k}} = \alpha\frac{\gamma^{2^k-1}}{1 - \gamma^{2^k}}$$

Beweis

1. Wir zeigen zunächst, daß für $x_{k-1}, x_k \in B_\varrho(x_0)$ gilt: $\|x_{k+1} - x_k\| \leq \frac{\omega}{2}\|x_k - x_{k-1}\|^2$:
   Es gilt:

$$\begin{aligned}\|x_{k+1} - x_k\| &= \|\Delta x_k\| = \| - f'(x_k)^{-1} f(x_k)\| \\ &= \left\| f'(x_k)^{-1} \left(f(x_k) - \underbrace{f(x_{k-1}) - f'(x_{k-1})\Delta x_{k-1}}_{=0} \right) \right\|\end{aligned}$$

   Wir verwenden nun die Beziehung

$$f(x_k) - f(x_{k-1}) = \int_0^1 f'(x_{k-1} + t\Delta x_{x-1})\Delta x_{k-1}\, dt$$

und damit ergibt sich:

$$\begin{aligned}\|x_{k+1}-x_k\| &= \left\| \int_0^1 f'(x_k)^{-1}\left(f'(x_{k-1}+t\Delta x_{k-1})-f'(x_{k-1})\right)\Delta x_{k-1}dt \right\| \\ &\le \int_0^1 \|f'(x_k)^{-1}\left(f'(x_{k-1}+t\Delta x_{k-1})-f'(x_{k-1})\right)t\Delta x_{k-1}\|\frac{dt}{t}\end{aligned}$$

Wir verwenden nun die zweite Voraussetzung, wobei $x_k = x$, $x_{k-1}+t\Delta x_{k-1} = y$ und $x_{k-1} = z$ ist (damit ist $y - z = t\Delta x_{k-1}$) und erhalten:

$$\begin{aligned}\|x_{k+1}-x_k\| &\le \int_0^1 \omega\cdot t^2\|\Delta x_{k-1}\|^2\frac{dt}{t} \\ &= \omega\cdot\|\Delta x_{k-1}\|^2\cdot\int_0^1 t\,dt \\ &= \frac{\omega}{2}\|\Delta x_{k-1}\|^2\end{aligned}$$

2. Induktion über k. Wir nehmen an, daß $x_0, x_1, \ldots, x_k \in B_\varrho(x_0)$ liegen. Dann gilt:

$$\begin{aligned}\|x_{k+1}-x_k\| = \|\Delta x_k\| &\le \frac{\omega}{2}\|\Delta x_{k-1}\|^2 \\ &\le \frac{\omega}{2}\cdot\frac{\omega^2}{4}\|\Delta x_{k-2}\|^4 \\ &\le \ldots \le \left(\frac{\omega}{2}\right)^{1+2+\ldots+2^{k-1}}\|\Delta x_0\|^{2^k} \\ &= \left(\frac{\omega}{2}\right)^{2^k-1}\|\Delta x_0\|^{2^k} \\ &= \frac{2}{\omega}\left\|\frac{\omega}{2}\Delta x_0\right\|^{2^k} \\ &\le \frac{2}{\omega}\gamma^{2^k}\end{aligned}$$

Für $0 \le \ell \le k$ ist damit

$$\begin{aligned}\|x_{k+1}-x_\ell\| &\le \|x_{k+1}-x_k\| + \|x_k - x_{k-1}\| + \ldots + \|x_{\ell+1}-x_\ell\| \\ &\le \frac{2}{\omega}\gamma^{2^\ell}\left(\ldots + \gamma^{7\cdot 2^\ell} + \gamma^{3\cdot 2^\ell} + \gamma^{2^\ell} + 1\right) \\ &\le \frac{2}{\omega}\cdot\frac{\gamma^{2^\ell}}{1-\gamma^{2^\ell}}\end{aligned}$$

Für $\ell = 0$ ist

$$\|x_{k+1}-x_0\| \le \frac{2}{\omega}\frac{\gamma}{1-\gamma} = \frac{\alpha}{1-\gamma} = \varrho$$

Damit liegt auch x_{k+1} in $B_\varrho(x_0)$.
Für $k, \gamma \to \infty$ gilt: $\|x_{k+1} - x_\ell\| \leq 0$. Damit bilden sie eine Cauchy-Folge, womit in $\mathbb{R}^n$ dies auch einer Konvergenz entspricht.
Wir müssen nur noch zeigen, daß dies einer Nullstelle entspricht. Wir wissen ja, daß gilt

$$x_{k+1} = x_k - f'(x_k)^{-1} f(x_k)$$

Damit gilt für $k \to \infty$:

$$x^* = x^* - f'(x^*)^{-1} f(x^*)$$

und damit $f(x^*) = 0$. □

4.8 Bemerkung (Praktische Durchführung des Algorithmus)
Wir haben x_0 gegeben.
Man löst nun iterativ $f'(x_k)\Delta x_k = -f(x_k)$, z.B. durch eine LR-Zerlegung.
Wir setzen nun $x_{k+1} = x_k + \Delta x_k$.
Dieses Verfahren führen wir solange durch, bis $k = k_{\max}$ oder bis $\|\Delta x_k\| \leq \texttt{tol}$, wobei `tol` eine vorgegebene Toleranz ist.

Alternativ könnte man auch $\|f(x_k)\| \leq \texttt{tol}$ als Abbruchkriterium nehmen. Dies wird in der Praxis jedoch nicht durchgeführt, denn wenn $f(x) = 0$ gilt, dann liefert $Af(x) = 0$ dieselben Nullstellen für eine reguläre Matrix A. Außerdem ändert diese Matrix auch nicht das Iterationsverfahren. Das Newton-Verfahren ist *affin-invariant*.
Allerdings ändert sich dadurch die Norm $\|Af(x_k)\| \leq \texttt{tol}$. Damit würde sich das Abbruchkriterium ändern. Dieses sollte sich aber nicht ändern, also nehmen wir nicht diese Norm

4.9 Proposition (Rechenaufwand des gewöhnlichen Newton-Verfahrens)
Wir brauchen pro Iterationsschritt eine f'-Auswertung und eine LR-Zerlegung, was je nach Größe der Matrix aufwendig ist.
Daher suchen wir nach einer Vereinfachung des Verfahrens.

4.2 Vereinfachtes Newton-Verfahren

4.10 Algorithmus (Prinzipielles Vorgehen)
Wir haben einen Startwert x_0 gegeben und berechnen $A \approx f'(x_0)$ sowie die LR-Zerlegung dieser Matrix.
Wir lösen dann für $k = 0, 1, 2, \ldots$:

$$A\Delta x_k = -f(x_k)$$

und setzen dann $x_{k+1} = x_k + \Delta x_k$.
Damit bleibt der Rechenaufwand bei einer einzigen LR-Zerlegung einer Ableitungsmatrix sowie den einzelnen Funktionsauswertungen.

4.11 Satz (Konvergenz des vereinfachten Newton-Verfahrens)
Sei f zwei mal stetig diffbar und sei (x_k) durch das vereinfachte Newton-Verfahren definiert. Dann gilt für den Fehler $e_k = x_k - x^*$:

$$e_{k+1} = (I - A^{-1}f'(x_k))e_k + \mathcal{O}(\|e_k\|^2)$$

Beweis
Es gilt wegen der Taylorentwicklung um x_k:

$$\begin{aligned} 0 = f(x^*) &= f(x_k - e_k) \\ &= \underbrace{f(x_k)}_{=-A\Delta x_k} - f'(x_k)e_k + \mathcal{O}(\|e_k\|^2) \\ &= -A(e_{k+1} - e_k) - f'(x_k)e_k + \mathcal{O}(\|e_k\|^2) \end{aligned}$$

Damit ergibt sich:

$$Ae_{k+1} = Ae_k - f'(x_k)e_k + \mathcal{O}(\|e_k\|^2)$$

womit die Behauptung bewiesen wäre. □

4.12 Satz (Newton-Mysovskii für das vereinfachte Newton-Verfahren)
Sei $D \subseteq \mathbb{R}^n$ offen und $f : D \to \mathbb{R}^n$ stetig diffbar. Dann sei $x_0 \in D$ und es gelten die folgenden Voraussetzungen:

1. $\|\Delta x_0\| \leq \alpha$
2. $\|I - A^{-1}f'(y)\| \leq \gamma < 1 \quad \forall y \in D$
3. Für $\varrho = \frac{\alpha}{1-\gamma}$ sei $\overline{B_\varrho(x_0)} \subseteq D$.

Dann bleibt die Folge (x_k) des vereinfachten Newton-Verfahrens in $\overline{B_\varrho(x_0)}$ und konvergiert gegen eine Nullstelle x^* von f.
Es gilt zudem:

$$\|x_{k+1} - x_k\| \leq \gamma\|x_k - x_{k-1}\|$$

4.13 Bemerkung
Das vereinfachte Newton-Verfahren ist im Prinzip eine Fixpunktiteration, womit sich für den Beweis der Banachsche Fixpunktsatz anbietet.

Beweis
Es gilt $x_{k+1} = \varphi(x_k)$ mit $\varphi(x) = x - A^{-1}f(x)$.

Die zweite Bedingung besagt, daß φ eine Kontraktion in $\overline{B_\varrho(x_0)}$ ist, denn es gilt nach dem Schrankensatz:

$$\|\varphi(x) - \varphi(y)\| \leq \sup_{\xi \in [x,y]} \underbrace{\|\varphi'(\xi)\|}_{=I - A^{-1} f'(\xi)} \cdot \|x - y\|$$

und damit:

$$\|\varphi(x) - \varphi(y)\| \leq \gamma \|x - y\|$$

mit $\gamma < 1$.
Außerdem gilt:

$$\varphi : \overline{B_\varrho(x_0)} \to \overline{B_\varrho(x_0)}$$

denn

$$\begin{aligned} \|\varphi(x) - x_0\| = \|\varphi(x) - \varphi(x_0) + \varphi(x_0) - x_0\| &\leq \underbrace{\|\varphi(x) - \varphi(x_0)\|}_{\leq \gamma\varrho} + \underbrace{\|\varphi(x_0) - x_0\|}_{\alpha} \\ &\leq \gamma\varrho + \alpha = \varrho \end{aligned}$$

Somit ist φ eine Kontraktion auf $\overline{B_\varrho(x_0)}$.
und nach dem Banachschen Fixpunktsatz ergibt sich, daß diese Folge gegen einen eindeutigen Grenzwert konvergiert. □

4.3 Abstiegsrichtungen und gedämpftes Newton-Verfahren

4.14 Motivation
Wie geht man vor, wenn der Startpunkt x_0 schlecht gewählt ist?

Sei $D \subseteq \mathbb{R}^n$, $f : D \to \mathbb{R}^n$ stetig diffbar.
Unser Ziel ist es, eine Nullstelle von f zu finden.
Es gilt für ein bel. $A \in \mathbb{R}^{n \times n}$ invertierbar, daß

$$f(x) = 0 \iff A^{-1} f(x) = 0 \iff \|A^{-1} f(x)\|_2^2 = \min$$

4.15 Definition (Niveaumenge)
Eine *Niveaumenge* ist

$$\{x \in D : \|A^{-1} f(x)\|_2^2 \leq c\}$$

Wir haben also eine Art Potentialtopf, mit vielen Ringen und suchen eine Abstiegsrichtung, daß man von Kreis zu Kreis geht, sodaß $c_0 > c_1 > c_2$ der Wert von $\|A^{-1} f(x)\|_2^2$ abnimmt.

4.16 Algorithmus
Wir suchen eine Abstiegsrichtung Δx und $\lambda > 0$ so, daß

$$\|A^{-1}f(x_0 + \lambda\Delta x)\|_2^2 < \|A^{-1}f(x_0)\|_2^2$$

Wir betrachten also

$$\begin{aligned}\|A^{-1}f(x_0 + \lambda\Delta x)\|_2^2 &= f(x_0 + \lambda\Delta x)^T A^{(-1)^T} A^{-1} f(x_0 + \lambda\Delta x) \\ &= \underbrace{f(x_0)^T A^{(-1)^T} A^{-1} f(x_0)}_{\|A^{-1}f(x_0)\|_2^2} \\ &\quad + 2\lambda \cdot \underbrace{f(x_0)^T (A^{-1})^T A^{-1} f'(x_0)\Delta x}_{<0} + \mathcal{O}(\lambda^2)\end{aligned}$$

4.17 Bemerkung (Methode des steilsten Abstiegs)
Für $A = I$ ist dies die *Methode des steilsten Abstiegs* $\Delta x = -f'(x_0)^T f(x_0)$
Dies steht orthogonal auf der Niveaulinie, denn für $\varphi(x) = \|f(x)\|_2^2 = f(x)^T f(x)$ ist

$$\nabla\varphi(x) = \varphi'(x) = 2f'(x)^T f(X)$$

und $\Delta x = -\frac{1}{2}\nabla\varphi(x_0)$.
Für die Niveaulinien gilt aber, daß $\varphi(x(t)) = \text{const}$ und damit:

$$\langle \nabla\varphi(x(t)), x'(t)\rangle = \varphi'(x(t)) \cdot x'(t) = 0$$

und damit

$$\langle \Delta x, x'(0)\rangle = 0$$

d.h. Δx steht senkrecht auf den Niveaulinien.

4.18 Beispiel
Sei $f(x_1, x_2) = \begin{pmatrix} x_1 \\ 100x_2 \end{pmatrix}$. Dann ist $\|f(x)\|^2 = x_1^2 + 10^4 x_2^2$.
Dabei sind die Niveaulinien schmale Ellipsen. Hier ist die Methodes des steilsten Abstiegs nicht optimal.

Würden wir allerdings die Matrix $A = \begin{pmatrix} 1 & 0 \\ 0 & 100 \end{pmatrix}$ wählen, dann wäre $A^{-1}f(x) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$.
Dann erhält man als Niveaulinien nur Kreise, d.h. man erhält eine schnelle Konvergenz.

4.19 Bemerkung (Wahl der Richtung)
Wir fragen uns, ob es eine Richtung Δx gibt, die für jede Wahl einer invertieren Matrix A eine Abstiegsrichtung ist, d.h. wir wollen, daß $f(x_0)^T A^{(-1)^T} A^{-1} f'(x_0)\Delta x < 0 \quad \forall A$.
Dies ist erfüllt für $\Delta x = -f'(x_0)^{-1} f(x_0)$.
Wir erhalten nämlich dann:

$$-\|A^{-1}f(x_0)\|_2^2 < 0 \quad f(x_0) \neq 0$$

4.20 Bemerkung
Die Newton-Richtung ist die einzige Richtung, die diese Bedingung erfüllt (ohne Beweis). Außerdem ist dies die Richtung des steilsten Abstiegs für $A = f'(x_0)$.

4.21 Motivation (zum gedämpften Newton-Verfahren)
Ziel ist es, das Newton-Verfahren für „schlechte“ Startwerte konvergieren zu lassen, aber die Newton-Richtung ist die Abstiegsrichtung für jede Wahl von A und die steilste Abstiegsrichtung für $A = f'(x_0)$.

4.22 Algorithmus (gedämpftes Newton-Verfahren)
Wir haben x_0 gegeben und für $k = 0, 1, 2, \ldots$ lösen wir: $f'(x_0)\Delta x_k = -f(x_k)$ wie beim gewöhnlichen Newton-Verfahren.
Wir setzen dann

$$x_{k+1} = x_k + \lambda_k \Delta x_k \quad 0 < \lambda_k \leq 1$$

Wir wählen dabei den Dämpfungsfaktor λ_k so, daß

$$\|A^{-1} f(x_{k+1})\| < \|A^{-1} f(x_k)\|$$

Für gewöhnlich nimmt man $\lambda_k \in \{1, \frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^k}, \ldots\}$.
Außerdem wählt man dann $A = f'(x_0)$ oder $A = f'(x_k)$.

4.4 Homotopiemethoden

4.23 Motivation (Einbettung der Funktion in eine Homotopie)
Wir möchten noch immer $f(x) = 0$ lösen.
Wir nehmen an, daß $f(x)$ in ein Problem $F(x, \tau) = 0$ eingebettet ist. Dabei ist τ der Homotopie-Parameter.
Dabei ist $F : \mathbb{R}^n \times [0, 1] \to \mathbb{R}^n$.

Für $\tau = 1$ seien die Lösungen $F(x, 1) = 0$ die Lösungen von $f(x) = 0$, also typischerweise $F(x, 1) = f(x)$.
Für $\tau = 0$ seien die Lösungen von $F(x, 0) = 0$ bekannt (oder leicht zu berechnen).

4.24 Beispiel
Sei $F(x, \tau) = \tau f(x) + (1 - \tau) f_0(x)$, wobei die Nullstellen von $f_0(x)$ bekannt sind.
Dies funktioniert sehr gut, wenn f_0 „nahe“ an f liegt.

Oft sind solche Einbettungen kanonisch aus physikalischer Einsicht gegeben.

4.25 Algorithmus (Einfache Homotopiemethode)
Wir tasten uns dabei von $\tau = 0$ zu $\tau = 1$ hoch mit $x(\tau_{v-1})$ als Startpunkt für benachbarte τ_v.

4.26 Algorithmus (Tangenten-Homotopiemethode)
Wir differenzieren $F(x(\tau), \tau)$ nach τ:

$$\frac{\partial F}{\partial x}(x(\tau), \tau)x'(\tau) + \frac{\partial F}{\partial \tau}(x(\tau), \tau) = 0$$

und lösen nach $x'(\tau)$ auf:

$$x'(\tau) = -\left(\frac{\partial F}{\partial x}\right)^{-1} \frac{\partial F}{\partial \tau}$$

und erhalten eine Tangentenrichtung.
Damit erhalten wir den neuen Startwert:

$$\hat{x}(\tau_m) = x(\tau_{m-1}) + (\tau_m - \tau_{m-1}) \cdot x'(\tau_{m-1})$$

Man wählt dann τ_m so, daß das Newton-Verfahren „schnell“ konvergiert.

4.5 Nichtlineare Ausgleichsrechnung

4.27 Motivation
Wir haben eine überbestimmte nichtlineare Gleichung $f(x) = 0$ mit $f : \mathbb{R}^n \to \mathbb{R}^m$ für $m > n$. Dies hat im allgemeinen keine Lösung. Daher suchen wir eine Lösung der Minimierungsaufgabe, d.h.
wir suchen $x^* \in \mathbb{R}^n$ mit $\|f(x^*)\|_2$ minimal.

4.28 Idee
Wir linearisieren f wie beim Newton-Verfahren und lösen zu einem gegebenen x_0 dann

$$\|f(x_0) + f'(x_0)(x^* - x_0) + \mathcal{O}(\|x^* - x_0\|^2)\|_2 = \min$$

Dabei vernachlässigen wir die quadratischen Terme und erhalten dann ein lineares Ausgleichsproblem und erhalten damit eine neuen Startwert:

$$\|f(x_0) + f'(x_0)(x_1 - x_0)\|_2 = \min$$

Dies löst man iterativ mit dem Gauß-Newton-Verfahren.

4.29 Algorithmus (Gauß-Newton-Verfahren)
Für $k = 0, 1, \ldots,$ berechne $f(x_k)$ und $f'(x_k)$, bestimme dann Δx_k als Lösung eines linearen Ausgleichsproblems mittels

$$\|f(x_k) + f'(x_k)\Delta x_k\|_2 = \min$$

Setze danach $x_{k+1} = x_k + \Delta x_k$

4.30 Bemerkung (Konvergenzbetrachtung des Gauß-Newton-Verfahrens)
Wir betrachten $\varphi(x) = \|f(x)\|_2^2 = f(x)^T f(x)$. Eine notwendigige Bedingung für das Vorliegen eines Minimums ist $\nabla\varphi(x) = 0 = 2f'(x)^T f(x)$. Theoretisch könnte man auf diesen Ausdruck das Newton-Verfahren anwenden, dann bräuchte man aber die zweite Ableitung von f. Dies möchten wir vermeiden.
Wir betrachten nun $g(x) = f'(x)^T f(x)$. Wir wissen, daß $g(x^*) = 0$ gilt, falls x^* das Minimum ist.
Wir erhalten:

$$g(x) = g(x_k) + g'(x_k) \cdot (x - x_k) + \mathcal{O}(\|x - x_k\|^2)$$

Wie sieht $g_i(x)$ aus?

$$g_i(x) = \sum_{j=1}^{m} \frac{\partial f_j}{\partial x_i}(x) f_j(x)$$

Wir betrachten nun

$$\frac{\partial g_i}{\partial x_\ell}(x) = \sum_{j=1}^{m} \frac{\partial^2 f_j}{\partial x_\ell \partial x_i}(x) f_j(x) + \sum_{j=1}^{m} \frac{\partial f_j}{\partial x_i}(x) \frac{\partial f_j}{\partial x_\ell}(x)$$

In Matrixform ergibt sich:

$$g'(x)v = B(x)(f(x), v) + f'(x)^T f'(x)v$$

wobei $B : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^n$ billinear definiert wird durch

$$(B(x)(u, v))_i = \sum_{\ell=1}^{n} \sum_{j=1}^{m} \frac{\partial^2 f_j}{\partial x_\ell \partial x_i}(x) u_j v_\ell$$

Damit ist

$$\begin{aligned} 0 = g(x) &= g(x_k) + g'(x_k)(x - x_k) + \mathcal{O}(\|x - x_k\|^2) \\ &= \underbrace{f'(x_k)^T f(x_k)}_{} + f'(x_k)^T f'(x_k)(x - x_k) + B(x_k)(f(x_k), x - x_k) + \mathcal{O}(\|x - x_k\|^2) \end{aligned}$$

Bei dem Gauß-Newton-Verfahren ermitteln wir

$$\|f(x_k) + f'(x_k)(x_{k+1} - x_k)\|_2 = \min$$

Mit der Normalengleichung ergibt sich:

$$f'(x_k)^T f'(x_k)(x_{k+1} - x_k) = -f'(x_k)^T f(x_k)$$

wobei hier die rechte Seite wieder in der obigen Gleichung zu finden ist. Setzt man dies ein, folgt

$$\begin{aligned} 0 = & -f'(x_k)^T f'(x_k)(x_{k+1}x_k) + f'(x_k)^T f'(x_k)(x - x_k) \\ & + B(x_k)(f(x_k), x - x_k) + \mathcal{O}(\|x - x_k\|^2) \\ & = f'(x_k)^T f'(x_k)(x - x_{k+1}) + B(x_k)(f(x_k), x - x_k) + \mathcal{O}(\|x - x_k\|^2) \end{aligned}$$

4.31 Satz (Fehler beim Gauß-Newton-Verfahren)
Sei $f : \mathbb{R}^n \to \mathbb{R}^m$ mit $m > n$ drei mal stetig diffbar und $\operatorname{rg} f'(x) = n$ und sei x^* die Lösung von $\|f(x)\|_2 = \min$.
Für den Fehler $e_k = x_k - x^*$ des Gauß-Newton-Verfahrens gilt:

$$e_{k+1} = -\left(f'(x_k)^T f'(x_k)\right)^{-1} B(x_k)(f(x_k), e_k) + \mathcal{O}(\|e_k\|^2)$$

4.32 Bemerkung (Diskussion des Konvergenzverhaltens)

1. Falls $f(x^*) = 0$, dann ist $f(x_k) = f(x^*) + f'(x_k)e_k + \mathcal{O}(\|x_k\|^2)$. Dann erhält man eine lokale quadratische Konvergenz.
2. Falls $\|f(x^*)\|_2$ genügend klein ist, erhält man lineare Konvergenz.
3. Falls $\|f(x^*)\|$ zu groß ist, kann das Verfahren auch divergieren.

5 Numerische Verfahren für AWP gewöhnlicher Differentialgleichungen

5.1 Motivation
Wir haben eine Anfangswertaufgabe $y'(t) = f(t, y(t))$ gegeben mit $y(t_0) = y_0$ und suchen dafür eine Lösung $y : [t_0, T] \to \mathbb{R}^\alpha$, wobei $f : U \to \mathbb{R}^\alpha$ mit $U \subseteq \mathbb{R} \times \mathbb{R}^\alpha$ mit $(t_0, y_0) \in U$.

5.1 Einige Beispiele von Differentialgleichungen

5.2 Beispiel (aus der Physik)
Als klassisches Beispiel hat man ein mathematisches Pendel:
Dabei ist $s = \ell \cdot \alpha$, wobei α der Winkel, s die Auslenkung und ℓ die Länge des Fadens ist.
Dabei gilt dann $ms''(t) = -mg \sin \alpha(t)$ und man erhält damit die Differentialgleichung

$$\alpha''(t) = -\frac{g}{\ell} \sin \alpha(t)$$

was eine homogene Differentigleichung 2. Ordnung ist, die sich als System von Differentialgleichungen 1. Ordnung schreiben läßt:
Mit $\omega = \alpha'$ erhält man

$$\alpha' = \omega, \qquad \omega' = -\frac{g}{\ell} \sin \alpha$$

mit $y = \begin{pmatrix} \alpha \\ \omega \end{pmatrix}$

5.3 Beispiel (aus der Chemie)
Bei einer chem. Reaktion gibt es Substanzen X, Y, Z die wie folgt reagieren:

$$X \to Y, \qquad Y + Z \to X + Z, \qquad Y + Y \to Z + Y$$

Sei $x(t)$ die Konzentration von X zur Zeit t. Zu welchen Gesetzmäßigkeiten genügen die Substanzen?

Dies führt zum Massenwirkungsgesetz, das auf Differentialgleichungen führt.
Dabei sind k_1, k_2, k_3 Reaktionsparameter und wir erhalten die folgende Differentiagleichung

$$\begin{aligned} x' &= -k_1 x + k_2 yz \\ y' &= k_1 x - k_2 yz - k_3 y^2 \\ z' &= k_3 y^2 \end{aligned}$$

Dazu gibt es natürlich gewisse Anfangsbedingungen, mit Hilfe derer man den Reaktionsverlauf zu jedem bel. Zeitpunkt berechnen kann

5.4 Beispiel (aus der Biologie)
Hier gibt es Modelle zur Populatiostheorie, was auf Voltaire (um 1920) zurück geht.
Sei $y(t)$ die Anzahl der Speisefisch zur Zeit t und $z(t)$ die Anzahl der Raubfisch zur Zeit t.
Dabei ergibt sich das folgende System von Differentialgleichungen:

$$\begin{aligned} y' &= ay - byz \\ z' &= -cz + dyz \end{aligned}$$

Es gibt aber auch komplizierter Modelle, etwa Epimediemodelle, usw.

5.5 Bemerkung (Raumdiskretisierung bei PDE)
Wir diskutieren dies am Beispiel der Wärmeleitungsgleichung:

$$\frac{\partial u}{\partial t}(x,t) = \frac{\partial^2 u}{\partial^2 x}(x,t) + g(x,t) \quad 0 \leq x \leq 1$$

mit der Randbedingung $u(0,t) = u(1,t) = 0$ für $t > 0$ und der Anfangsbedingung: $u(x,0) = u_0(x)$

Die numerische Lösung dieses Ansatzes erfolgt durch eine Diskretisierung bezüglich x, d.h. wir unterteilen das Intervall in N Teilintervalle, d.h. $1 = N\Delta x$ und erhalten dadurch auch $(N+1)^2$ Gitterpunkte.
Dazu ersetzen wir $\frac{\partial^2}{\partial x^2}$ durch einen Differenzenquotienten, der nur auf den Gitterpunkten lebt.
Wir approximieren dann:

$$y_i(t) \approx u(\underbrace{i\Delta x}_{=x_i}, t) \quad (i = 0, \ldots, N+1)$$

was definiert ist durch:

$$y_i'(t) = \frac{1}{\Delta x^2}\left(y_{i+1}(t) - 2y_i(t) + y_{i-1}(t)\right) + g(x_i, t)$$

Wir setzen außerdem entsprechend der Randbedingungen dann

$$y_0(t) = y_{N+1}(t) = 0$$

Wir erhalten auf diese Weise ein großes System von gewöhnlichen Differentialgleichungen.

5.2 Bemerkungen, Erinnerungen an gewöhnliche Differentialgleichungen

5.6 Bemerkung

1. Eine Differentialgleichung zweiter Ordnung $y'' = f(t, y, y')$ läßt sich als System von Differentialgleichungen erster Ordnung schreiben:

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

2. Bei nichtautonomen Differentialgleichungen ist die Differentialgleichung von t abhängig, d.h. $y' = f(t, y)$ mit $y(t_0) = y_0$
   Diese Differentialgleichungen sind äquivalent zu autonomen Systemen der Dimension $d + 1$:

   $$t' = 1$$
   $$y' = f(t, y)$$

   Damit ist mit $\hat{y} = \begin{pmatrix} t \\ y \end{pmatrix}$:

   $$\begin{pmatrix} t \\ y \end{pmatrix}(t_0) = \begin{pmatrix} t_0 \\ y_0 \end{pmatrix}$$

   und es ergibt sich als neue Differntialgleichung $\hat{y}' = F(\hat{y})$.

5.7 Erinnerung (an Existenz und Eindeutigkeit)
Wir haben $U \subseteq \mathbb{R} \times \mathbb{R}^d$ offen, zusammenhängend und $(t_0, y_0) \in U$.
Außerdem sei $f : U \to \mathbb{R}^d$ stetig und erfülle lokal eine Lipschitzbedigung bezüglich der zweiten Komponente, d.h.

$$\forall K \subseteq U \; K \text{ kompakt } \forall (t, y), (t, z) \in K \; \exists L = L(K) \quad \|f(t, y) - f(t, z)\| \leq L\|y - z\|$$

Dies ist auf jeden Fall erfüllt, falls f stetig diffbar ist. Dann ist

$$L = \max_{(t,y)\in K} \left\| \frac{\partial f}{\partial y}(t, y) \right\|$$

Dann gilt:

1. Es gibt ein offenes Intervall I mit $t_0 \in I$.
2. Es gibt genau ein $y : I \to \mathbb{R}^d$ mit

   $$y'(t) = f(t, y(t)) \quad \forall t \in I, \qquad y(t_0) = y_0$$

   Dies ist der Existenzsatz von Picard-Lindelöf.

3. Die Lösung kann bis an den Rand von U fortsetzt werden, genauer:
   $\forall K \subseteq U, K$ kompakt mit $(t_0, y_0) \in K \quad \exists t_1 > t_0$, so daß die Lösung des Anfangswertproblems auf $[t_0, t_1]$ existiert und $t_1, y(t_1) \notin K$ ist.

Beweis (Beweisidee für den Satz von Picard-Lindelöf)
Wir wissen, daß

$$y(t) = y_0 + \int_0^t f(s, y(s))ds$$

und damit iterieren wir dies k-Mal:

$$y^{k+1}(t) = y_0 + \int_0^t f(s, y^{(k)}(s))\, ds$$

Dann führen wir eine Fixpunktiteration in $\mathcal{C}[I]$ durch und verwenden den Banachschen Fixpunktsatz, da wir den Operator durch die Lipschitzkonstante beschränken können. □

5.8 Beispiel
Betrachte $y' = y^2$ mit $y(0) = 1$ und $U = \mathbb{R} \times \mathbb{R}$ ist lokal Liptschitz-stetig. Dann ist

$$y(t) = \frac{1}{1-t} \quad 0 \leq t < 1$$

eine Lösung.

5.9 Satz (Einfluß von Störungen der Anfangswerte auf die Lösung)
Sei $\langle ., . \rangle$ ein Skalarprodukt auf $\mathbb{R}^d$ und $\| \cdot \|$ sei die zugehörige Norm mit $\|y\|^2 = \langle y, y \rangle$.
Es gelte die „einseitige Lipschitz-Bedignung“:

$$\langle f(t,u) - f(t,v), u - v \rangle \leq \ell \|u - v\|^2 \quad \forall (t,u), (t,v) \in U$$

Seien $y, z : I \to \mathbb{R}^d$ Lösungen der Differentialgleichungen $y' = f(t,y)$ und $y(t_0) = y_0$, $z' = f(t,z)$ und $z(t_0) = z_0$.
Dann gilt:

$$\|y(t) - z(t)\| \leq e^{\ell(t-t_0)} \cdot \|y_0 - z_0\| \quad \forall t \in I$$

Beweis
Betrachte

$$\begin{aligned}\frac{d}{dt}\|y(t)-z(t)\|^2 &= \frac{d}{dt}\langle y(t)-z(t), y(t)-z(t)\rangle \\ &= 2\langle y'(t)-z'(t), y(t)-z(t)\rangle \\ &= 2\langle f(t,y(t))-f(t,z(t)), y(t)-z(t)\rangle \\ &\leq 2\ell\|y(t)-z(t)\|^2\end{aligned}$$

Falls $y(t_0) \neq z(t_0)$, so gilt wegen der Eindeutigkeit der Lösungen, daß $y(t) \neq z(t) \quad \forall t \in I$.
Mit $\varphi(t) = \|y(t)-z(t)\|^2$ erhalten wird:

$$\frac{d}{dt}\ln\varphi(t) = \frac{\varphi'(t)}{\varphi(t)} \leq 2\ell$$

Integriert man dies auf beiden Seiten von t_0 bis t erhalten wir

$$\underbrace{\ln\varphi(t) - \ln\varphi(t_0)}_{=\ln\frac{\varphi(t)}{\varphi(t_0)}} \leq 2\ell(t-t_0)$$

Wegen der Monotonie der Exponentialfunktion ergibt sich:

$$\frac{\varphi(t)}{\varphi(t_0)} \leq e^{2\ell(t-t_0)}$$

und damit ergibt sich

$$\|y(t)-z(t)\|^2 = \varphi(t) \leq e^{2\ell(t-t_0)}\varphi(t_0) = e^{2\ell(t-t_0)}\|y_0-z_0\|^2$$

Durch Wurzelziehen ergibt sich dann schließlich

$$\|y(t)-z(t)\| \leq e^{\ell(t-t_0)}\|y_0-z_0\|$$

□

5.10 Bemerkung

1. Der Fehler in y_0 kann sich dem Faktor $e^{\ell(t-t_0)}$ auswirken.
2. Falls f Lipschitz-stetig ist auf U, d.h. $\|f(t,u)-f(t,v)\| \leq L\|u-v\| \quad \forall (t,u),(t,v) \in U$, dann gilt insbesondere die einseitige Lipschitz-Bedingung mit $\ell \leq L$ (dies ergibt sich unmittelbar aus der Cauchy-Schwarze-Ungleichung).

   Für das bestmögliche ℓ kann unter Umständen $\ell \ll L$ gelten. Differentialgleichungen, bei denen dies gilt, werden als *„steife“ Differentialgleichungen* bezeichnet.

5.11 Beispiel
Betrachte $y' = -1000y$ für $t \in [0,1]$. Dabei ist

$$\| -1000u - (-1000v)\| = \underbrace{1000}_{=L} \|u - v\|$$
$$\langle -1000u - (-1000v), u - v\rangle = \underbrace{-1000}_{=\ell} \|u - v\|^2$$

Dies macht bei der Fehlerfortplanzung einen gewaltigen Unterschied aus.

Allgemein ergibt sich bei $y' = \lambda y$ dann $y(t) = e^{\lambda(t-t_0)}y_0$ die Lösung und es ergibt sich $\ell = \lambda$ und $L = |\lambda|$.

1. Für $\lambda < 0$ werden Fehler im Anfangswert ausgedämpft. Dieses System nennt man „asympotisch stabil“.
2. Für $\lambda = 0$ erhält man eine konstante Funktion. Dies nennt man „stabil“.
3. Für $\lambda > 0$ driftet die Funktion auseinander. Dann ist die Funktion „instabil“.

5.3 Euler-Verfahren

5.12 Historische Notiz (Euler-Verfahren)
Das Euler-Verfahren ist das einfachste und älteste Verfahren zur näherungsweisen Lösung von $y' = f(t,y)$ mit $y(t_0) = y_0$ und geht auf Leonhard Euler, 1768 zurück.

5.13 Idee
Wir ersetzen lokal die (unbekannte) Lösung durch die bekannte Tangente:
Wir nehmen h als Schrittweite:

$$t_1 = t_0 + h$$
$$y_1 = y_0 + hy'(t_0) = y_0 + hf(t_0, y_0)$$

Wir nehmen dieses y_1 als Approximation der Lösung an der Stelle t_1.
Wir nehmen dieses y_1 als Startwert für den nächsten Schritt und setzen

$$y_2 = y_1 + hf(t_1, y_1)$$

für $t_2 = t_1 + h = t_0 + 2h$, usw.

Wir erhalten dann $y_n \approx y(t_n)$ mit $t_n = t_0 + nh \quad \forall n = 0, 1, \ldots$.

5.14 Algorithmus (Euler-Verfahren)
Es sei $y' = f(t,y)$ gegeben mit dem Startwert $y(t_0) = y_0$.
Setze für $n = 0, 1, 2, \ldots$:

$$y_{n+1} = y_n + hf(t_n, y_n)$$

mit $t_{n+1} = t_n + h$.

5.15 Satz (Fehlerabschätzung beim Euler-Verfahren)
Sei $I = [t_0, T]$ ein Intervall und $f : I \times \mathbb{R}^d \to \mathbb{R}^d$ stetig diffbar und (global) Lipschitz-stetig, d.h.

$$\|f(t,u) - f(t,v)\| \leq L\|u - v\| \quad \forall t \in I, \forall u, v \in \mathbb{R}^d$$

Sei $y : I \to \mathbb{R}^d$ die Lösung des AWP: $y' = f(t,y)$ und $y(t_0) = y_0$
Wegen f stetig diffbar, ist y zwei-mal stetig diffbar:

$$y'' = \frac{\partial f}{\partial t}(t,y) + \frac{\partial f}{\partial y}(t,y) \cdot y'$$

Seien y_n für $n = 0, 1, \ldots$ durch das Euler-Verfahren definiert.

Dann gilt für den Fehler:

$$\|y_n - y(t_n)\| \leq M \cdot h \quad \forall t_0 + n \cdot h = t_n \leq T$$

mit

$$M = \frac{e^{L(T-t_0)} - 1}{L} \frac{1}{2} \max_{t \in I} \|y''(t)\|$$

Insbesondere gilt:

$$\lim_{h \to 0} \max_{n : t_n \in I} \|y_n - y(t_n)\| = 0$$

d.h. die Näherungslösung konvergiert gleichmäßig gegen die exakte Lösung des AWP, wenn die Schrittweite gegen 0 geht.

Beweis
Wir benötigen drei Schritte:

1. Wir untersuchen den *lokalen Fehler*, dies ist der Fehler nach einem Schritt, wenn wir bei der exakten Lösung starten:

   $$\underbrace{y(t_{n+1})}_{\text{exakt}} - \underbrace{(y(t_n) + hf(t_n, y(t_n)))}_{\text{Euler-Schritt}} = y(t_n + h) - y(t_n) - hy'(t_n)$$

   Dies entspricht gerade den ersten Gliedern der Taylorentwicklung und damit gilt:

   $$\underbrace{y(t_{n+1})}_{\text{exakt}} - \underbrace{(y(t_n) + hf(t_n, y(t_n)))}_{\text{Euler-Schritt}} = h^2 \int_0^1 (1 - \Theta) y''(t_n + \Theta h) d\Theta$$

und damit folgt, wenn wir die Normen betrachten:

$$\|y(t_{n+1}) - y(t_n) - hf(t_n, y(t_n))\| \leq Ch^2$$

mit

$$C = \frac{1}{2} \max_{t \in I} \|y''(t)\|$$

2. Wir betrachten die *Fehlerfortplanzung*:
Wir betrachten hierzu den Euler-Schritt zum Startwert v_n

$$v_{n+1} = v_n + hf(t_n, v_n)$$

und einen Eulerschritt zum Startwert w_n:

$$w_{n+1} = w_n + hf(t_n, w_n)$$

Wir betrachten, was sich als Fehler für die $n+1$-Schritte ergibt:

$$\begin{aligned} \|v_{n+1} - w_{n+1}\| &\leq \|v_n - w_n\| + h \underbrace{\|f(t_n, v_n) - f(t_n, w_n)\|}_{\leq L\|v_n - w_n\|} \\ &\leq (1 + hL)\|v_n - w_n\| \end{aligned}$$

3. *Fehlerakkummulierung*: „Fächer der Lady Windermere“
Bezeichne mit y_n^k die Näherung (an $y(t_n)$) zum AWP $y(t_k)$ nach $n-k$ Schritten. Insbesondere ist $y_n = y_n^0$ und $y(t_n) = y_n^n$

Die lokalen Fehler ergeben sich aus

$$\|y_{k+1}^k - y(t_{k+1})\| \leq Ch^2$$

Außerdem ist durch die Fehlerfortplanzung gegeben:

$$\begin{aligned} \|y_n^k - y_n^{k+1}\| &\leq (1 + hL)\|y_{n-1}^k - y_{n-1}^{k+1}\| \\ &\leq \dots \\ &\leq (1 - hL)^{n-k-1}\|y_{k+1}^k - \underbrace{y_{k+1}^{k+1}}_{= y(t_{k+1})}\| \leq (1 + hL)^{n-k-1}Ch^2 \end{aligned}$$

Durch Anwenden der Dreiecksungleich ergibt sich dann:

$$\begin{aligned} \|y_n - y(t_n)\| &= \|y_n^0 - y_n^n\| \\ &\leq \|y_n^0 - y_n^1\| + \|y_n^1 - y_n^2\| + \dots + \|y_n^{n-1} - y_n^n\| \\ &\leq Ch^2(1 + hL)^{n-1} + Ch^2(1 + hL)^{n-2} + \dots + Ch^2 \\ &= Ch^2 \frac{(1 + hL)^n - 1}{1 + hL - 1} = Ch \frac{(1 + hL)^n - 1}{L} \\ &\leq Ch \frac{e^{nhL} - 1}{L} \leq Ch \frac{e^{L(T - t_0)} - 1}{L} \end{aligned}$$

Damit ergibt sich die Konstante. □

5.4 „Steife“ Differentialgleichungen, implizites Euler-Verfahren

5.16 Motivation
Wir haben gesehen, daß beim (expliziten) Euler-Verfahren die Fehler mit dem Faktor $(1+hL)$ je Schritt verstärkt werden:

$$y_{n+1} = y_n + hf(t_n, y_n)$$
$$z_{n+1} = z_n + hf(t_n, z_n)$$

5.17 Beispiel (zum expliziten Euler-Verfahren)
Sei $y' = -1000y$ mit $t \in [0, 1]$ für $y_0 = 1$. die Exakte Lösung ist $y(t) = e^{-1000t}$.
Wir betrachten nun die Lösung dieses Verfahrens mit dem Euler-Verfahren:

$$y_{n+1} = y_n + h(-1000y_n) = (1 - 1000h)y_n$$

Man würde hier für eine fast annäherende Lösung sehr kleine Schrittweiten benötigen. Die numerische Lösung bleibt für zu große Schrittweiten nicht beschränkt – im Gegensatz zur exakten Lösung.

Dies können wir auf folgende Art und Weisen umgehen:

1. Wir benötigen sehr kleine Schrittweiten, also mind. $h < \frac{2}{1000}$, um eine sehr glatte Lösung zu approximieren, was zu einem hohen Rechenaufwand führt.
2. Wir suchen ein Verfahren, das ähnliche Stabilitätseigenschaften wie die Differentialgleichung hat.

Die zweite Variante führt dann zu einem neuen Algorithmus.

5.18 Algorithmus (Implizites Euler-Verfahren)
Wir führend die folgende Iteration ein:

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$$

wobei wir daraus y_{n+1} bestimmen, was im Allgemeinen nicht linear sein muß.

5.19 Beispiel
Wir betrachten das Beispiel $y' = \lambda y$ für $\lambda \leq 0$. Dann ist $y(t) = e^{\lambda t}y_0$, was für $t \geq 0$ beschränkt ist.

1. Das explizite Euler-Verfahren bringt:

   $$y_{n+1} = y_n + h\lambda y_n = (1 + h\lambda)y_n$$

   Dies ist nur beschränkt, falls $h \leq \frac{2}{|\lambda|}$

2. Durch das implizierte Euler-Verfahren erhalten wir allerdings:

$$\begin{aligned} y_{n+1} &= y_n + h\lambda y_{n+1} \\ &= \frac{1}{1-h\lambda} y_n \end{aligned}$$

Dies ist $\forall h > 0$ beschränkt.
Wir haben keine Schrittweiteneinschränkung durch λ.

5.20 Satz (Konvergenz beim impliziten Euler-Verfahren)
Sei $f : I \times \mathbb{R}^d \to \mathbb{R}^d$ mit $I = [t_0, T]$ stetig diffbar und erfülle die einseitige Lipschitz-Bedingung:

$$\langle f(t,u) - f(t,v), u - v \rangle \leq \ell \|u - v\|^2 \quad \forall t \in I \; \forall u, v \in \mathbb{R}^d$$

für ein Skalarprodukt $\langle ., . \rangle$ und $\| \cdot \|$ die dadurch induzierte Norm.

Dann gilt für den Fehler des impliziten Euler-Verfahrens wenn die Schrittweite $h\ell \leq \alpha < 1$:

$$\|y_n - y(t_n)\| \leq m \cdot h \quad t_n = t_0 + nh \in I$$

mit

$$m = \frac{e^{\ell(t-t_0)/(1-\alpha)} - 1}{\ell} \cdot \frac{1}{2} \max_{t \in I} \|y''(t)\|$$

5.21 Bemerkung
Unter der Voraussetzungen des Satzes hat das implizite Euler-Verfahren in jedem Schritt eine eindeutige Lösung y_{n+1}, falls $h\ell < 1$ ist.

5.22 Erinnerung
Für Lösungen $y(t), z(t)$ der Differentialgleichung gilt:

$$\|y(t) - z(t)\| \leq e^{\ell(t-t_0)} \|y(t_0) - z(t_0)\|$$

Es gilt aber immer, daß $\ell \leq L$, manchmal sogar, daß $\ell \ll L$ (dies ist bei steifen Differentialgleichungen der Fall)

Beweis (des Satzes)
Wir beweisen hier nur die Abschätzung. Die Existenz folgt aus der Bedingung, daß es sich um eine strikte Kontraktion handelt.
Der Beweis folgt fast analog wie beim expliziten Euler-Verfahren und läuft ebenfalls in drei Schritten ab. Wir verwenden dabei die gleiche Notation wie beim Beweis des Expliziten Verfahrens:

1. *Lokaler Fehler*:
   Wir betrachten einen impliziten Euler-Schritt zum Startwert $y(t_n)$:

$$\begin{aligned} y_{n+1}^n &= y(t_n) + hf(t_{n+1}, y_{n+1}^n) \\ y(t_{n+1}) &= y(t_n) + h\underbrace{f(t_{n+1}, y(t_{n+1}))}_{y'(t_{n+1}))} + d_{n+1} \end{aligned}$$

   Wir sehen, daß es sich dabei um eine Taylorentwicklung handelt, wobei d_{n+1} der Taylor-Restterm bei der Taylor-Entwicklung um t_{n+1} ist. Daher gilt analog zum expliziten Euler-Verfahren

$$\|d_{n+1}\| \leq Ch^2, \qquad C = \frac{1}{2}\max_{t\in I}\|y''(t)\|$$

   Nun subtrahieren wir die obigen Gleichungen und und machen eine Skalarmultiplikation mit $y_{n+1}^n - y(t_{n+1})$ und erhalten:

$$\begin{aligned} \|y_{n+1}^n - y(t_{n+1})\|^2 = h\underbrace{\langle f(t_{n+1}, y_{n+1}^n) - f(t_{n+1}, y(t_{n+1})), y_{n+1}^n - y(t_{n+1})\rangle}_{\leq \ell\|y_{n+1}^n - y(t_{n+1})\|^2} \\ + \underbrace{\langle d_{n+1}, y_{n+1}^n - y(t_{n+1})\rangle}_{\leq \|d_{n+1}\|^2 \cdot \|y_{n+1}^n - y(t_{n+1})\|^2} \end{aligned}$$

   Wobei die linke Abschätzung durch die Vor. und die rechte Abschätzung durch die Cauchy-Schwarz-Ungleichung folgt.
   Damit ergibt sich dann:

$$\begin{aligned} \underbrace{(1-h\ell)}_{>0}\|y_{n+1}^n - y(t_{n+1})\| &\leq Ch^2\|y_{n+1}^n - y(t_{n+1})\| \\ \|y_{n+1}^n - y(t_{n+1})\| &\leq \frac{Ch^2}{1-h\ell} \end{aligned}$$

2. *Fehlerfortplanzung*:
   Wir betrachten hier je einen impliziten Eulerschritt zu zwei verschiedenen Startwerten:

$$\begin{aligned} v_{n+1} &= v_n + hf(t_{n+1}, v_{n+1}) \\ w_{n+1} &= w_n + hf(t_{n+1}, w_{n+1}) \\ &= v_n + hf(t_{n+1}, w_{n+1}) + (w_n - v_n) \end{aligned}$$

   Wie im ersten Teil können wir nun die beiden Gleichungen subtrahieren und skalar mit $v_{n+1} - w_{n+1}$ multiplizieren und erhalten dafür:

$$\|v_{n+1} - w_{n+1}\| \leq \frac{\|v_n - w_n\|}{1-h\ell}$$

   Wir beachten, daß für $\ell \leq 0$ dann $\frac{1}{1-h\ell} \leq 1$ ist.

3. *Fehlerakkummulierung*:
   Wir erhalten analog zum expliziten Fall:

$$\begin{aligned}\|y_n - y(t_n)\| &\leq \frac{Ch^2}{1-h\ell} + \frac{Ch^2}{(1-h\ell)^2} + \ldots + \frac{Ch^2}{(1-h\ell)^{n-1}} + \frac{Ch^2}{(1-h\ell)^n} \\ &= \frac{Ch^2}{1-h\ell} \frac{\left(\frac{1}{1-h\ell}\right)^n - 1}{\frac{1}{1-h\ell} - 1}\end{aligned}$$

   Wegen $\frac{1}{1-h\ell} = 1 + \frac{h\ell}{1-h\ell}$ und der allgemeinen Abschätzung $1 + x \leq e^x$ gilt dann

$$\begin{aligned}\|y_n - y(t_n)\| &\leq \frac{Ch^2}{1-h\ell} \frac{\exp\left(\frac{nh\ell}{1-h\ell}\right) - 1}{\frac{h\ell}{1-h\ell}} \\ &= Ch \frac{e^{nh\ell/(1-h\ell)} - 1}{\ell}\end{aligned}$$

   Damit folgt die Behauptung. □

5.23 Bemerkung (Numerische Lösung des nichtlinearen Gleichungssystems)
Wir haben beim impliziten Euler-Verfahren die Gleichung

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$$

wobei wir y_{n+1}, welches nur implizit gegeben ist, berechnen.
Dies machen wir über eine einfache Fixpunktiteration:

$$y_{n+1}^{(k+1)} = y_n + hf\left(t_{n+1}, y_{n+1}^{(k)}\right)$$

Nach dem Banachschen Fixpunktsatz konvergiert dies, falls $y \mapsto y_n + hf(t_{n+1}, y)$ eine strikte Kontraktion ist, d.h. $hL < 1$, wobei L eine Lipschitzkonstante von f ist.
Dies führt allerdings zu einer Schrittweiteneinschränkung, die im wesentlichen wie beim expliziten Euler-Verfahren aussieht.
Das wollten wir allerdings vermeiden, denn bei bei steifen Differentialgleichungen ist das ungeeignet.

Daher wenden wir nun das Newton-Verfahren zur Lösung von $F(y) = 0$ an:

$$y^{(k+1)} = y^{(k)} - F'(y^{(k)})^{-1} F(y^{(k)})$$

Wir wissen bereits, daß wir „gute“ Startwerte zur Verfügung haben, also können wir das *vereinfachte* Newton-Verfahren benutzen, d.h.

$$y^{(k+1)} = y^{(k)} - F'(y^{(0)})^{-1} F(y^{(k)})$$

und hier ist $F(y) = y - y_n - hf(t_{n+1}, y)$, damit ist

$$F'(y) = I - h\frac{\partial f}{\partial y}(t_{n+1}, y)$$

Wir berechnen iterativ

$$y_{n+1}^{(k+1)} = y_{n+1}^{(k)} + \Delta y_{n+1}^{(k)}$$

mit $y_{n+1}^{(0)} = y_n$, wobei $\Delta y_{n+1}^{(k)}$ aus dem linearen Gleichungssystem kommt:

$$(I - hJ)\Delta_{n+1}^{(k)} = -\left(y_{n+1}^{(k)} - y_n - hf(t_{n+1}, y_{n+1}^{(k)})\right)$$

wobei $J \approx \frac{\partial}{\partial y} f(t_{n+1}, y_{n+1})$ die Jacobi-Matrix ist.

Typischerweise beobachtet man, daß 1-3 Iterationen meistens ausreichend sind, sodaß

$$\|y_{n+1}^{(k)} - y_{n+1}\| < \text{ lokaler Fehler des impliziten Euler-Verfahrens}$$

Eine größere Genauigkeit ist beim Lösen des LGS gar nicht sinnvoll.

5.24 Definition (lineares implizites Euler-Verfahren)
Beim *linearen impliziten Euler-Verfahren* führt man nur eine Newton-Iteration durch und nimmt $y_{n+1}^{(1)}$ als Startwert für den nächsten Schritt.

5.25 Bemerkung (Aufwand beim impliziten Euler-Vefahren)
Wir sehen, daß ein impliziter Euler-Schritt wesentlich aufwändiger ist als ein expliziter Euler-Schritt.
Wir müssen nämlich zusätzlich berechnen:

1. $\frac{\partial f}{\partial y}$-Auswertung. Dies ist eine $d \times d$-Matrix, die man näherungsweise mit nummerischer Differentiation berechnen muß, wobei man $d + 1$ Funktionsauswertungen hat.
   Wir benutzen für mehrere Schritte dieselbe Ableitungsmatrix J, welches wir über die Konvergenzgeschwindigkeit des vereinfachten Newton-Verfahrens steuert.
2. LR-Zerlegung von $I - hJ$, wofür man etwa d^3 Rechenschritte benötigt.
3. Rücksubstitution für das Lösen des LGS
4. Zusätzlich eine Funktionswerte pro Iteration

Beim expliziten Euler-Verfahren hat man allerdings nur eine einzige Funktionsauswertung.

Dieser zusätzliche Aufwand lohnt sich bei steifen Differentialgleichungen, weil dort aus Stabilitätsgründen größere Schrittweiten als beim explitite Euler-Verfahren gewählt werden können, um eine vorgegebene Genauigkeit zu erreichen.

5.26 Bemerkung (Nachteile beider Euler-Verfahren)
Wir haben sowohl beim expliziten als auch beim impliziten Euler-Verfahren verschiedene Nachteile:

1. Wir haben eine geringe Genauigkeit: Der Fehler ist proportional zu $\mathcal{O}(h)$.

   Wir betrachten im Folgenden Verfahrensklassen, der Genauigkeit $\mathcal{O}(h^p)$ mit $p > 1$, wobei das p die Ordnung des Verfahrens ist.

2. Wir möchten ein Verfahren haben, bei dem wir die Schrittweiten variieren können.

5.27 Bemerkung (Ausblick)
Wir werden als nächstes folgende Verfahren besprechen:

1. Runge-Kutta-Verfahren, die auf Quadraturformeln aufbauen.
2. Extrapolationsverfahren, die auf das Euler-Verfahren aufbauen und dieses Extrapolieren.
3. Mehrschrittverfahren, bei denen nicht nur der letzte Schritt, sondern auch die vorherigen Schritte berücksichtigt werden.

Dabei muß jeweils zwischen Verfahren für nichtsteife und steife Differentialgleichungen unterschieden werden.

5.5 Runge-Kutta-Verfahren

5.28 Motivation
Wir betrachten eine Differentialgleichung $y' = f(t, y)$ und $y(t_0) = y_0$ auf $I = [t_0 + T]$

5.29 Algorithmus
Wir unterteilen unser Intervall in $t_i = t_0 + i \cdot h$ und wissen, daß

$$y(t_1) = y_0 + \int_{t_0}^{t_1} y'(t)\,dt$$

Wir tun nun so, als würden wir bereits $y'(t)$ kennen und ersetzen das Integral durch eine Quadraturformel mit Knoten $c_1, \ldots, c_s$ und den Gewichten $b_1, \ldots, b_s$ und erhalten:

$$y(t_1) \approx y_0 + h \sum_{i=1}^{s} b_i \underbrace{y'(t_0 + c_i h)}$$

Wir kennen aber den hinteren Ausdruck gar nicht.
Wir wissen aber, daß

$$y'(t_0 + c_i h) = f(t_0 + c_i h, y(t_0 + c_i h))$$

ist, wobei wir wiederrum die rechte Seite nicht kennen. Diese können wir aber wie $y(t_1)$ berechnen:

$$y(t_0 + c_i h) = y_0 + \int_{t_0}^{t_0 + c_i h} y'(t)\,dt \approx y_0 + h \sum_{j=1}^{s} a_{ij} y'(t_0 + c_j h)$$

Nun setzen wir

$$y_1 = y_0 + h \sum_{i=1}^{s} b_i Y_i'$$

mit

$$Y_i' = f(t_0 + c_i h, Y_i)$$

Damit erhalten wir

$$Y_i = y_0 + h \sum_{j=1}^{s} a_{ij} Y_j'$$

5.30 Algorithmus (Explizites Runge-Kutta-Verfahren)
Falls $a_{ij} = 0$ für $j \geq i$, dann können die Y_i explizit nacheinander berechnet werden.
Man spricht dann vom *expliziten Runge-Kutta-Verfahren* und erhält den folgenden Algorithmus:

$$\begin{array}{lll} Y_1 = y_0 & & Y_1' = f(t_0 + c_1 h, Y_1) \\ Y_2 = y_0 + h a_{21} Y_1' & & Y_2' = f(t_0 + c_2 h, Y_2) \\ \ldots Y_i & = y_0 + h \sum_{j=1}^{i-1} a_{ij} Y_j' Y_i' & = f(t_0 + c_i h, Y_i) \end{array}$$

Zuletzt setzen wir:

$$y(t_1) = y_1 = y_0 + h \sum_{j=1}^{s} b_j Y_j'$$

Wir benötigen s Funktionsauswertungen im Schritt $y_0 \to y_1$.

Das Runge-Kutta-Verfahren ist also durch die Koeffizenten a_{ij}, b_j und c_i festgelegt, was man üblicherweise in einem Schema schreibt:

$$\begin{array}{c|c} c_i & a_{ij} \\ \hline & b_j \end{array}$$

5.31 Beispiel

1. Das explizite Euler-Verfahren gehört zur Klasse der Runge-Kutta-Verfahren mit $s = 1$ und man kann es darstellen als
$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$$
2. Das implizites Euler-Verfahren gehört ebenfalls zur Klasse der Runge-Kutta-Verfahren mit $s = 1$:

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

5.32 Algorithmus (Verfahren von Runge)
RUNGE kam 1895 auf die Idee, das Euler-Verfahren zur Mittelpunktsregel zu modifizieren und setzte deshalb:

$$y_1 = y_0 + hf\left(t_0 + \frac{h}{2}, \underbrace{y_0 + \frac{h}{2} f(t_0, y_0)}\right)$$

wobei der hintere Term einen Euler-Schritt zur Schrittweite $\frac{h}{2}$ entspricht.
Damit erhalten wir folgenden Algorithmus:

$$\begin{aligned} Y_1 &= y_0 \\ Y_1' &= f(t_0, y_0) \\ Y_2 &= y_0 + \frac{h}{2} Y_1' \\ Y_2' &= f(t_0 + \frac{h}{2}, Y_2) \\ y_1 &= y_0 + hY_2' \end{aligned}$$

Wir erhalten ein zweistufiges Verfahren mit den folgenden Koeffizienten:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array}$$

5.33 Bemerkung (Fehlerabschätzung)
Wir sehen, daß man durch eine Taylorentwicklung der numerischen Lösung folgendes erhält:

$$y_1 = y_0 + hf(t_0, y_0) + \frac{h^2}{2}\left(\frac{\partial f}{\partial t}(t_0, y_0) + \frac{\partial f}{\partial y}(t_0, y_0) f(t_0, y_0)\right) + \mathcal{O}(h^3)$$

Für die exakte Lösung erhalten wir die folgende Taylorentwicklung:

$$y(t_0 + h) = \underbrace{y(t_0)}_{y_0} + h \underbrace{y'(t_0)}_{f(t_0, y_0)} + \frac{h^2}{2} y''(t_0) + \mathcal{O}(h^3)$$

Dies ergibt sich, weil

$$y''(t) = \frac{\partial f}{\partial t}(t, y(0)) + \frac{\partial f}{\partial y}(t, y(0)) \underbrace{y'(t)}_{= f(t_0, y_0)}$$

Damit entsprechen sich die Terme bis auf ihre h^3-Ordnungen.
Damit ist der lokale Fehler:

$$y_1 - y(t_0, h) = \mathcal{O}(h^3)$$

was wesentlich besser als das Euler-Verfahren ist.

5.34 Algorithmus (Das klassische Runge-Kutta-Verfahren)
Hierbei erweitert man die Simpson-Regel zur numerischen Lösung von Differentialgleichungen. Man erhält das folgende Koeffizienten-Tablaeu, welches vierstufig ist ($s = 4$):

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

5.35 Bemerkung (Fehlerabschätzung)
Durch Taylorentwicklung ergibt sich als lokaler Fehler

$$y_1 - y(t_0 + h) = \mathcal{O}(h^5)$$

5.36 Definition (Ordnung eines Runge-Kutta-Verfahrens)
Ein RKV hat die Ordnung p genau dann, wenn für jedes AWP $y' = f(t, y), y(t_0) = y_0$ mit einem $(p + 1)$-mal stetig diffbaren f für den lokalen Fehler gilt:

$$y_1 - y(t_0 + h) = \mathcal{O}(h^{p+1})$$

5.37 Beispiel

1. Das Euler-Verfahren hat Ordnung 1
2. Das Verfahren von Runge hat Ordnung 2
3. Das klassische Runge-Kutta-Verfahren hat Ordnung 4

5.38 Bemerkung (Anwendung in der Praxis)
Heutzutage verwendet man das Verfahren von Dormand/Prince (1985) mit Ordnung 5 bzw. 8, was man in *Hairer/Hobelt/Wanner: Solving orderary differential equations I* nachlesen kann.

5.39 Satz (Globaler Fehler)
Sei $f \in \mathcal{C}^{p+1}$ und das RKV habe die Ordnung p. Dann gilt für den globalen Fehler:

$$\|y_n - y(t_n)\| \leq M \cdot h^p, \quad t_n = t_0 + nh \in [t_0, T]$$

wobei M eine Konstante unabhängig von n und h ist.

Beweis
Dieser Beweis verläuft völlig analog zum Euler-Verfahren in denselben drei Schritten.
Wir schreiben nächst die Iteration auf:

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

wobei

$$\Phi(t, y, h) = \sum_{i=1}^{s} b_i Y_i'(t, y, h)$$

mit

$$Y_i'(t, y, h) = f(t_0 + c_i h, Y_i(t, y, h))$$
$$Y_i(t, y, h) = y + h \sum_{j=1}^{s} a_{ij} Y_i'(t, y, h)$$

1. Für den lokalen Fehler ergibt sich, daß dieser $\leq C \cdot h^{p+1}$, was sich direkt durch die Definition der Ordnung (und Kompaktheitsargumente) ergibt.

2. Für die Fehlerfortpflanzung wissen wir, daß Φ (lokal) Lipschitz-stetig ist, denn sie ist eine Hintereinanderausführung von (lokal) Lipschitz-stetigen Funktionen. Damit ist

   $$L_\Phi \leq \sum_{i=1}^{s} |b_i| L_f + \mathcal{O}(h)$$

   Für zwei verschiedene Iterationen:

   $$v_{n+1} = v_n + h\Phi(t_n, v_n, h)$$
   $$w_{n+1} = w_n + h\Phi(t_n, w_n, h)$$

   ergibt sich der Fehler:

   $$\|v_{n+1} - w_{n+1}\| \leq (1 + hL_\Phi)\|v_n - w_n\|$$

3. Über den Fächer der Lady Windermere ergibt sich über das analoge Argument wie beim Euler-Verfahren erhalten wir:

   $$\|y_n - y(t_n)\| \leq M \cdot h^p$$

   mit

   $$M = \frac{e^{L_\Phi(T - t_0)} - 1}{L_\Phi} \cdot C$$

□

5.40 Bemerkung (Ausblick)
Welche Bedingungen müssen die Runge-Kutta-Koeffizienten a_{ij}, b_j, c_i erfüllen, damit das RKV die Ordnung p hat?
Kann man diese Koeffizienten konstruieren?

Wir werden im folgenden Kapitel eine Antwort auf die erste Frage finden.
Die zweite Frage ist relativ komplex und würde den Rahmen dieser Vorlesung sprengen.

5.6 Taylor-Entwicklungen und Bäume

5.41 Motivation
Wir betrachten in diesem Abschnitt eine autonome Differentialgleichung $y' = f(y)$ und sei $f \in \mathcal{C}^\infty$

5.42 Erinnerung (Taylorentwicklung)
Die Taylorentwicklung war

$$y(t_0 + h) = \sum_{k=0}^{p} y^{(k)}(t_0)\frac{h^k}{k!} + \mathcal{O}(h^{p+1})$$

5.43 Konstruktion (Entwicklungen der Differentialgleichungen)
Wir erhalten durch Ableiten die folgenden Ausdrücke:

$$\begin{aligned}
y' &= f(y) \\
y'' &= f'(y)y' = f'(y)f(y) = f'f \\
y''' &= f''(y)(y', y') + f'(y)y'' = f''(y)(f(y), f(y)) + f'(y)f'(y)f(y) = f''(f, f) + f'f'f \\
y^{(4)} &= f'''(y)(y', y', y') + 3f''(y)(y', y'') + f'(y)y'''
\end{aligned}$$

usw. Wir sehen darin bereits eine gewisse Struktur.
Die allgemeine Ableitung $y^{(k)}$ ist eine Linearkombination von

$$y^{(k)} = f^{(m)}(y^{(k_1)}, \ldots, y^{(k_m)}) \text{ mit } \sum_{i=1}^{m} k_i = k - 1 \quad 1 \leq m \leq k - 1$$

5.44 Idee (graphische Darstellung)
$f^{(m)}$ sei ein Knoten mit m Ästen und $y^{(k_1)}, \ldots, y^{(k_m)}$ werden auf die Äste geschrieben.

5.45 Definition (Bäume)
Wir bezeichnen mit $\mathcal{T}$ (trees) die Menge der *Bäume* (mit ausgezeichnetem Knoten, der Wurzel), diese wird auf die folgende Weise rekursiv definiert:

1. $\bullet \in \mathcal{T}$
2. Mit $\tau_1, \ldots, \tau_m \in \mathcal{T}$ mit $m \in \mathbb{N}$ ist auch das ungeordnete m-Tupel $\tau = [\tau_1, \ldots, \tau_m] \in \mathcal{T}$. Graphisch wird dabei an alle vorhandenen Bäume eine Wurzel unten drangehängt.

5.46 Definition (Ordnung eines Baumes)
Die Ordnung eines Baumes ist die Anzahl seiner Knoten, d.h.

1. $|\bullet| = 1$
2. $|\tau| = |\tau_1| + \ldots + |\tau_m| + 1$ für $\tau = [\tau_1, \ldots, \tau_m]$

5.47 Definition (Elementares Differential)
Wir definieren rekursiv das *elementare Differential*:

1. $F(\bullet) = f(y_0)$
2. Für $\tau = [\tau_1, \ldots, \tau_m]$ ist $F(\tau) = f^{(m)}(y_0)\,(F(\tau_1), \ldots, F(\tau_m))$

5.48 Satz (Aussehen der k-ten Ableitung mittels Bäumen)
Es gilt für die Ableitung von y:

$$y^{(k)}(t_0) = \sum_{\tau \in \mathcal{T}, |\tau| = k} \alpha(\tau) F(\tau)$$

für gewisse ganzzahlige positive Koeffizienten $\alpha(\tau) \in \mathbb{N}$, welche nicht von der Differentialgleichung abhängen.

Beweis
Wir beweisen dies per Induktion, wobei wir den Induktionsanfang bereits besprochen haben. Wir fahren also mit dem Induktionsschritt direkt fort

Wir wissen, daß $y^{(k)}(t_0)$ eine Linearkombination von

$$f^{(m)}(y_0)\left(y^{(k_1)}(t_0), \ldots, y^{(k_m)}(t_0)\right) \text{ mit } \sum k_i = k - 1$$

ist.
Wir benutzen nun die Induktionsvoraussetzung, daß für $\ell < k$ dann $y^{(\ell)}(t_0)$ eine Linearkombination von elementaren Differentialen zu Bäumen der Ordnung ℓ ist.
Damit ist

$$\underbrace{f^{(m)}(y_0)\,(F(\tau_1), \ldots, F(\tau_m))}_{F(\tau)}$$

mit $|\tau_1| + \ldots + |\tau_m| + 1 = k$ und $\tau = [\tau_1, \ldots, \tau_m]$. □

5.7 Bedingungsgleichung für das Runge-Kutta-Verfahren

5.49 Konstruktion (Taylorentwicklung des Runge-Kutta-Verfahrens)
Sei $y' = f(y)$ und $y(t_0) = y_0$. Dann ist

$$y_1 = y_0 + h \sum_{i=1}^{s} b_i Y_i'$$
$$Y_i = y_0 + h \sum_{j=1}^{s} a_{ij} Y_j'$$

mit $Y_i' = f(Y_i)$.
Wir betrachten nun y_i, Y_i, Y_i' als Funktionen von h und betrachten dazu die Taylorentwicklungen.
Hierzu betrachten wir die Ableitungen nach h:

$$\begin{aligned} Y_i' &= f(Y_i) \\ \dot{Y}_i' &= f'(Y_i)\dot{Y}_i \\ \ddot{Y}_i' &= f''(Y_i)(\dot{Y}_i, \dot{Y}_i) + f'(Y_i)\ddot{Y}_i \dots \end{aligned}$$

Allgemein erhält man

$$f^{(m)}(Y_i)\left(Y_i^{(k_1)}, \dots, Y_i^{(k_m)}\right) \text{ mit } k_1 + \ldots + k_m = k-1 \quad \text{für } 1 \le m \le k-1$$

mit denselben Koeffizienten wie bei $y^{(k)}$.

Wir wollen nun

$$Y_i = y_0 + h\sum_{j=1}^{s} a_{ij} Y_j'$$

nach h um die Stelle 0 mit Taylor entwickeln. Dabei müssen wir die Leibniz-Regel verwenden:

$$(u \cdot v)^{(\ell)} = \sum_{j=1}^{\ell} \binom{\ell}{j} u^{(j)} v^{(\ell-j)}$$

Damit erhalten wir:

$$Y_i^{(\ell)} = h\sum_{j=1}^{s} a_{ij}\left(Y_j'\right)^{(\ell)} + \ell\sum_{j=1}^{s} a_{ij}(Y_j')^{(\ell-1)}$$

da wir aber an der Stelle $h = 0$ auswerten, erhalten wir:

$$Y_i^{(\ell)}(0) = \ell\sum_{j=1}^{s} a_{ij}(Y_j')^{\ell-1}(0)$$

Dies setzen wir wieder in $f^{(m)}(Y_i)$ ein und erhalten, daß $(Y_i')^{(k-1)}(0)$ eine Linearkombination ist von

$$k_1 \cdot \ldots \cdot k_m \sum_{j_1=1}^{s} \cdot \ldots \cdot \sum_{j_m=1}^{s} a_{ij_1} \cdot \ldots \cdot a_{ij_m} f^{(m)}(y_0)\left((Y_{j_1}')^{(k_1-1)}(0), \dots, (Y_{j_m}')^{(k_m-1)}(0)\right)$$

mit denselben Koeffizienten wie zuvor und wir wissen außerdem, daß $k_1 + \ldots + k_m = k-1$ für $m \le k-1$ ist.

Wir wollen nun versuchen, diese Formel rekursiv zu verwenden.

5.50 Definition (elementare Differentiale)
Für $\tau \in \mathcal{T}$ setzen wir rekursiv:

$$\varphi_i'(\bullet) = 1$$
$$\varphi_i'(\tau) = |\tau_1| \cdot \ldots \cdot |\tau_m| \sum_{j_1,\ldots,j_m} a_{ij_1} \cdot \ldots \cdot a_{ij_m} \cdot \varphi_{j_1}'(\tau_1) \cdot \ldots \cdot \varphi_{j_m}'(\tau_m)$$

für $\tau = [\tau_1, \ldots, \tau_m]$ und wir erhalten damit

$$\varphi(\tau) = |\tau| \cdot \sum_{i=1}^{s} b_i \varphi_i'(\tau)$$

5.51 Satz
Es gilt

$$y_1^{(k)}(0) = \sum_{\tau \in \mathcal{T}, |\tau| = k} \varphi(\tau)\alpha(\tau)F(\tau)$$

wobei α wie im vorherigen Kapitel definiert ist.

Beweis
Die obige Formel und eine Induktion liefern, daß

$$(Y_i')^{(k-1)}(0) = \sum_{\tau \in \mathcal{T}, |\tau| = k} \varphi_i'(\tau)\alpha\tau F(\tau)$$

und wir erhalten damit:

$$y_1^{(k)}(0) = k \sum_{i=1}^{s} b_i (Y_i')^{(k-1)}(0)$$

□

5.52 Satz (Hinreichende Bedingung für die Ordnung)
Falls

$$\varphi(\tau) = 1 \quad \forall \tau \in \mathcal{T}, |\tau| \leq p$$

gilt, dann hat das Runge-Kutta-Verfahren die Ordnung p.

Beweis
Sei $\varphi(\tau) = 1$ für alle Bäume $\tau \in \mathcal{T}$ der Ordnung $|\tau| \leq p$.
Seien $\Phi_i'(\tau), \Phi(\tau)$ wie φ_i', φ definiert, aber ohne die Faktoren $|\tau_j|$ und $|\tau|$. Dann gilt

$$\varphi(\tau) = \gamma(\tau)\Phi(\tau)$$

wobei $\gamma(\tau)$ rekursiv definiert ist durch

$$\gamma(\bullet) = 1$$
$$\gamma(\tau) = |\tau|\gamma(\tau_1) \cdot \ldots \cdot \gamma(t_m)$$

für $\tau = [\tau_1, \ldots, \tau_m]$.
Die Bedingungsgleichung $\varphi(\tau) = 1$ ist damit äquivalent zu $\Phi(\tau) = \frac{1}{\gamma(\tau)}$.
$\Phi(\tau)$ berechnet man leicht wie folgt:
Wir fügen zu jedem Knoten einen Summationsindex $i, j, k, \ldots$ hinzu. Dann ist

$$\Phi(\tau) = \sum_{i,j,k,\ldots} b_i \psi_{i,j,k,\ldots}$$

wobei $\psi_{i,j,k,\ldots}$ ein Produkt ist mit Faktoren a_{jn}, wenn ein Knoten j mit einem höheren Knoten n verbunden ist. □

5.53 Bemerkung (Konstruktion expliziter Runge-Kutta-Verfahren)
Sei $a_{ij} = 0$ für $i \leq j$.
Dann setzen wir

$$y_1 = y_0 + h \sum_{i=1}^{s} b_i Y_i'$$
$$Y_i = y_0 + h \sum_{j=1}^{i-1} a_{ij} Y_j'$$

mit $Y_i' = f(Y_i)$ mit s als „Stufenzahl“.
Wir möchten eine möglichst hohe Ordnung p mit möglichst kleiner Stufenzahl s haben.

5.54 Satz (Höchste Ordnung eines expliziten Runge-Kutta-Verfahrens)
Es gilt stets: $s \geq p$.

Beweis
Wir betrachten die Bedingungsgleichung für den längsten Baum der Ordnung p. Dann ist

$$\sum_{i_1,\ldots,i_p} b_{ip} a_{i_1,i_2} \cdot \ldots \cdot a_{i_{p-1},i_p} = \frac{1}{p!}$$

Beim expliziten RKV ist $a_{ij} = 0$ für $i \leq j$. Damit der Summand ungleich Null ist, muß aber $s \geq i_1 > i_2 > i_3 > \ldots > i_p \geq 1$ gelten.
Und damit muß $s \geq p$ sein. □

5.55 Zusammenfassung (Übersicht der Ordnungen)
Wir können nun die uns bekannten Verfahren einordnen:

	s	p
Euler	1	1
Runge	2	2
klass. RK	4	4

Es gibt kein 5-stufiges RKV der Ordnung 5.
Eine kleine Übersicht über größere Ordnungen erhalten wir durch folgende Bedingung:

p	1	2	3	4	5	6	7	8	9	10
s	1	2	3	4	6	7	9	11	≥ 12	≥ 13
Anz. Bedindungsgleichungen	1	2	4	8	17	37	85	200	486	1205

5.8 Schrittweitensteuerung beim Runge-Kutta-Verfahren

5.56 Motivation
Wir möchten nun die Schrittweiten so steuern, daß der Fehler in jedem Teilintervall akzeptabel ist.
Verwenden wir eine äquidistante Unterteilung, so haben wir teilweise einen hohen lokalen Fehler und teilweise einen sehr kleinen lokalen Fehler.

Diese Problematik haben wir bereits bei den Quadraturformeln kennengelernt.
Unser Ziel ist es, das Intervall, in dem die Lösung berechnet werden soll, so zu unterteilen, daß die *lokalen Fehler* in jedem Teilintervall etwa gleich groß sind und unterhalb einer gewissen Fehlertoleranz liegt.

5.57 Idee
Zur Schätzung des lokalen Fehlers benutzen wir eine Idee:
Wir verwenden nicht ein Runge-Kutta-Verfahren, sondern wir verwenden zwei Runge-Kutta-Verfahren, die die selben Funktionsauswertungen benutzen. Dies verläuft wieder analog zur Idee der eingebetteten Quadraturformel. Die Funktionsauswertungen stecken also in den Y_i', d.h. das zweite Runge-Kutta-Verfahren muß die gleichen Koeffizienten a_{ij} haben. Das andere Runge-Kutta-Verfahren hat also eine andere Gewichtung und wir nennen dies analog ebenfalls *eingebettetes Runge-Kutta-Verfahren.*

5.58 Konstruktion (Eingebettetes Kunge-Kutta-Verfahren)
Wir benutzen 2 RKV mit denselben Funktionsauswertungen:

$$Y_i' = f(t_0 + c_i h, Y_i), \qquad Y_i = y_0 + h \sum_{j=1}^{s} a_{ij} Y_j'$$

und haben ein RKV der Ordnung p:

$$y_1 = y_0 + h \sum_{i=1}^{s} b_i Y_i'$$

und ein eingebettes RKV der Ordnung $p+1$:

$$\hat{y}_1 = y_0 + h\sum_{i=1}^{s} \hat{b}_i Y_i'$$

Wir betrachten nun die lokalen Fehler

$$y_1 - y(t_0+h) = Ch^{p+1} + \mathcal{O}(h^{p+2})$$
$$\hat{y}_1 - y(t_0+h) = \mathcal{O}(h^{p+2})$$

mit $C = C(t_0, y_0)$

Wir vernachlässigen alle Terme der Ordnung h^{p+2} und bezeichnen den lokalen Fehler von y_1 mit

$$\texttt{err} := y_1 - \hat{y}_1 = h\sum_{i=1}^{s} e_i Y_i' \quad \text{mit } e_i = b_i - \hat{b}_i$$

Wir erinnern uns an die Bedingungsgleichungen und wissen über unsere RKV:

$$|\tau|\sum_{i=1}^{s} b_i \varphi_i'(\tau) = 1 \qquad \forall \tau \in \mathcal{T} \text{ mit } |\tau| \le p$$
$$|\tau|\sum_{i=1}^{s} \hat{b}_i \varphi_i'(\tau) = 1 \qquad \forall \tau \in \mathcal{T} \text{ mit } |\tau| \le p+1$$

Damit also $y_1 - \hat{y}_1$ eine sinnvolle Fehlerschätzung ist, sollen für y_1 keine Bedindungsgleichung der Ordnung $p+1$ erfüllt sein, d.h.

$$|\tau|\sum_{i=1}^{s} b_i \varphi_i'(\tau) \neq 1$$

für mindestens ein $\tau \in \mathcal{T}$ mit $|\tau| = p+1$.

5.59 Bemerkung (Praxistauglichkeit)
In der Praxis wird gewöhnlich das Programm `DOPRI5` verwendet, was ein RKV der Ordnung 5 mit einem eingebetten RKV der Ordnung 6 benutzt.

5.60 Bemerkung (Schrittweitensteuerung)
Auf der Schätzung von `err` beruht die *Schrittweitensteuerung.*
Wir geben uns eine Fehlertoleranz `tol` vor (z.B. 10^{-3}) und wollen die Schrittweite h bestimmen, sodaß

$$\|\texttt{err}\| \approx \texttt{tol}$$

meist nimmt man für `err` eine skalierte Norm, d.h.

$$\sqrt{\sum_k \left(\frac{\texttt{err}}{s_k}\right)^2}$$

wobei s_k benutzerdefinierte Skalierungsfaktoren sind, z.B.

$$s_k = 1, \qquad s_k = |\hat{y}_{1,k}|, \qquad s_k = \max\{|\hat{y}_{1,k}|, |y_{1,k}|, 10^{-6}\}$$

Hat man für h die Werte $y_1, \hat{y}_1$ und `err` berechnet, so ist

$$\texttt{err} \approx C \cdot h^{p+1}$$

Das Ideale h^* wäre eines, für welches gilt:

$$\|C(h^*)^{p+1}\| = \texttt{tol}$$

d.h. wir berechnen $C = \frac{\texttt{err}}{h^{p+1}}$ und berechnen damit über die Beziehung

$$\|\texttt{err}\| \cdot \left(\frac{h^*}{h}\right)^{p+1} = \texttt{tol}$$

und wir erhalten damit als ideale Schrittweite:

$$h^* = h \sqrt[p+1]{\frac{\texttt{tol}}{\|\texttt{err}\|}}$$

Wir wählen im nächsten Schritt nun dieses h^* als Schrittweite und nehmen dabei an, daß $C(t_0 + h, \hat{y}_1) \approx C(t_0, y_0)$.
Als Anfangswert für den nächsten Schritt nehmen wir den genaueren Wert $\hat{y}_1$.

5.61 Algorithmus (zur Schrittweitensteuerung)
Wir konstruieren eine Funktion zur Berechnung des Runge-Kutta-Verfahrens:
$RK(d, f, t, y, t_{\text{end}}, \texttt{tol}, h)$
Dabei haben wir die folgenden Parameter:

- d: Dimension des Problems
- f ist die Funktion der Differentialgleichung
- t und y sind die Anfangswerte
- t_{end} ist das Ende des Intervalls
- `tol` ist die Fehlertoleranz
- h ist die Anfangsschrittweite, welche später durch die Schrittweitensteuerung überschrieben wird.

Wir gehen dann wie folgt vor:

1. $h = \min\{h, t_{\text{end}} - t\}$
2. Berechne $\hat{y}_1, \texttt{err}$ und $h^* = h \sqrt[p+1]{\frac{\texttt{tol}}{\|\texttt{err}\|}}$

3. Falls $\|\texttt{err}\| > \texttt{tol}$, verwerfen wir den Schritt, setzen $h = 0,9 \cdot h^*$ durch, wobei $0,9$ ein Sicherheitsfaktor ist. Dann führen wir den Schritt 2 nochmals durch.

4. Falls $\|\texttt{err}\| \leq \texttt{tol}$ ist, wird der Schritt akzeptiert und wir setzen:
   $y = \hat{y}_1$ und $t = t + h$
   Evtl. können wir den Zwischenschritt speichern.
   Falls $t < t_{\text{end}}$ ist, setzen wir $h = \min\{0,9 \cdot h^*, 5h\}$, wobei wir die 5 willkürrlich so gewählt haben, daß die Schrittweiten nicht extrem zunehmen.
   Wir fahren mit Schritt 1 weiter.
   Falls $t \geq t_{\text{end}}$ ist, sind wir fertig.

5.9 Beispiele von Mehrschrittverfahren

5.62 Motivation
Wir möchten wie gehabt das Problem $y' = f(t, y)$ und $y(t_0) = y_0$ lösen.
Wir nehmen an, wir haben bereits mehrere Werte berechnet.
Beim Runge-Kutta-Verfahren wird die komplette Vergangenheit vergessen.
Die Idee des Mehrschrittverfahrens ist es, die „alten“ Lösungen mit in die Rechnung einzubeziehen und ohne großen Aufwand zu einem hinreichend guten Ergebnis zu kommen.

In der Klasse der Mehrschrittverfahren gibt es zwei generelle Beispiele: Die Adams-Verfahren und die BDF-Verfahren. Das Adams-Verfahren beruht wie das Runge-Kutta-Verfahren auf der numerischen Integration, während das BDF-Verfahren auf der numerischen Differentiation beruht.

5.63 Konstruktion (Explizites Adams-Verfahren (Adams-Bashforth))
Wir nehmen an, daß wir bereits $y_0, \ldots, y_n$ kennen und möchten y_{n+1} berechnen. Es gilt stets:

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

Wir ersetzen nun den Integranten durch ein Polynom $p(t)$, welches die bereits bekannten Punkte interpoliert, d.h. wir berechnen ein Interpolationspolynom durch die Punkte

$$(t_n, f_n), (t_{n-1}, f_{n-1}), \ldots, (t_{n-k+1}, f_{n-k+1})$$

Wir benutzen zur Interpolation die Newtonsche Interpolationsformel. Wir möchten dazu annehmen, daß unsere Punkte ein äquidistantes Gitter haben und wir bezeichnen

$$\nabla f_n = f_n - f_{n-1}$$
$$\nabla^2 f_n = \nabla f_n - \nabla f_{n-1}$$

wobei $f_n := f(t_n, y_n)$ ist (*Achtung:* Hierbei handelt es sich bei ∇ *nicht* um den Gradienten!)
Wir erhalten damit:

$$p(t) = p(t_n + \theta h) = f_n + \theta \nabla f_n + \frac{\theta(\theta+1)}{2!}\nabla^2 f_n + \ldots + \frac{\theta(\theta+1)\cdot\ldots\cdot(\theta+k-2)}{(k-1)!}\nabla^{k-1} f_n$$

Wir berechnen damit

$$\int_{t_n}^{t_{n+1}} p(t)\,dt = h\int_0^1 p(t_n+\theta h)d\theta = h\left(f_n + \frac{1}{2}\nabla f_n + \frac{5}{12}\nabla^2 f_n + \ldots\right)$$

Wir erhalten dann für die Iteration folgendes Verfahren:

$$\begin{aligned} y_{n+1} &= y_n + \int_{t_n}^{t_{n+1}} p(t)\,dt \\ &= y_n + h\left(f_n + \gamma_1\nabla f_n + \gamma_2\nabla^2 f_n + \ldots + \gamma_{k-1}\nabla^{k-1} f_n\right) \end{aligned}$$

wobei sich für die Koeffizienten γ_i folgendermaßen festgelegt sind:

$$\gamma_i = \int_0^1 \frac{\theta(\theta+1)\cdot\ldots\cdot(\theta+i-1)}{i!}d\theta$$

feste Werte, die nicht von der Differentialgleichung abhängig sind.
Wir fassen hierbei einige Werte in einer Tabelle an:

i	0	1	2	3	4	5
γ_i	1	$\frac{1}{2}$	$\frac{5}{12}$	$\frac{3}{8}$	$\frac{251}{720}$	$\frac{95}{288}$

5.64 Beispiel

1. Für $k=1$ ist $y_{n+1} = y_n + hf_n$, womit wir beim expliziten Euler-Verfahren sind.
2. Für $k=2$ ist $y_{n+1} = y_n + h\left(\frac{3}{2}f_n - \frac{1}{2}f_{n-1}\right)$, falls y_0, y_1 bekannt sind. Damit kann man die restlichen y_i problemlos berechnen.
3. Für $k=3$ ist $y_{n+1} = y_n + h\left(\frac{23}{12}f_n - \frac{16}{12}f_{n-1} + \frac{5}{12}f_{n-2}\right)$

5.65 Bemerkung
Man benötigt immer die ersten y_i für $i \geq k$ Werte. Diese muß man mit einem anderen Verfahren berechnen, um überhaupt einen Rekursionsanfang zu haben. Dies geschieht in der Regel über ein Runge-Kutta-Verfahren oder das Euler-Verfahren.

5.66 Historische Notiz
Dieses Verfahren wurde 1855 von Adams begründet.

5.67 Bemerkung (Problematik)
Wir haben einen recht großen lokalen Fehler, da das Polynom $p(t)$ außerhalb des Interpolationsintervalls $[t_{n-k+1}, t_n]$ verwendet wird.
Wir haben nämlich, wie wir bereits bei den Interpolationspolynomen festgestellt haben, eine schlechte Approximation an $f(t, y(t))$ in $[t_n, t_{n+1}]$, vor allem bei einem höheren Grad.

5.68 Konstruktion (Implizite Adams-Verfahren (Adams-Moulton))
Wir ersetzen $f(t, y(t))$ durch ein Polynom $p^*(t)$, welches ebenfalls wie beim expliziten Verfahren die Punkte $(t_n, f_n), \ldots, (t_{n-k+1}, f_{n-k+1})$ *und* den gesuchten Punkt (t_{n+1}, f_{n+1}) interpoliert.

Damit erhalten wir die folgende Newtonsche Interpolationsformel:

$$\begin{aligned} p^*(t) &= p^*(t_n + \theta h) \\ &= f_{n+1} + (\theta - 1)\nabla f_{n+1} + \frac{(\theta-1)\theta}{2!}\nabla^2 f_{n+1} \\ &\quad + \ldots + \frac{(\theta-1)\theta(\theta+1)\cdot\ldots\cdot(\theta+k-2)}{k!}\nabla^k f_{n+1} \end{aligned}$$

Wir berechnen damit wieder

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} p^*(t)\,dt = y_n + h\left(f_{n+1} + \gamma_1^*\nabla f_{n+1} + \ldots + \gamma_k^*\nabla^k f_{n+1}\right)$$

mit den Koeffizienten

$$\gamma_j^* = \int_0^1 \frac{(\theta-1)\theta(\theta+1)\cdot\ldots\cdot(\theta+j-2)}{j!}d\theta$$

welche wir wieder in einer Tabelle darstellen können:

j	0	1	2	3	4	5
γ_j^*	1	$-\frac{1}{2}$	$-\frac{1}{12}$	$-\frac{1}{24}$	$-\frac{19}{720}$	$-\frac{3}{180}$

5.69 Beispiel

1. Für $k = 0$ erhalten wir das implizite Euler-Verfahren: $y_{n+1} = y_n + hf_{n+1}$
2. Für $k = 1$ erhalten wir die Trapezregel: $y_{n+1} = y_n + h\left(\frac{1}{2}f_{n+1} + \frac{1}{2}f_n\right)$
3. Für $k = 2$ erhalten wir $y_{n+1} = y_n + h\left(\frac{5}{12}f_{n+1} + \frac{8}{12}f_n - \frac{1}{12}f_{n-1}\right)$

5.70 Bemerkung (Lösung eines Iterationsschrittes beim impliziten Verfahren)
Das implizites Adams-Verfahren hat bei den Iterationsschritten nach dem Banachschen Fixpunktsatz eine eindeutige Lösung für y_{n+1}, falls es sich um eine Kontraktion handelt.
Dies ist der Fall, wenn

$$hL\left(\sum_{j=0}^{k}\gamma_j^*\right) < 1$$

für L die Lipschitz-Konstante von f.

Den Startwert des impliziten Adams-Verfahren erhalten wir durch das Anwenden des expliten Adams-Verfahren, d.h. das explizite Adams-Verfahren ist Predictor und das implizite Adams-Verfahren ist ein Corrector, weshalb das Verfahren oft auch als *Predictor-Corrector-Verfahren* genannt wird.

5.71 Konstruktion (BDF-Verfahren)
Die BDF-Verfahren beruhen auf der numerischen Differentiation, dabei steht BDF für „backward differentiation formulas“.

Sei $p(t)$ ein Interpolationspolynom durch die Lösungen

$$(t_{n+1}, y_{n+1}), (t_n, y_n), \ldots, (t_{n-k+1}, y_{n-k+1})$$

Dann ist

$$\begin{aligned} p(t) &= p(t_n + \theta h) \\ &= y_{n+1} + (\theta - 1)\nabla y_{n+1} + \frac{(\theta-1)\theta}{2!}\nabla^2 y_{n+1} \\ &\quad + \ldots + \frac{(\theta-1)\theta(\theta+1)\cdot\ldots\cdot(\theta+k-2)}{k!}\nabla^k y_{n+1} \end{aligned}$$

Wir haben nun verschiedene Möglichkeiten:

1. Wir fordern, daß $p'(t_n) = f(t_n, y_n)$
   Damit erhalten wir, daß
   $$\frac{\partial}{\partial\theta} p(t_n + \theta h) = p'(t_n + \theta h) \cdot h$$
   und damit ist
   $$\delta_1 \nabla y_{n+1} + \delta_2 \nabla^2 y_{n+1} + \ldots + \delta_k \nabla^k y_{n+1} = h f_n$$
   wobei
   $$\delta_j = \frac{\partial}{\partial\theta} \frac{(\theta-1)\theta\cdot\ldots\cdot(\theta+j-2)}{j!}\Big|_{\theta=0}$$
   Für verschiedene k erhalten wir verschiedene Verfahren:

$k=1:$	$y_{n+1} - y_n = h f_n$	explizites Euler-Verfahren
$k=2:$	$\frac{1}{2} y_{n+1} - \frac{1}{2} y_{n-1} = h f_n$	explizite Mittelpunktsregel

   Für $k \geq 3$ ist die Lösung instabil.

2. Wir fordern, daß $p'(t_{n+1}) = f(t_{n+1}, y_{n+1})$ und erhalten damit eine Klasse von impliziten Verfahren und erhalten:
   $$\delta_1^* \nabla y_{n+1} + \delta_2^* \nabla^2 y_{n+1} + \ldots + \delta_k^* \nabla^k y_{n+1} = h f_n$$
   wobei
   $$\delta_j^* = \frac{\partial}{\partial\theta} \frac{(\theta-1)\theta\cdot\ldots\cdot(\theta+j-2)}{j!}\Big|_{\theta=0} = \frac{1}{j}$$
   Für verschiedene k erhalten wir verschiedene Verfahren:

$k=1:$	$y_{n+1} - y_n = h f_{n+1}$	implizites Euler-Verfahren
$k=2:$	$\frac{3}{2} y_{n+1} - 2 y_n + \frac{1}{2} y_{n-1} = h f_{n+1}$	

Diese Verfahren sind für $k \geq 7$ instabil.
Für $1 \leq k \leq 6$ sind diese Verfahren zur Lösung steifer Differentialgleichungen viel verwendet.

5.10 Ordnung von Mehrschrittverfahren

5.72 Motivation
Analog zu den RKV möchten wir einen Ordnungsbegriff für Mehrschrittverfahren einführen.

5.73 Definition (Lineare Mehrschrittverfahren)
Wir betrachten allgemein eine Klasse der *linearen Mehrschrittverfahren* in Erweiterung zu den uns bekannten Beispielen (Adams-Verfahren und BDF-Verfahren), welche wir folgendermaßen charakterisieren wollen:

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h \cdot \sum_{j=0}^{k} \beta_j f_{n+j}$$

mit $\alpha_k \neq 0$ und wenigstens $\alpha_0 \neq 0$ oder $\beta_0 \neq 0$
Dabei sind außerdem die Startwerte $y_0, y_1, \ldots, y_{k-1}$ gegeben, womit wir mit dieser Formel $y_k, y_{k+1}, \ldots$ berechnen können.

Dieses Verfahren ist explizit $\iff \beta_k = 0$

5.74 Definition (Ordnung eines MSV)
Ein Mehrschrittverfahren (MSV) hat genau dann die *Ordnung* p, wenn für jedes Anfangswertproblem
$y' = f(t,y), y(t_0) = y_0$ mit $f \in \mathcal{C}^{p+1}$ gilt bei exakten Startwerten $y_j = y(t_0 + jh)$ für $j = 0, 1, \ldots, k-1$:

$$y_k - y(t_0 + kh) = \mathcal{O}(h^{p+1})$$

5.75 Hilfssatz (erste Charakterisierung der Ordnung eines MSV)
Wir betrachten einen linearen Differenzenoperator

$$\mathscr{L}(y,t,h) = \sum_{j=0}^{k} \left(\alpha_j y(t+jh) - h\beta_j y'(t+jh)\right)$$

wobei $\mathscr{L} : \mathcal{C}^1(\mathbb{R}, \mathbb{R}^d) \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^d$
Dann hat das MSV die Ordnung $p \iff \mathscr{L}(y,t,h) = \mathcal{O}(h^{p+1}) \quad \forall t \in \mathbb{R},\ \forall y : \mathbb{R} \to \mathbb{R}^d$ und $y \in \mathcal{C}^{p+1}$ und für $h \to 0$.

Beweis
Wir betrachten zunächst die Rückrichtung:
Sei y die Lösung einer Differentialgleichung $y' = f(t, y)$. Dann ist

$$\mathscr{L}(y,t,h) = \sum_{j=0}^{k} \left(\alpha_j y(\overbrace{t+jh}^{=t_j}) - h\beta_j f(t+jh, y(t+jh)) \right)$$

$$0 = \sum_{j=0}^{k-1} \alpha_j y(t+jh) + \alpha_k y_k - \sum_{j=0}^{k-1} h\beta_j f(t+jh, y(t+jh)) - h\beta_k f(t+kh, y_k)$$

Die zweite Gleichung ergibt sich aus dem MSV mit exakten Startwerten.
Wir subtrahieren nun die beiden Formeln voneinander und erhalten:

$$0 = \alpha_k(y(t_k) - y_k) - h\beta_k \left(f(t_k, y(t_k)) - f(t_k, y_k)\right) + \mathscr{L}(y,t,h)$$

Für explizite Verfahren ($\beta_k = 0$) ergibt sich trivialerweise die Behauptung.
Sei also $\beta_k \neq 0$ und sei L die (lokale) Lipschitzkonstante von f. Damit erhalten wir

$$|\alpha_k| \cdot \|y(t_k) - y_k\| \leq h|\beta_k| \cdot L\|y(t_k) - y_k\| + \|\mathscr{L}(y,t,h)\|$$

für genügend kleine h, also $h|\beta_k|L \leq \frac{1}{2}|\alpha_k|$ folgt dann:

$$\underbrace{\frac{1}{2}|\alpha_k|}_{\neq 0} \cdot \|y(t_k) - y_k\| \leq \|\mathscr{L}(y,t,h)\| \leq Ch^{p+1}$$

Durch Division erhalten wir damit

$$\|y(t_k) - y_k\| = \mathcal{O}(h^{p+1})$$

Die andere Richtung des Beweises erfolgt vollkommen analog. □

5.76 Satz (Charakterisierung der Ordnung)
Das MSV hat genau dann die Ordnung p, wenn

$$\sum_{j=0}^{k} \alpha_j j^q = q \sum_{j=0}^{k} \beta_j j^{q-1} \quad \forall q \leq p$$

Für $q = 0$ interpretieren wir die Bedingung als

$$\sum_{j=0}^{k} \alpha_j = 0$$

5.77 Bemerkung
Diese Charakterisierung bedeutet, daß die Differentialgleichungen $y' = qt^{q-1}$ für $q \leq p$ bei Vorgabe exakter Startwerte durch das MSV exakt gelöst werden.
Dies erhalten wir, wenn wir die Bedingung mit h^q multiplizieren.

Beweis (des Satzes)
Betrachte wie im Hilfssatz den linearen Differenzenoperator und entwickeln ihn mit der Taylorentwicklung:

$$\begin{aligned}\mathscr{L}(y,t,h) &= \sum_{j=0}^{k} \left(\alpha_j y(t+jh) - h\beta_k y'(t+jh)\right) \\ &= \sum_{j=0}^{k} \left(\alpha_j \sum_{q=0}^{p} \frac{y^{(q)}(t)}{q!} j^q h^q + \mathcal{O}(h^{p+1}) - h\beta_j \sum_{r=0}^{p-1} \frac{y^{(r+1)}(t)}{r!} j^r h^r + \mathcal{O}(h^{p+1}) \right) \\ &= \sum_{q=0}^{p} \frac{y^{(q)}(t)}{q!} h^q \left(\sum_{j=0}^{k} \alpha_j j^q - q \sum_{j=0}^{k} \beta_j j^{q-1} \right) + \mathcal{O}(h^{p+1})\end{aligned}$$

Dabei ergibt sich die letzte Zeile durch eine Umbennenung des Summationsindex $r+1 = q$.
Damit erhalten wir schließlich

$$\sum_{j=0}^{k} \alpha_j j^q = q \sum_{j=0}^{k} \beta_j j^{q-1} \quad \forall q \leq p$$

was aber äquivalent dazu ist, daß $\mathscr{L}(y,t,h) = \mathcal{O}(h^{p+1})$, womit nach dem Hilfssatz die Behauptung folgt. □

5.78 Zusammenfassung (Ordnungen von bekannten Verfahren)

1. Der k-Schritt im impliziten Adams-Verfahren hat die Ordnung $k+1$, weil das IPP $p^*(t) = qt^{q-1}$ für alle $q \leq k+1$ ist.
2. Der k-Schritt im expliziten Adams-Verfahren hat die Ordnung k.
3. Der k-Schritt im BDF-Verfahren hat die Ordnung k.

5.79 Konstruktion (eines MSV mit hoher Ordnung)
Wir möchten nun für $k = 2$ ein MSV konstruieren mit möglichst hoher Ordnung, d.h. wir erhalten:

$$\alpha_0 y_n + \alpha_1 y_{n+1} + \alpha_2 y_{n+2} = h(\beta_0 f_n + \beta_1 f_{n+1})$$

wir wählen wegen $\alpha_2 \neq 0$ dann o.B.d.A. dann $\alpha_2 = 1$, womit wir nur noch vier Koeffizienten zu wählen haben.
Es müssen folgende Bedingungsgleichungen gelten:

$$\begin{aligned} q = 0: &\quad \alpha_0 + \alpha_1 + 1 = 0 \\ q = 1: &\quad \alpha_1 + 2 = \beta_0 + \beta_1 \\ q = 2: &\quad \alpha_1 + 4 = 2\beta_1 \\ q = 3: &\quad \alpha_1 + 8 = 3\beta_1 \end{aligned}$$

Womit wir durch einfache Rechnung erhalten, daß $\alpha_0 = -5, \alpha_1 = 4, \beta_0 = 2, \beta_1 = 4$ sind.

Unser Verfahren hat nun die folgende Iteration:

$$y_{n+2} + 4y_{n+1} - 5y_n = h(f_{n+1} - 2f_n)$$

mit Ordnung 3.

5.80 Beispiel
Sei $y' = y$. Dann ist die exakte Lösung $y(t) = e^t$ und wir würden sehen, daß beim Auftragen die erste Schritt gut aussehen, aber das Schaubild irgendwann beginnt, wild zu oszilieren. Je kleiner die Schrittweite ist, desto schlechter wird das Schaubild.
Dieses Verfahren ist also extrem instabil.

5.81 Bemerkung
Bei MSV ist eine hohe Ordnung – anders als bei Runge-Kutta-Verfahren – für die Konvergenz des Verfahrens nicht hinreichend. Wir versuchen dazu eine Erklärung zu finden.

5.82 Bemerkung (Erklärung für die Instablität)
Wir haben in unserem Beispiel $y_{n+2} + 4y_{n+1} - 5y_n = 0$ für $h \to 0$.
Wir untersuchen dies mit dem charakteristischen Polynom und erhalten

$$\zeta^2 + 4\zeta - 5 = (\zeta + 5)(\zeta - 1)$$

Damit ist

$$y_n = C_1(+1)^n + C_2(-5)^n$$

Dabei sorgt der hintere Term für eine starke Oszillation, da er selbst für kleine C_2 dominiert, wenn n größer wird.

Wir müssen also dafür sorgen, daß das charakterisische Polynom keine großen negativen Nullstellen hat.

5.11 Stabilität von Mehrschrittverfahren

5.83 Erinnerung
Wir hatten beim Mehrschrittverfahren die Konstruktion

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h \sum_{j=0}^{k} \beta_j f_{n+j}$$

5.84 Definition (Stabilität)
Ein MSV heißt *stabil* (0-stabil), falls alle Lösungen der Differenzengleichung

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = 0$$

für $n \to \infty$ beschränkt bleiben.

5.85 Satz (Charakterisierung der Stabilität bei MSV)
Ein MSV ist genau dann stabil, wenn die NST $\zeta_1, \ldots, \zeta_k$ von

$$\alpha(\zeta) = \sum_{j=0}^{k} \alpha_j \zeta^j$$

die folgenden Bedingungen erfüllen:

1. $|\zeta_j| \leq 1 \quad \forall j \leq k$
2. Falls $|\zeta_j| = 1$ für ein j, so ist dieses ζ_j eine einfache NST.

Beweis
Seien $\zeta_1, \ldots, \zeta_\ell$ die verschiedenen NST von $\alpha(\zeta)$ mit den Vielfachheiten $m_1, \ldots, m_\ell$, so ist jede Lösung der Differenzengleichung von der folgende Form:

$$y_n = p_1(n)\zeta_1^n + \ldots + p_\ell(n)\zeta_\ell^n$$

wobei p_j ein Polynom vom Grad $\leq m_j - 1$ ist.

Man kann durch Nachrechnen überprüfen, daß dies tatsächlich eine Lösung der Differenzengleichung ist.
Mit einem Dimensionsargument erhält man, daß jede Lösung der Differenzengleichung auch von dieser Gestalt ist.
Die Dimension des Lösungsraums dieser Gleichung ist k.
Aber die Dimension des Raums, der durch $\sum p_j(n)\zeta_j^n$ aufgespannt ist, ist $m_1+m_2+\ldots+m_\ell = k$
Damit hat tatsächlich jede Lösung die oben angegebene Form.

Damit muß für die Beschränktheit gelten, daß alle $|\zeta_j| \leq 1$ und falls $|\zeta_j| = 1$, darf das Polynom davor vom Grad 0 sein, womit die Behauptung gezeigt. □

5.86 Beispiel

1. Beim Adams-Verfahren erhalten wir

$$\alpha(\zeta) = \zeta^k - \zeta^{k-1} = (\zeta - 1)\zeta^{k-1}$$

Dies ist stabil, weil 1 immer eine NST von $\alpha(\zeta)$ ist wegen der Ordnungsbedingung

$$\sum_{j=0}^{k} \alpha_j = 0$$

Diese NST tritt jedoch nur einmal auf und alle anderen NST sind 0.

2. Die BDF-Verfahren sind stabil für $k \leq 6$ und instabil für $k \geq 7$.

5.87 Motivation (Frage)
Welche Ordnung kann ein *stabiles* MSV überhaupt haben?

5.88 Satz (erste Ordnungsschranke von Dahlquist)
Für ein stabiles k-Schrittverfahren der Ordnung p gilt:
$p \leq k + 2$, falls k gerade ist und
$p \leq k + 1$, falls k ungerade ist.

5.89 Historische Notiz
Dies ist 1956 herausgefunden worden.

5.12 Konvergenz von MSV

5.90 Erinnerung
Sei im Folgenden $y' = f(t, y), y(t_0) = y_0$ ein Anfangswertproblem und $f : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ mit $f \in \mathcal{C}^\infty$.
Sei außerdem das folgende MSV gegeben mit $f_n = f(t_n, y_n)$:

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h \sum_{j=0}^{k} \beta_j f_{n+j}$$

5.91 Satz (hinreichende Bedingung für die Konvergenz eines MSV)
Das MSV sei *stabil* und habe die Ordnung $p \geq 1$.
Für die Startwerte gelte $y_j - y(t_j) = \mathcal{O}(h^p)$ für $j \leq k - 1$
Dann ist das MSV *konvergent* der Ordnung p, d.h.

$$y_n - y(t_n) = \mathcal{O}(h^p) \text{ glm für } t_n \in [t_0, T]$$

Beweis
Als wesentlich Beweisidee werden wir das MSV in ein Einschritt-Verfahren in einer höheren Dimension umschreiben und dann die Beweismethoden für die Einschrittverfahren anwenden (über den Fächer der Lady Windermere)

Sei $E = \mathbb{R}^d$. Wir setzen nun

$$Y_n = \begin{pmatrix} y_{n+k-1} \\ y_{n+k-2} \\ \vdots \\ y_{n-1} \end{pmatrix} \in E^{k+1}, \qquad Y_0 = \begin{pmatrix} y_{k-1} \\ \vdots \\ y_0 \\ 0 \end{pmatrix} \in E^{k+1}$$

Wir schreiben das MSV nun in ein Verfahren für Y um. Wegen

$$y_{n+k} = -\sum_{j=0}^{k} \frac{\alpha_k}{\alpha_k} y_{n+j} + h \sum_{j=0}^{k} \frac{\beta_j}{\alpha_k} f_{n+j}$$

erhalten wir, wobei wir ohne Einschränkung $\alpha_k := 1$ setzen:

$$Y_{n+1} = \underbrace{\begin{pmatrix} -\alpha_{k-1} & \dots & -\alpha_0 & 0 \\ 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & 0 \end{pmatrix}}_{=:A} Y_n + h \cdot \underbrace{\begin{pmatrix} \beta_k & \dots & \beta_0 \\ 0 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 0 \end{pmatrix}}_{=:B} \underbrace{\begin{pmatrix} f_{n+k} \\ \vdots \\ f_n \end{pmatrix}}_{=:F_{n+1}}$$

Wegen $F_{n+1} = f(Y_{n+1})$ erhalten wir damit

$$Y_{n+1} = AY_n + hBf(Y_{n+1})$$

Der Konvergenzbeweis wird nun ähnlich sein wie beim Euler-Verfahren:

1. lokaler Fehler:
   Wegen
   $$Y(T_n) = \begin{pmatrix} y(t_{n+k+1}) \\ \vdots \\ y(t_{n-1}) \end{pmatrix}$$
   erhalten wir nach der Definition der Ordnung
   $$\|Y_1 - Y(t_1)\| \leq C \cdot h^{p+1}$$
2. Fehlerfortpflanzung:
   Seien
   $$V_{n+1} = AV_n + hBf(V_{n+1})$$
   $$W_{n+1} = AW_n + hBf(W_{n+1})$$
   Durch Subtraktion der beiden Gleichungen ergibt sich
   $$\|V_{n+1} - W_{n+1}\| \leq \|A\| \cdot \|V_n - W_n\| + h\|B\| \cdot L \cdot \|V_{n+1} - W_{n+1}\|$$

wobei L die Lipschitz-Konstante von f ist.
Wir werden später zeigen, daß es wegen der Stabilität des MSV eine Norm auf E^{k+1} gibt, sodaß für die zugehörige Matrixnorm $\|A\| \leq 1$ gilt. Damit ist

$$\|V_{n+1} - W_{n+1}\| \leq \frac{1}{1 - h\|B\|L}\|V_n - W_n\|$$

3. Fehlerakkumulierung:
   Völlig analog zu früheren Beweisen erhalten wir:

   $$y_n - y(t_n) = \mathcal{O}(h^p)$$

Wir müssen nun noch zeigen, daß $\|A\| \leq 1$ gilt, um den Beweis zu vervollständigen.
Dies zeigen wir durch die folgenden beiden Hilfssätze. □

5.92 Bemerkung (Merkhilfe)
Stabilität und Konsistenz implizieren die Konvergenz

5.93 Bemerkung
Tatsächlich gilt hierbei auch die Umkehrung, was aber viel zu aufwendig zum Beweisen wäre. Diese Richtung ist sowieso interessanter.

5.94 Hilfssatz
Sei A eine $(k+1) \times (k+1)$-Matrix mit den EW $\lambda_1, \ldots, \lambda_{k+1}$.
Es gelte außerdem

1. $|\lambda_i| \leq 1 \quad \forall i$
2. Für $|\lambda_i| = 1$, ist λ_i ein geometrisch einfacher EW von A, d.h. der zugehörige Jordan-Block haben die Dimension 1.

Es existiert eine Norm auf E^{k+1}, sodaß für die zugehörige Matrixnorm gilt

$$\|A\| = \sup_{0 \neq x \in E^{k+1}} \frac{\|Ax\|}{\|x\|} \leq 1$$

Beweis
Wir ordnen λ_i so, daß

$$1 = |\lambda_1| = \ldots = |\lambda_k| > |\lambda_{k+1}| \geq \ldots \geq |\lambda_{k+1}|$$

Wir zerlegen dies nun in eine JNF, d.h. es gibt eine Matrix T, sodaß

$$T^{-1}AT = J$$

Dabei steht auf der Nebendiagonale in den Jordanblöcken entweder 0 oder ε.
Wir wählen dann ε so klein, daß $\|J\|_\infty \leq 1$ ist.
Für $x \in E^{k+1}$ setzen wir

$$\|x\| = \|T^{-1}x\|_\infty$$

Dann ist

$$\|Ax\| = \|T^{-1}Ax\|_\infty = \|T^{-1}ATT^{-1}x\|_\infty = \underbrace{\|T^{-1}AT\|_\infty}_{\leq 1} \cdot \underbrace{\|T^{-1}x\|_\infty}_{=\|x\|} \leq \|x\|$$

Damit folgt die Behauptung. □

5.95 Hilfssatz
Sei $\alpha(\zeta) = \sum_{j=0}^{k} \alpha_j \zeta^j$ mit $\alpha_k = 1$ ein Polynom, das die Nullstellenbedingung für die Stabilität von MSV erfüllt. Dann erfüllt

$$A = \begin{pmatrix} -\alpha_{k-1} & \dots & -\alpha_0 & 0 \\ 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & 0 \end{pmatrix}$$

die Voraussetzungen des ersten Hilfssatzes.

Beweis
Wir betrachten $\det(A - \lambda I)$ und entwickeln nach der letzten Spalte und erhalten

$$\begin{aligned} \det(A - \lambda I) &= -\lambda(\pm\alpha_0 - \lambda(\mp\alpha_1 - \lambda(\pm \dots))) \\ &= -\lambda(\alpha_0 + \lambda\alpha_1 + \dots + \lambda^{k-1}\alpha_{k-1} + \lambda^k) \\ &= \lambda\alpha(\lambda) \end{aligned}$$

Womit die Behauptung folgt. □

5.13 Extrapoliertes Euler-Verfahren

5.96 Motivation
Wir betrachten das explizite Euler-Verfahren $y_{n+1} = y_n + hf(t_n, y_n)$ zur Differentialgleichung $y' = f(t, y), y(t_0) = y_0$, wobei $f \in \mathcal{C}^\infty$.
Wir bezeichnen im Folgenden $y(t, h) = y_n$ für $t = t_n = t_0 + nh$.

5.97 Proposition (Asymptotische h-Entwicklung des Fehlers)
Es existieren $e_i \in \mathcal{C}^\infty$, die die folgende Entwicklung erlauben:

$$y_n = y(t, h) = y(t) + he_1(t) + h^2 e_2(t) + \dots + h^N e_N(t) + \mathcal{O}(h^{N+1})$$

wobei $e_i(t_0) = 0$.

Beweis

1. Wir konstruieren eine abgebrochene h-Entwicklung:

   $$\hat{y}(t,h) = \hat{y}_n = y(t) + he_1(t) + h^2 e_2(t) + \ldots + h^N e_N(t)$$

   so, daß $\hat{y}_0 = y_0$.
   Dabei ist

   $$\hat{y}_{n+1} = \hat{y}_n + h\hat{f}(t_n, \hat{y}_n) = \hat{y}_n + hf(t_n, \hat{y}_n) + \mathcal{O}(h^{N+2})$$

   Diese Formel müssen wir zeigen.

   Dies zeigen wir über Induktion:
   $N = 1$: Wir setzen $\hat{y}(t,h) - y(t) = he_1(t)$ und betrachten

   $$\begin{aligned}
   \hat{y}(t+h,h) - \hat{y}(t,h) - hf(t,\hat{y}(t,h)) &= y(t+h) - y(t) \\
   &\quad + h(e_1(t+h) - e_1(t)) - hf(t,\hat{y}(t,h)) \\
   &= \Big(hy'(t) + \frac{h^2}{2}y'(t) + \mathcal{O}(h^3)\Big) \\
   &\quad + h\Big(he_1'(t) + \mathcal{O}(h^2) - h(f(t,y)t)\Big) \\
   &\quad + f'(t,y(t))he_1(t) + \mathcal{O}(h^2) \\
   &= h^2\Big(\underbrace{\frac{1}{2}y''(t) + e_1'(t) - f'(t,y(t))e_1(t)}\Big) \\
   &\quad + \mathcal{O}(h^3)
   \end{aligned}$$

   wobei sich die zweite Zeile über die Taylorentwicklung ergeben hat.
   Wir müssen noch zeigen, daß die markierte Stelle des letzten Terms Null ist, d.h. dies gilt, wenn e_1 die Lösung der folgenden inhomogenen linearen ODE ist:

   $$e_1' = f'(t,y(t))e_1 - \frac{1}{2}y''(t)e_1(t_0) = 0$$

   Weil f glatt ist, ist diese ODE lösbar und damit existiert e_1.

   Für $N = 2$ müssen wir zeigen:

   $$\hat{y}(t,h) - y(t) = he_1(t) + h^2 e_2(t)$$

   wobei e_1 eine Lösung der vorigen ODE sein soll. Dann entwickeln wir dies analog wie beim Fall von $N = 1$ mit Taylor und erhalten:

   $$\begin{aligned}
   \hat{y}(t+h) - \hat{y}(t) - hf(t,\hat{y}(t)) &= \Big(\ldots + \frac{h^3}{6}y'''(t) + \mathcal{O}(h^4)\Big) \\
   &\quad + h\Big(\ldots + \frac{h^2}{2}e_1''(t) + \mathcal{O}(h^3)\Big) \\
   &\quad + h^2(he_2'(t) + \mathcal{O}(h^2)) \\
   &\quad - h\Big[\ldots + f''(t,y(t))h^2 e_2(t) \\
   &\quad + \frac{1}{2}f''(t,y(t))(he_1(t), he_1(t))\Big]
   \end{aligned}$$

Die ...-Terme entfallen, weil e_1 Lösung der obigen ODE ist.
Für die Terme dritter Ordnung erhalten wir:

$$= h^3 \left(\underbrace{\frac{1}{6}y'''(t) + \frac{1}{2}e_1''(t) + e_2'(t) - f'(t, y(t))e_2(t) - \frac{1}{2}f''(t, y(t))(e_1(t), e_1(t))}_{} \right) + \mathcal{O}(h^4)$$

Der angegebene Term muß wieder Null werden. e_2 muß also Lösung der ODE dieses Ausdrucks sein.
Wegen den gleichen Argumenten wie oben existiert so eine Funktion.

Wir können durch den ...-(Pseudo)Beweis den Rest zeigen und lassen einfach einmal verbotenerweise den Induktionsschritt weg.
Damit gilt

$$\hat{y}_{n+1} = \hat{y}_n + h\hat{f}(t_n, \hat{y}_n) = \hat{y}_n + hf(t_n, \hat{y}_n) + \mathcal{O}(h^{N+2})$$

2. Wir machen nun von diesem Ausdruck eine Stabilitätsanalyse.
Wegen $\hat{y}_0 = y_0$ und dieser Formel erhalten wir für den lokalen Fehler:

$$\hat{y}_1 - y_1 = \mathcal{O}(h^{N+2})$$

Weil f lokal Lipschitz ist (wegen $f \in \mathcal{C}^\infty$), kann man Lady Windermeres Fächer anwenden und erhalten:

$$\hat{y}_n - y_n = n\mathcal{O}(h^{N+2}) = \mathcal{O}(h^{N+1})$$

□

5.98 Algorithmus (h-Extrapolation)
Wir nehmen eine Grundschrittweite H und $h_j = \frac{H}{n_j}$, üblicherweise $n_j = j$.

Wir extrapolieren nun das Euler-Verfahren $y_{n+1} = y_n + h_j f(t_n, y_n)$ und berechnen zuerst $T_{j1} = y(t, h_j)$ und erhalten über die übliche Extrapolation:

$$T_{j,k} = T_{j,k-1} + \frac{T_{j,k-1} - T_{j-1,k-1}}{\left(\frac{n_j}{n_{j-k+1}}\right) - 1}$$

Wir wissen aus der Extrapolation

$$T_{j,k} - y(t_0 + H) = \frac{(-1)^k}{n_j \cdot \ldots \cdot n_{j-k+1}} e_k(t_0 + H)H^k + \mathcal{O}(H^{k+1})$$

Je weiter wir also im Schema laufen, desto höher ist die Ordnung unseres Verfahrens.
Genauer gesagt erhält man Runge-Kutta-Verfahren.

5.14 Verfahren von Gragg

5.99 Motivation
Unser Ziel ist es, explizite numerische Verfahren zu finden, für welches der Fehler eine asymptotische h^2-Entwicklung hat.
Dazu betrachten wir eine Idee, die wir bereits in der numerischen Differentiation gesehen haben:

$$\frac{y(t+h)-y(t-h)}{2h} = y'(t) + h^2 e_1(t) + h^4 e_2(t) + \ldots + h^{2N} e_N(t) + \mathcal{O}(H^{N+2})$$

wobei wir e_i erwarten, die möglichst glatt sind.

5.100 Konstruktion
Wir betrachten die symmetrische explizite Mittelpunktsregel:

$$\frac{y_{k+1}-y_{k-1}}{2h} = f(t_k, y_k)$$

damit $y_{k+1} = y_{k-1} + 2hf(t_k, y_k)$

Uns stellt sich also das Problem, daß die h^2-Entwicklung nicht „zerstört“ wird.
Gragg hat dann dafür das explizite Euler-Verfahren benutzt und erhält:

$$y_1 = y_0 + hf(t_0, y_0)$$

5.101 Algorithmus
Sei $H = h\ell$ für ℓ gerade und $\ell \in \mathbb{N}$ und wir betrachten den folgenden Algorithmus:

1. $y_1 = y_0 + hf(t_0, y_0)$
2. $y_{n+1} = y_{n-1} + 2hf(t_n, y_n) \quad n = 1, \ldots, \ell$
3. $S_\ell = \frac{1}{4}(y_{\ell-1} + 2y_\ell + y_{\ell+1})$, dies ist ein sog. *smoothing-step*, der nicht notwendig ist.

5.102 Satz (asymptotische h-quadrat-Entwicklung)
Sei $f \in \mathcal{C}^\infty$ und y_n durch den vorigen Algorithmus berechnet, dann gilt

$$y_n = y(t_n) + h^2 a_1(t_n) + h^4 a_2(t_n) + \ldots + h^{2N} a_N(t_n) + \mathcal{O}(h^{2N+2}) \qquad n \text{ gerade}$$
$$y_n = y(t_n) + h^2 b_1(t_n) + h^4 b_2(t_n) + \ldots + h^{2N} b_N(t_n) + \mathcal{O}(h^{2N+2}) \qquad n \text{ ungerade}$$

dabei sind $a_i, b_i \in \mathcal{C}^\infty$ mit $a_i(t_0) = 0$, aber $b_i(t_0) \neq 0$

Stichwortverzeichnis

H

I

K

L

M

N

O

P

Q

R

S

T

V

W

Z